HSLU Lucerne University of Applied Sciences and Arts

Engineering and Architecture

Linköping Electronic Conference Proceedings No. 218

16th Modelica & FMI Conference Proceedings

September 8 – 10, 2025
Lucerne University of Applied Sciences and Arts (HSLU)

Modelica Association



Linköping Electronic Conference Proceedings No. 218

Proceedings of the 16th International Modelica&FMI Conference
Lucerne, Switzerland, September 8-10, 2025

Editors:
Dr. Dirk Zimmer (DLR) and Prof. Ulf Christian Müller (HLSU)

Published by:
Modelica Association and Linköping University Electronic Press

ISBN: 978-91-8118-266-8
Series: Linköping Electronic Conference Proceedings, No. 218
ISSN: 1650-3686
eISSN: 1650-3740
DOI: https://doi.org/10.3384/ecp218

Organized by:
Modelica Association
c/o PELAB, IDA, Linköpings Universitet

S-58183 Linköping
Sweden

and

Lucerne University of Applied Sciences and Arts (HSLU)
Engineering and Architecture
Institute of Mechanical Engineering and Energy Technology IME
CH-6048 Horw
Switzerland

Conference location:
Lucerne University of Applied Sciences and Arts
Campus Zug-Rotkreuz,
Suurstoffi 1,
CH-6343 Rotkreuz,
Switzerland

Copyright © Modelica Association, 2025



Dear Participants of the 16th International Modelica & FMI Conference,

Our biennial conference is the primary event for our community to share the latest developments in the Modelica Association's standards. With numerous advancements since the last conference, there is much to discuss. Here are the key milestones achieved:

- **FMI** now supports over 250 simulation tools, reinforcing its position as the world's leading standard for model exchange and co-simulation.
- The first version of new layered standards, notably **FMI-LS-BUS** for network communication, has been released. This concept enables decentralized standard development.
- **SSP 2.0** has been launched, enhancing its value for industrial applications. Explore its impact through dedicated user presentations and tutorials.
- **Modelica Standard Library (MSL) v4.1** is out, offering critical, backward-compatible improvements over v4.0. MSL development will now shift to a continuous integration model.

I extend my gratitude to our members, especially the Modelica Association project leaders, for their dedication in driving this progress. This surge in activity is reflected in the conference program. With over 100 scientific papers submitted, we carefully selected those for presentation. It's exciting to see diverse domains covered, including new fields like pharmacology and marine applications. As our standards gain traction in industry, we've introduced a dedicated track for industrial user presentations, welcoming contributors from fields like wastewater treatment and aerospace.

While the Modelica Association began with a single modeling language, it now supports a suite of standards for modeling and simulation. This is evident in the record 15 tutorials offered at this conference, covering these standards and their applications. Having achieved these milestones, the Modelica Association is setting new goals in three key areas:

International Expansion
Our biennial conference schedule allows for Modelica events in America and Asia between our main conferences. The North American Modelica Users Group (NAMUG) has built a thriving community in the U.S., with the next American conference scheduled for Atlanta, Georgia, October 12-14, 2026. In Asia, where Modelica is gaining popularity, particularly in China, we are forming a local team. To strengthen this community, we proudly announce the 2026 Asian Modelica Conference from September 21 to 22 in Hangzhou, China. The Modelica board is committed to ensuring robust representation for our Asian members.

Cross-Layered Standards
Layered standards, initially developed for FMI, enable decentralized creation of sub-standards for specific applications without altering the main standard. These often involve enhanced meta-information relevant to other standards like SSP and the Modelica language. The Association will work to harmonize these efforts, fostering cross-layered standards applicable to both FMI and Modelica.

Education Initiative
Equation-based modeling requires specialized training. While many professors and consultants provide this, the Association's teaching materials need updating. We invite educators to join us in organizing these resources. Additionally, developing simpler, more robust libraries will lower the entry barrier. Many tool developers are also reducing this barrier by offering web-based services with user-friendly interfaces. The Modelica Association supports these efforts to advance global education in modeling and simulation.

Concluding Remarks
Recent advancements in artificial intelligence (AI) are disrupting information technology, including modeling and simulation. Tools like AI-driven co-pilots and neural network components are transforming how we program and interact with software. While it's hard to predict how Modelica and its standards will evolve alongside AI, their core value remains:

- Open-source libraries based on open standards are invaluable for both humans and AI.
- Widely applicable standards for model exchange and co-simulation benefit both humans and AI.
- Standardized formats for model parameterization and results are equally essential for humans and AI.

In these dynamic times, we must remain grounded in our mission: creating useful mathematical descriptions of the world and making them accessible, executable, and shareable through open standards.
I am confident that the 16th International Modelica & FMI Conference will showcase outstanding examples of this mission. I wish you all an inspiring and productive exchange!

Dirk Zimmer **and** **Ulf Christian Müller**
(Chairman of the Modelica Association) *(Local Organizer at HSLU)*





DOI
10.3384/ecp218

Platin Sponsor



Gold Sponsors









Silver Sponsors

















s

Keynote Speeches



Physics-Informed AI

AI is increasingly being used in the fast and accurate simulation of physical systems. In this keynote, we will discuss how knowledge of the underlying physics can be explicitly incorporated into AI systems for physical simulations. Key examples will highlight the potential gains as well as the involved challenges.

Opportunities and Challenges in Design and Operation of Integrated Energy Systems



The complexity of modern integrated energy systems demands the systematic use of systems engineering methods and tools to address key challenges across the product lifecycle. This keynote will explore three related and critical areas:

1) the demand for diverse model fidelities and analysis,
2) maintaining consistency across design layers, and
3) the importance of seamless tooling and integration.

Real-world, HVAC specific examples will highlight how these challenges are being tackled in practice and the opportunities they present. For over two decades, Carrier has relied on Modelica as a cornerstone technology for model-based product design. The journey began with the development of control systems for transportation refrigeration, where transient simulations allowed for rapid control function development and verification.

Today, while controls development remains a vital application, Carrier has expanded Modelica's use across the entire product lifecycle – from conceptual design, through testing and verification, to business sales tools, and into operation with monitoring and diagnostics. Design and operation of highly integrated energy systems such as data centers and district heating systems also calls for new methods and tools. Differentiated applications has driven diversified analysis with Modelica models. Modelica-based steady-state simulation has emerged as a back-bone in product design, sales tools, as well as in field diagnostics. Steady-state and transient optimization is also a necessary element across the product life cycle. The need for diversified analysis and computation is further amplified by application of systems engineering methods where requirements, design space exploration, and validation and verification are key elements.

Panelists

For the first time at the International Modelica Conference, we will have a Panel discussion. The topic is:
Open Standards vs. Open source, Collaboration or Conflict?

We choose panelists from a broad range of companies and organizations using Modelica for their product design and research. Our distinguished Panelists are:



Adam Moravanszky, Senior Director Simulation Technology, NVIDIA. At NVIDIA, Adam leads teams advancing open source simulation projects such as PhysX and Newton, and chairs the Physics Working Group of the Alliance for OpenUSD (AOUSD) open standard.

DOI
10.3384/ecp218



Clément Coïc, Technology Cluster Lead - Digital Mechatronics, Siemens Healthineers. Clément leads innovation projects that investigate the value and interconnection of new digital technologies for Siemens Healthineers purposes. A specific focus is carried into leveraging system simulation for better control or path planning of robotic systems, enhancing Machine Learning models with Physics, and modeling the physical behavior of PhysicalAI developments. His works heavily relies on open standards, like Modelica and FMI, and open-source modules - like Julia's SciML ecosystem, ROS or OpenCV and Yolo, to name a few.



Dirk Zimmer, Chairman of the Modelica Association. Expert for equation-based modeling and simulation at the Institute of Robotics and Mechatronics, German Aerospace Center (DLR). The open Modelica standards provide a real technical, economical as well as societal benefit. Promoting them in Dirk's role as Chairman is hence an honor. Also, DLR profits from them: in the field of robotics the multi-domain simulation and control aspects play a crucial role for further advancement.



Marius Dupuis, CEO, ASAM e.V. The CEO's role at ASAM is to manage a team that keeps things running, kicks off new endeavors, and makes sure we are seen and heard. Marius' role is right in the midst of system simulation and (open) standards. It comprises getting members on board, motivating them to initiate and conduct standardization projects, providing a roadmap and legal framework that fit the market, and keeping the association in a good shape.



Michael Wetter, Computational Senior Scientist, Lawrence Berkeley National Laboratory (LBNL). Leading the development of modular modeling, design and controls based on Modelica and FMI for building and district energy and control systems. Michael Wetter has been leading the Modelica Buildings Library which is used in industry and academia for the design of HVAC equipment, building and district energy systems, and data center cooling systems. He also spearheaded the development of digitalization of building control design-installation-commissioning, leading to the proposed ASHRAE Standard 231P, expected to be approved within the next few months, which uses a subset of Modelica to standardize control logic formulation in a vendor-independent format.



Hans-Martin Heinkel, Project Leader prostep SmartSE, and Bosch expert in MBSE and simulation-based engineering. Hans-Martin has more than 35 years of experience in systems engineering, system simulation, Hardware-in-the-Loop, and the integration of simulation into development processes. His main focus, both within Bosch and in the prostep project, is on collaboration based on simulation-supported engineering.

Organizing Committees

Program Chairs
Prof. Dr. Ulf Christian Müller, University of Applied Sciences and Arts Lucerne, Switzerland
Dr. Dirk Zimmer, German Aerospace Center, Germany

Program Board
Prof. Francesco Casella, Politecnico di Milano, Italy
Prof. Lars Mikelsons, University of Augsburg, Germany
Dipl.-Math. Christian Bertsch, Robert Bosch GmbH, Germany
Dr. Hubertus Tummescheit, Model Based Innovation LLC, USA
Dr. Rui Gao, RIGO TECH Co. Ltd., Japan
Prof. Martin Sjölund, Linköping University, Sweden

Program Committee
Dr. Dai Araki, Toshiba Digital Solutions Corporation, Japan
Prof. Bernhard Bachmann, Hochschule Bielefeld - University of Applied Sciences and Arts, Germany
Dr. John Batteh, JuliaHub, USA
Prof. Albert Benveniste, INRIA, France
Dipl.-Math. Volker Beuter, VI-grade GmbH
Dr. Torsten Blochwitz, Keysight Technologies Deutschland GmbH, Germany
Dr. Scott Bortoff, Mitsubishi Electric Research Laboratories, USA
Prof. Robert Braun, Linköping University, Sweden
Dr. Dag Brück, Dassault Systèmes AB, Sweden
Dr. Johannes Brunnemann, XRG Simulation GmbH, Germany
Dr. Benoît Caillaud, IRISA / INRIA, Rennes, France
Dr. Yan Chen, Pacific Northwest National Lab, USA
Prof. Christoph Clauss, Fraunhofer IIS EAS Dresden, Germany
Dr. Clément Coïc, Siemens Healthineers AG, Germany
Dr. Edo Drenth Volvo, Autonomous Solutions, Sweden
Dr. Hilding Elmqvist, Mogram AB, Sweden
Dr. Olaf Enge-Rosenblatt, Fraunhofer IIS EAS Dresden, Germany
Prof. Gianni Ferretti, Politecnico di Milano, Italy
Leo Gall, LTX Simulation GmbH, Germany
Dr. Rui Gao, RIGO TECH Co., Ltd, Japan
Prof. Cláudio Gomes, Aarhus University, Denmark
Prof. Anton Haumer, OTH Regensburg, Germany
Dr. Dan Henriksson, Dassault Systemes
Dr. Yutaka Hirano, Woven Planet Holdings, Inc., Japan
Dr. Andreas Junghanns, Synopsys, Germany
Dr. Daeoh Kang, IVH, South Korea
Prof. Jiri Kofranek, Charles Univesrity, Czech Republik
Dr. Christopher Laughman, Mitsubishi Electric Research Laboratories, USA
Dr. Oliver Lenord, Robert Bosch GmbH, Germany
Prof. Alessandro Maccarini, Aalborg University, Denmark
Dipl.-Ing. Pierre Mai, PMSFIT, Germany
Prof. Antonello Monti, RWTH Aachen University, Germany
Prof. Thierry Stephane Nouidui, The United African University of Tanzania, South Africa
Prof. Christoph Nytsch-Geusen, Universität der Künste Berlin, Germany
Dr. Hans Olsson, Dassault Systèmes, Germany
Prof. Martin Otter, DLR, Institute of System Dynamics and Control, Germany
Prof. Maurizio Palmieri, Dept. of Information Engineering, University of Pisa, Italy
Dr. Kaustubh Phalak, Ingersoll Rand, USA
Dr. Adrian Pop, Linköping University, Sweden
Dr. Katrin Proelss, Airbus Operations GmbH, Germany
Dipl.Ing. Clemens Schlegel, Schlegel Simulation GmbH, Germany
Dr. Klaus Schuch, AVL List GmbH
Dr. Michael Sielemann, Hensoldt, Germany
Prof. Arne Speerforck, TU Hamburg, Germany
Dr. Wilhelm Tegethoff, TLK-Thermo GmbH, Germany
Dr. Bernhard Thiele, DLR, Germany
Prof. Alfonso Urquia, Universidad Nacional de Educación a Distancia (UNED), Spain
Prof. Luigi Vanfretti, Rensselaer Polytechnic Institute, USA
Dr. Antoine Viel, Siemens DISW,
Dr. Volker Waurich, TU Dresden, Germany
Dr. Michael Wetter, Lawrence Berkeley National Laboratory, USA
Dr. Stefan Wischhusen, XRG Simulation GmbH, Germany
Prof. Liu Yanfang, Beihang University, China

Local Organizers
Prof. Dr. Ulf Christian Müller, University of Applied Sciences and Arts Lucerne
Ivo Steiner, University of Applied Sciences and Arts Lucerne

Content

General Modelica **17**

Modelica Tool Development *17*

Improved Unit Inference and Checking in Modelica 17

Implicit Unit Conversion in Modelica 27

Model Disambiguation Technology in MWORKS.Sysplorer 39

Data Reconciliation for Industrial Experiments 45

Chemics, Pharmacology and Medicin *55*

Chemical 2.0 (Free open-source Modelica library) 55

Pharmacolibrary - Free Library to Model Pharmacology 61

Combining static and dynamic optimization approaches for path planning, with collision … 73

Digital Twin *81*

Prototypical Control for the Digital Twin of Aircraft Environmental Control System 81

Aspects and Ideas for the FMI-based Modeling of Railway Digital Twins 91

Collaborative Digital Twin Development for Railway Braking and Traction Applications 101

Simulation and Optimization *109*

Diagnosing Newton's Solver Convergence Failures in the Initialization of Modelica Models 109

Modelica2Pyomo: a tool to translate Modelica models into Pyomo optimization models 117

Enhancing Collocation-Based Dynamic Optimization through Adaptive Mesh Refinement 127

Modeling Paradigms and Language Experiments *139*

Validation of Dynamic Simulation Models using Metamorphic Testing and Given-When-Then …... 139

The Value of Enforcing a Strict Modeling Methodology within Modelica 147

Context-Oriented Equation-based Modeling in ModelingToolkit.jl 155

Context-Oriented Modelica for Advanced Variability Management 165

New Translation Methods and Tools *175*

Benchmarking the Modular Structural Analysis Algorithm 175

Resizable Arrays in Object-Oriented Modeling 189

Constant Time Causalization using Resizable Arrays 203

Modiator - A Web App for Modelica Simulation 211

Energy **223**

Power System Simulation *223*

OpenIWPI: Open-Instance Wave-Phasor Interface Library for Power System Simulation Studies … 223

Expanding an Open-Source Modelica-Compliant Package of Generic Renewable Energy Source … 233

Analytical Treatment of Hollow Toroid Flux Tubes 243

Master controller concept for power flexible energy systems 253

Thermal Management for Green Energy Systems *263*
Modelling, Simulation and Validation of thermal propagation for 3D discretized battery … 263
Development and Experimental Validation of an Unglazed Photovoltaic-Thermal Collector … 273
Modelling Aquifer Thermal Energy Storage (ATES) System with Buoyancy Flow 283
Media Property Modeling *293*
TAeZoSysPro: A Modelica Library for Thermal Aeraulic and Buildings Thermodynamics Calculations 293
Development of a Refrigerant Mixture Package for Dynamic Simulation of Auto-Cascade Refrigeration: …..... 303
A Generic Non-Miscible Liquid-Gas Medium Model in Modelica with Analysis of Incompressibility … 313
Pumps and Vapor Compression *325*
Centrifugal Pump Model of the DLR Thermofluid Stream Library..... 325
Frost/Defrost Models for Air-Source Heat Pumps with Retained Water Refreezing Considered 335
A Dynamic Analysis of Refrigerant Mass in Vapor Compression Cycles..... 345
Energy Generation Systems..... *353*
Modelling and Impact of Hydraulic Short Circuit Operation in Pumped Hydro Energy Storage..... 353
Introducing the NewLib Library and its application to multi-level, large-scale solar …..... 363
Further Application of Modelica-Based Nuclear Power System Simulation: Tasks in Different …... 373
An Open-Source Industrial-Grade Collection of Renewable Energy Source Generic Models … 383
Fuel Cell Modeling and Control *395*
Dynamic modelling of an Ammonia to Power application at high efficiency using a solid … 395
Direct Ammonia Solid Oxide Fuel Cell Stack: Modelling and Experimental Validation..... 405
Dynamic Simulation of a PEM Electrolysis System 413
Model-Based Control Design for a Multi-Stacks SOC System..... 425
Control & AI 435
Modelica & AI *435*
Towards Integration of PeN-ODEs in a Modelica-based workflow..... 435
Efficient Training of Physics-enhanced Neural ODEs via Direct Collocation and Nonlinear …..... 445
Status of the SMArtInt Library: Simple Modelica Artificial Intelligence Interface..... 459
Zero-Shot Parameter Estimation of Modelica Models using Patch Transformer Networks 469
Robotics *479*
Integration of Physical and AI Models Using Open and Interoperable Standards: A Model-Based … 479
Modelica FMI based hybrid reinforcement learning enhanced trajectory planning for an … 489
Automatic Modelica Package and Model Generation from Templates and Data Files with Python, … 497
Control for HVAC and Buildings *505*

Modelica Meets ASHRAE: Towards A Digital Standard for Building Control 505
CDL-PLC translator: From Modelica HVAC control design to IEC 61131 PLC implementation..... 515
Yet Another Residential District Simulator: yards for Controller Development in the Residential …. 525
FMI for Embedded Systems and Virtual Prototyping..... 535
Facilitating the use of Physics-Based Simulations on Embedded Devices by running FMUs …..... 535
FMI Meets SystemC: A Framework for Cross-Tool Virtual Prototyping..... 545
Hybrid Simulation Models for Embedded Applications: A Modelica and eFMI approach..... 555
Control- and AI-based Methods with FMI for Automotive 567
A Study on Vehicle Suspension Loads Prediction Method Based on Hybrid Road Simulation … 567
Br(e)aking the Boundaries of Physical Simulation Models: Neural Functional Mock-up Units … 575
VDCWorkbench: A Vehicle Dynamics Control Test & Evaluation Library for Model and AI-based … 585
Simulation of Embodied Cyber Physical System Based on Modelica/MWORKS: A Case Study of … 591
Control Applications in Modelica 603
On the precise and efficient representation of industrial controllers in Modelica..... 603
Master controller for offshore wind power and hybrid grids 611
The FlightControl library for aircraft control design applications 619
Quasi-Periodic Feedforward Control Based on Inverse Model Tabled FFT..... 629
FMI and Related..... 639
FMI Tool Development..... 639
Liaison: an open-source tool for distributed co-simulations 639
A Tool for the Implementation of Open Neural Network Exchange Models in Functional Mockup … 645
Input Smoothing for Faster Co-Simulation using FMI..... 653
Comparing the Predictive Event Handling Algorithm LookAhead to Rollback and Early Return..... 663
Layered Standards..... 671
Towards a Common Standard for Uncertainty Quantification..... 671
LS-SA: Developing an FMI layered standard for holistic & efficient sensitivity analysis … 681
FMI Layered Standard for Network Communication: Applications in Networked ECU Development693
FMI for Energy Systems 703
An innovative heterogeneous modeling approach to build a cooling system for battery thermal …..... 703
Enhancing Large-Scale Power Systems Simulations through Functional Mockup Unit-based … 713
Design and Simulation Validation of Steam Power Systems Based on MBSE 727
Workflows in Systems Engineering..... 741
The Fundamental Modeling Practices and Specifications to support the Preservation and … 741
Automation Nation: Taming Complex V&V Workflows 751

Integration of Geometric Tolerance Analysis in System Simulations via Functional Mock-up … 765
Maritime Applications *775*
Shared sea-environment definition and realization for maritime and offshore co-simulations 775
Decreasing Risk in the Design of Large Coupled Systems via Co-Simulation-Based Optimization … 783
ShipSIM: A Modelica Library for Ship Maneuverability Modeling and Simulation 791
Modelica driven development of the thermal management control system for a zero emission …..... 801
Automotive *809*
Race Car Braking System Thermal Model for Real Time Use in a Driving Simulator 809
Development of a Multi-Physical Simulation Platform for Durability Prediction for Hyundai …..... 819
Solid-State Battery-Systems and Thermal Management for Electric Long-Distance Buses 831
Poster 841
A low complexity physics-based aging model for lithium ion cells with solid electrolyte …..... 841
Integrating a Seasonal Thermal Energy Storage FMU in a MATLAB/Simscape Thermal Source … . 851
Scalable Higher-order Nonlinear Solvers via Higher-order Automatic Differentiation 861
MultiEnergySystem: A Modelica Library for Dynamic Modeling and Simulation of District …..... 871
Calibration of a Chiller Modelica model with experimental data 881
A Dynamic Simulation Model of Outdoor Swimming Pool with Thermal Energy Storage, Boiler … 891
Railway Marketplace for Data, Know-How and Services 901
Dynamic Simulation of Off-Grid Energy Island with Wind-PV-Storage Hydrogen Production 909
From Simulation to Reality: Deployment of Reinforcement Learning-Based Neural Network … 921
Modeling and Simulation of a Direct Heat Recovery System for Cabin Heating in Battery-Powered … 931
Physics-Based Dynamic Modeling of Solar-Powered Off-Grid Cold Storage for Perishables …..... 939
Selective Evaluation of RHS during Multi-Rate Simulation 943
Absolut Modelica library 949
Identification and Elimination of Instabilities During Simulation of Highly Stiff Vehicle …..... 961
Requirement Verification with CRML and OpenModelica 971
A Thermal Digital Twin of Asphalt Pavements: Implementation and Application to an Instrumented … 981
Status of the TransiEnt Library: Transient Simulation of Complex Integrated Energy Systems 989
Dynamic modeling of a liquid piston compressor system including conjugate heat transfer 1001
Rumoca: Towards a Translator from Modelica to Algebraic Modeling Languages 1009
Safe and Efficient Control of a Brayton Cycle Heat Pump Using Reinforcement Learn.....1017

DOI 10.3384/ecp218

Author Index

Aarsæther, Karl Gunnar	775
Abdelhak, Karim	109, 203
Adusumalli, Vijaya-Bhaskar	91
Ahmic, Kenan	585
Aka, Julius	653
Akbar, Shariq	851
Alvarez-Rodríguez, Jose María	741
Andersson, Joel	1009
Angerer, Johannes	841
Arias, Eliécer	981
Arnold, Tim	921
Assmann, Uwe	155, 165
Ataollahi, Narges	405
Bachmann, Bernhard	109, 127, 203, 445, 943
Bajand, Amin	671
Bartosz, Marcin	611
Basumallik, Sagnik	713
Bayer, Peter	851
Becker, Christian	989
Belforte, Luca	841
Belot, Igor	363
Bemporad, Alberto	469
Benedikt, Martin	663
Bensmann, Astrid	353
Benthin, Jörn	989
Benveniste, Albert	175
Berthold, Matthias	545
Bertsch, Christian	693
Besbes, Karim	703
Bex, Lucas	525
Beyers, Inga	353
Bhattacharjee, Srijita	233
Biegler, Lorenz T.	117
Blanco Mula, Blas	101
Blaud, Pierre	891
Blervaque, Hubert	313
Blochwitz, Torsten	681
Boldo, Didier	45
Bolognese, Michele	395, 405, 645
Borel, Pascal	45, 293, 881
Bortoff, Scott	345
Bosbach, Nils	545
Bosio, Fatma	1017
Bosio, Félix	425
Braun, Robert	751
Brembeck, Jonathan	555, 585
Briant, David	819
Brück, Dag	653
Brun, Joshua	921
Brunnemann, Johannes	459, 801, 989
Buffoni, Lena	671, 971
Bünte, Tilman	629
Bürger, Christoff	555
Busch, Alexander	263
Caillaud, Benoît	175
Canon, Gustavo	981
Cansu Yücel, Fanli	909
Carral, Luis	793
Casella, Finn	263
Casella, Francesco	109, 117
Chakrabarty, Ankush	469
Chedid, Teddy	363
Chen, Liping	39, 909
Chen, Yan	505
Chen, Ying	591
Chen, Yinqi	39
Chen, Zhipeng	39
Chiaramello, Marco	383
Christiansen, Søren Waagø	253
Chupin, Antoine	293
Coïc, Clément	73
Colleoni, Arnaud	819
Condie, Micah	1009
Dahash, Abdulrahman	851
Dashtaki, Mohammad Ali	713
Davelaar, Frans	45
de Castro, Marcelo	223
De Pascali, Matteo Luigi	117
Deconinck, Geert	525
Deligant, Michael	1001
Delmas, Pierre	363
Deshpande, Vedang	345
Devaprasad, Karthik	505
Dhumane, Rohit	303
Ding, Ji	909
Dong, Yuanhui	909
Du, Lin	591
Düsing, Martin	325
Edelman, Alan	861
Egsgaard Kallesen, Thomas	253
Ehrlich, Paul	505
Eiden, Jörg	459, 801
El Feghali, Joy	383
Elizondo Arrieta, Fabian	981
Elmqvist, Hilding	189, 211
Enferad, Erfan	535
Ensbury, Theodor	811
Ericson, Liselott	931
Fachini, Fernando	233
Fernández Ballesteros, Maria Dolores	793
Fish, Garron	819
Flesch, Robert	459

Forberger, Marco 901
Forgione, Marco 469
Fournier, Nathan 363
Franke, Martin 383
Franke, Rüdiger 611
Gao, Hui 727
Garbay, Louis 383
Garcia, Pierre 363
Gautier, Antoine 505
Gebhart, Raphael 325
Gehring, Rainer 961
Gejo, Staša 961
Genser, Simon 663
Gillner, Markus 283, 989
Gonnelle, Alexis 363
Goppert, James 1009
Gorman, Dan 303
Götz, Sebastian 155
Gräber, Manuel 653
Guironnet, Adrien 383
Gundermann, Julia 681
Gutsche, Christian 155
Gutsche, Christian 155, 165
Hafeez Saeed, Muhammad 525
Hagemeier, Anne 989
Hällqvist, Robert 671, 751
Han, Xingjian 727
Hanke, Tim 459, 989
Hanke-Rauschenbach, Richard 353
Hannebohm, Philip 109, 445, 943
Heckmann, Andreas 91
Hégo, Axelle 425
Hellmuth, Jan Friedrich 831
Helsen, Lieve 273, 525
Henningsson, Erik 653
Heyer, Annika 989
Hippmann, Gerhard 101
Hofmann, Andreas 435
Hooshyar, Hossein 713
Hu, Jianjun 505, 515
Huang, Dongping 303
Huang, Lei 727
Huang, Zhichao 39
Hui, Lixin 591
Hyun, Minsu 567
Jansen, Jelger 273
Jardin, Audrey 45
Jardin, Audrey 45, 971
Jarmolowitz, Fabian 575
Jezek, Filip 61
Jumayev, Kahramon 693
Jung, Jaewung 819
Jünger, Lukas 545
Junghanns, Andreas 693
K S, Rajkumar 303
Kamp, Tobias 555
Kärnell, Samuel 931
Klemets, Kristian 139
Kofranek, Jiri 61
Köhler, Jürgen 831
Krebeck, Lukas 353
Krus, Petter 751
Kulhanek, Tomas 61
Kyllingstad, Lars 785
Lagerstedt, Fabian 931
Lamas Galdo, Maria Isabel 793
Lambert, Quentin 27
Langenkamp, Linus 127, 445
Larsson, L. Viktor 671
Laughman, Christopher 345, 469
Lazzari, Riccardo 871
Lebrun, Victor-Marie 819
Lefrancois, Maxime 535
Lenord, Oliver 653, 671
Lepais, Corentin 81
Letournel, Etienne 363
Leupers, Rainer 545
Leva, Alberto 603
Li, Yuxin 909
Lienig, Jens 765
Liu, Jian 727
Liu, Mingjia 591
Liu, Qi 39, 727, 909
Liu, Wei 727
Ma, Jiacheng 335
Ma, Peiquan 591
Mahmoudi, Amirhoushang 939
Mai, Pierre R. 693
Malandain, Mathias 175
Manngård, Mikael 139
Marsollier, Félix 293, 313
Martelli, Emanuele 117
Martinelli, Emanuele 395, 405
Masannek, Marco 73
Matejak, Marek 55, 61
Mathonnière, Sylvain 425
Mazurié, Baptiste 45
Meertens, Lone 273
Meierhofer, Alexander 901
Menne, Benedikt 693
Mesa-Moles, Luis Corona 45
Miao, Keming 861
Mikelsons, Lars 435, 575, 681, 961
Momesso, Alberto 811
Montsarrat, Christophe 881
Moulouel, Rafik 293
Mourtaji, Imad 891
Mst, Ajaykumar 741

Müller, Reiko 621
Müller, Simon 851
Müller, Ulf Christian 921
Muro, Marcelo 871
Mutter, Sylvan 921
Nahodovic, Elmir 653, 671
Najafi, Masoud 693
Nepveu, Francois 363
Nielsen, Rene Just 253
Nirmala, Nirmala 405
Nowotny, Jochen 901
Nyström, Rasmus 611
Oldemeyer, Carsten 497
Olsson, Fredrik 639
Olsson, Hans 17, 671, 681
Östh, Jan 639
Otter, Martin 189, 211, 671
Paez, Ana 881
Parquier, Philibert 383
Pfitzer, Tim 693
Picarelli, Alessandro 811, 819
Piga, Dario 469
Pignède, Antoine 497
Pinto de Castro, Ricardo 585
Plavec, Luka 383
Pollak, Markus 413, 831
Pop, Adrian 671, 971
Poßeckert, Alexander 91
Pourramezan, Reza 713
Prakash, Anand 505, 515
Pratticò, Luca 395, 645
Prieto, Quentin 819
Pritoni, Marco 505
Prokopets, Volodymyr 155, 981
Puente Varela, Basilio 793
Qiao, Hongtao 345
Rackauckas, Christopher 861
Raigné, Guillaume 363
Rein, Johannes 555
Reiner, Matthias 489
Reiterer, Stefan H. 901
Reynaud, Tom 535
Ribas Tugores, Carles 949
Ribbe, Jan 693
Rojas-Ordoñez, Sebastian 479
Romero, Alberto 841
Rommes, Stef 61
Rosenlund, Erik 751
Rösth, Marcus 931
Rouge, Clémence 1001
Ruppert, Stefanie 989
Sadjina, Severin 785
Sahin, Kadir 819
Sanchez-Heres, Luis 639
Sassaroli, Guido 871
Schirmer, Jens 765
Schmidt, Herbert 243
Schmidt, Meik 545
Schmitt, David 851
Schuch, Klaus 693
Schulte, Andreas 831
Schulze, Christian 653
Segura, Mikel 479
Seidl, Christoph 155
Seidnitzer-Gallien, Carina 949
Sergi, Thomas 921
Shahi, Mina 939
Skjong, Stian 775, 785
Solai, Elie 1001
Sommer, Daniel 575
Song, Chen 681
Specklin, Mathieu 1001
Speerforck, Arne 283, 989
Steffen, Tom 989
Steinkellner, Elias 841
Steinmann, Christoph 765
Stellato, Massimo 811
Stromberger, Christopher 819
Stuntner, Benjamin 901
Sudheerbabu, Gaadha 139
Süvern, Markus 693
Tan, Songchen 861
Täuber, Patrick 693
Tegethoff, Wilhelm 413, 831
Testi, Matteo 395, 405, 645
Theel, Finn 801
Thorade, Matthis 335
Thummerer, Tobias 575, 681
Thüring, André 413
Tidefelt, Henrik 27
Tischer, Felix 663
Tobolar, Jakub 585
Tobolář, Jakub 629
Tran, A. Phong 1017
Truscan, Dragos 139
Tummescheit, Hubertus 741
Ultsch, Johannes 585
Urbani, Michele 645
Urbansky, Julian 989
Van Dam, Alje 801
van der Linden, Franciscus L. J. 325
van der Stelt, Teus 109
Van Groos, Boudewijn 801
Vandamme, Antoine 671
Vanfretti, Luigi 223, 233, 713
Verleyen, Lucas 525
Vidovic, Ivan 901
von Ohlen, Carsten 801

Walther, Karl	515
Wang, Tengyue	591
Wang, Xiang	373
Wang, Xiang	373, 591
Wang, Zizhe	155, 165
Watzenig, Daniel	663
Weber, Niels	325
Weber, Robert	961
Westphal, Jan	989
Wetter, Michael	505, 515
Wiegel, Béla	989
Williams, Mark	741
Winter, Christoph	585
Woodbury, Abigaile	1009
Wrede, Konstantin	765
Zahmatkesh, Bahareh Bakhsh	939
Zeißler, Alexander	981
Zeng, Cun	591
Zeng, Ruirui	727
Zhang, Ao	373
Zhang, Haiming	909
Zhang, Shubin	909
Zhang, Ziang	713
Zhou, Chong	39
Zhou, Fanli	39
Zhou, Xuehua	591
Zhou, Zhiguo	591
Zimmer, Dirk	81, 147
Zotter, Gerald	949
Zulueta, Ekaitz	479
Zuo, Li	909

Improved Unit Inference and Checking in Modelica

Hans Olsson[1]

[1]Dassault Systemes AB, Sweden, hans.olsson@3ds.com

Abstract

This paper will present a new unified algorithm for unit checking and inference, and showing the benefits for various libraries.

The Modelica Language supports declaring units for variables using the SI-standard. This allows dimensional checking to detect possible errors in equations. The units for variables make it easier to interpret, input and plot their values. When we infer the unit of a variable we get the same benefits also for variables without a declared unit. We will use unit inference and checking for the combination, even if the check is primarily a dimensional check.

Both dimensional checking and unit inference are already implemented in several Modelica tools, but not consistently. The original motivation for this paper was to understand the different approaches, and demystify the unit handling with the goal of making it more available. Based on that understanding, this paper will also present a new unified algorithm combining the different strengths, and showing the results for various libraries.

Keywords: unit, Modelica, Hindley-Milner

1 Unit background

This will explain the various considerations for the units of variables and unit equality in expressions.

1.1 Allowed units

The stated goal in Modelica is that variables should use SI-units – for the unit-attribute, the displayUnit-attribute is not considered in this paper. The SI-system (BIPM 2019) is based on seven base quantities, and every quantity is a product of them raised to various powers (called the dimension of the quantity). If all powers are zero it is called dimensionless. The seven base quantities correspond to the seven base units of the SI-system, and all units are directly expressed – without any conversion factors. This eliminates some causes of errors, provided only SI-base-units are used. However, when equations contain different prefixed SI-units (e.g., mm instead of m or g instead of kg[1]), or SI-acceptable units like bar or kWh the dimensional analysis does not suffice, and that is an important aspect that will be considered. Additionally to directly quote (BIPM 2019)

> "In practice, with certain quantities, preference is given to the use of certain special unit names to facilitate the distinction between different quantities having the same dimension. ... Even though torque has the same dimension as energy (SI unit joule), the joule is never used for expressing torque."

Plane angles (with unit radian "rad") and solid angles (with unit steradian "sr") are a supplementary units in the SI-system, and even if they are useful for human understanding they are formally dimensionless. Since radians are important we follow Mattsson and Elmqvist (2008) and the FMI-standard (see (Modelica Association 2014)) and treat them similarly as the seven base units when possible – but allow mismatches.

It has also been proposed to add new orthogonal non-physical base quantities like currency to Modelica; the approaches in this paper support them as well.

1.2 Unit Strictness

This paper builds on the proposal from Casella (2016) for unit checking. The basic idea is that literal values inside multiplicative expressions are dimensionless, while literal values on their own can have any unit. That has been found to be a pragmatic solution that allows checking and infering units in a large number of existing Modelica models without any changes. Consider

```
parameter SI.Capacitance C=2; // Ok
SI.Energy E1=C*v^2/2; // Ok
SI.Energy E2=C*v*4; // Error
SI.Voltage v;
```

On the first line `2` gets the desired unit, but on the next line two `2` are dimensionless. The `E2` line is rejected even if we could find a unit for `4` that makes it correct.

Note that for this proposal the equal sign does not fully represent equality, since replacing `C` by `2` in another equation leads to unit errors – or incorrect unit inference. In particular this is an issue if the model was originally constructed by manually substituting expressions based on equality.

The approaches described in this paper work with other rules as well. Alternatives include allowing numeric literals to have any unit everywhere except for `1/x` see (Mattsson and Elmqvist 2008), the same without the exception, or only allowing the zero literal to have any dimension see (Kennedy 1996). The future work will contain some proposals to modify the rules, both to strengthen them for some common cases, and weaken them for experimentally derived equations.

[1]Remember that the base-unit for mass is kg

1.3 Arrays

Most arrays are homogenous, i.e., all of their elements have the same unit. However, there are some arrays that are heterogenous – in particular the state of the transfer-function (each element is the derivative of the previous one) and polar coordinates as an array of two elements (magnitude and angle). Thus this paper will allow arrays to be heterogenous, but try to infer the common unit if it exists and/or infer units for specific elements (for array elements with literal subscripts – they are seen as scalars).

One can consider variants of this, one would be to enforce that all arrays are homogenous, and another would be to perform all unit inference on the scalar level. Neither extreme seems acceptable, but the first approach will be discussed more later.

Instead the solution is to view an array built as `C=[A,B]` as only having a unit if it is homogenous (the implications will be discusssed later). Similar considerations apply to subscripted elements of arrays.

The subscripts themselves are integers and should thus be dimensionless.

1.4 Unit for variables

The general idea is that a variable has a statically determined unit, or is free to have any unit. That may seem obvious, but apart from heterogenous arrays some libraries (Zimmer, Meißner, and Weber 2022) have variables getting different units depending on an evaluable parameter. Another case are constants such as `eps` – that are treated like small literal values. Trying to deduce a unit would be especially problematic for package constants, because the same constant can be used in unrelated parts of a model.

For algorithms (in particular in functions) one could imagine that one variable is re-used to store values with different units – we will consider that an error. Other common cases where one variable have multiple units are s-parameterizations, where positive and negative values have different units – that can be handled by making both cases dimensionless in the model, see (Mattsson and Elmqvist 2008). An important consideration is that variables are free to not have a unit, this allows building blocks that can be re-used with different units.

2 Approaches to unit inference and checking

There are multiple approaches to unit inference and checking in various programming languages, both related to the nature of the language and to the goals of the check.

2.1 Simple unit inference and checking

In traditional imperative programming languages it is straightforward to propagate unit forwards from variables, so that the unit of `z=x*y` gets the unit of `x` multiplied by the unit of `y`, and addition will just check that the two terms have the same unit, and give the result that unit. This has been implemented multiple times, see (Hilfinger 1998) for an early variant for ADA.

This propagation up-wards is insufficient in an equation-oriented language such as Modelica. Dymola has traditionally extended this with also propagating the unit down-wards, i.e., if we know the units of `z` and `x` in `z=x*y` we conclude the unit of `y` is the unit of `z` divided by the unit `x`. The basic framework for unit checking in Dymola is described by Mattsson and Elmqvist (2008) including down-ward propagation. The unit-checking in Dymola has after that paper built on that framework and can be configured in different ways. In this paper the rules for literals (literals have any unit, except for `1/x` where `1` is dimensionless) are replaced by the stricter rules from Casella (2016) that only allow literals to have any unit in additive contexts so that we can for `z=3*x+2` propagate the unit from `x` to `z`. Specifically the multiplicative literal 3 is dimensionless (allowing propagating the unit), but the additive literal 2 has any unit (avoiding a unit error).

A practical application of this can be seen in the CoupledClutches example for `sin1`, which simplified has:

```
  RealInput torque.tau(unit="N.m");
equation
  sin1.y = sin1.offset + sin1.amplitude*sin
      (2*pi*sin1.f*time);
  connect(sin1.y, torque.tau);
```

Here we propagate “N.m” through the connection and then down-wards to `sin1.offset` and `sin1.amplitude` (knowing that the output of `sin` is dimensionless).

This approach needs to be iterated (normally just a handful of times), and at first seems ad-hoc.

It does, however, work fairly well in practice. The benefits of the approach are:

- The results are understandable, for each inferred variable one can directly see which equation that the unit originates from. That is useful for analyzing errors for unit checking where having a clear understanding of how the various variables got their unit helps in finding the error.
- For unit inference it can preserve the exact units used; so if a variable has unit “N.m” variables that are equal to it can also get unit “N.m” instead of “kg.m2.s-2” or the even more confusing “J”. This is also important when not using SI-base-units, e.g., “mol/l”. Apart from "N.m" being required by the SI-standard for torque this also works together with non-SI displayUnits (suggesting “kcal” or “kWh” as display unit for a torque is not good).
- Simple and consistent handling of arrays variables:
  - We can treat homogenous arrays as scalars, and if we infer a unit for an array that is valid for the entire array. Conversely, if we know that array is heterogenous we can treat it as having unknown unit (see (Mattsson and Elmqvist 2008)).

DOI 10.3384/ecp21817

  - It is straightforward to limit the unit inference to going in one direction (conditional unit assignments). An example where this is useful is an equation involving array indexing. If the entire array has one unit that should also be the unit for the array element, but in order to support heterogenous arrays the inference cannot go from unit of an array element to unit of the entire array.
  - It is also straightforward to handle a `C=[A,B]` such that if we have a unit for `C` we can infer the unit for `A` and `B`, and if `A`, `B` both have a unit and it is the same, we can infer that `C` has the same unit. It is deliberate that if only `A` has a unit we do *not* assume that `B` (and `C`) must have that unit as well.
  - Array elements with literal indicies can be treated as scalars.

That it works for arrays is not limited to simple cases like adding two vectors with units, but also extends to scalar products, cross products, and matrix multiplications.

2.2 Hindley-Milner unit inference

The Hindley-Milner (or Damas–Hindley–Milner) unit inference traces it origins to the Hindley-Milner algorithm for type inference, and is presented as Damas-Milner by Kennedy (1996) (an alternative reference is his PhD thesis from the same year). It has previously been discussed for Modelica by Broman, Aronsson, and Fritzson (2008) and proposed by Lambert and Tidefelt (2024). We will go through it in detail, but it basically collects all unit-equations and in one sense solves them one by one and then substitutes the result in the other equations (and substitutions), until all of the unit equations have been handled.

The benefits of this approach are:

- Uses all of the unit information and nothing more, i.e., it is complete and consistent.
- Known convergence and practically linear.

2.2.1 Details

The first step in this approach is to collect all of the unit equations as equations.

Consider the model

```
model Elec
  Real pot1(unit="V"), pot2(unit="V");
  Real v, i;
  Real R(unit="Ohm");
equation
  v = i * R;
  v = pot2 - pot1;
end Elec;
```

The variable `pot1` is called p_1 below, and similarly for `pot2`. This gives the unit-equations including substitutions for known units (we use `u[v]` for the unit of `v` as a symbolic value, and reserve `v.unit` for the corresponding string).

$$u[p_1] \to V = kg \cdot m^2 \cdot s^{-3} \tag{1}$$
$$u[R] \to \Omega = V \cdot A^{-1} = kg \cdot m^2 \cdot s^{-3} \cdot A^{-1} \tag{2}$$
$$u[v] = u[i] \cdot u[R] \tag{3}$$
$$u[v] = u[p_1] \tag{4}$$
$$u[v] = u[p_2] \tag{5}$$

In practice it makes sense to have all unit-equations in the form $1 = \ldots$ – to ensure that they are normalized, $u[x]$ is the unit or dimensionality of the variable `x`, and substitutions are used for *solved* equations. The substitutions are seen as solved even if they may depend on other (unknown) units. Variables with unit-modifiers are also represented as substitutions.

Another example, which will be discussed later, is:

```
model SecondOrder
  input Real u;
  parameter Real w, D, k(unit="1")=1;
  Real y(unit="1"), yd;
equation
  der(yd) = w*(w*(k*u-y)-2*D*yd);
  yd = der(y);
end SecondOrder;
```

The declaration of k gives:

$$u[k] \to 1 \tag{6}$$

The declaration of y gives:

$$u[y] \to 1 \tag{7}$$

The subexpression `k*u-y` gives:

$$u[y] = u[k] \cdot u[u] \tag{8}$$

Introduce α_1`=w*(k*u-y)` giving:

$$u[\alpha_1] = u[w] \cdot u[y] \tag{9}$$

The subexpression α_1`-2*D*yd` gives

$$u[\alpha_1] = u[D] \cdot u[yd] \cdot \tag{10}$$

The equation `der(yd)=w*(`$\alpha_1 - \ldots$`)` gives:

$$u[yd] \cdot s^{-1} = u[w] \cdot u[\alpha_1] \cdot \tag{11}$$

The equation `yd=der(y)` gives:

$$u[yd] = u[y] \cdot s^{-1} \tag{12}$$

Equations 8 to 11 all derive from one Modelica equation. The variable α_1 is only introduced as part of unit-checking and may more accurately be described as only introducing $u[\alpha_1]$ corresponding to the unit of `w*(k*u-y)`, but not α_1 itself. Using such intermediate variables makes the construction of the unit-equations more straightforward even for complicated equations. Additionally they are needed in the proposed combined algorithm for some cases. They can trivially be excluded when reporting the result of the unit inference.

One could even introduce $\alpha_2 = k \cdot u - y$ replacing (8) by $u[\alpha_2] = u[y]$ and $u[\alpha_2] = u[k] \cdot u[u]$, and for (9) use $u[\alpha_1] =$

$u[w] \cdot u[\alpha_2]$. However, since $u[\alpha_2] = u[y]$, one can use $u[y]$ as above.

This example will be used to illustrate that Hindley-Milner can give additional units compared to the simple algorithm (which cannot find the units for `D` and `w`).

If performed naively a second intermediate step (normally included in the first step) solves some issues:

- Combine exponents for the same variable, which may even cause a variable to disappear from the unit equation (if it appears on both sides of the equation with equal exponents).
- To support `sqrt` and `nthRoot` the original unit-equations may contain rational exponents for the unit-variables, to simplify the analysis each unit-equation is raised to a suitable power to get integer coefficients (either for just the unit-variables or also extended to known units).

The third step is to take these unit-equations (in any order) and for each find a unit-variable with the exponent that is smallest in magnitude. As listed by Kennedy (1996) there are number of cases:

- There is no unit-variable, just check that equation is unit-correct.
- There is only one unit-variable remaining, find its unit and substitute its unit in the other unit-equations and substitutions.
- There are multiple unit-variables, and the smallest exponent is +/-1 or more generally the smallest exponent divides the exponents of all other unit-variables. In this case solve for such a unit-variable (as a product of fixed units and a product of other unit-variables).
- The remaining gcd-case where we introduce a new unit-variable by dividing the other exponent as much as possible, and forming a new unit-equation with the remainders, and then iterate on this unit-equation until we get to one of the other cases. This will convergence in a finite number of steps and can be seen as way of finding the greatest common divisor of those exponents, see Appendix B for details.

As an alternative to the last step one could allow rational numbers as exponents for variables, and use the penultimate step in all remaining cases (Lambert and Tidefelt 2024). The main disadvantage would be that it spreads the rational numbers.

The only really difficult part of this algorithm is understanding that it actually works, and implementing the gcd-case (or using rational numbers). Actually working mean that the algorithm finishes and it either produces a unit error, or the same unit inference (independent of the order of the unit-equations) and that the results are consistent. The gcd-case is the only concern for showing that the algorithm finishes, and that relies on the convergence of the Euclidean algorithm for computing the gcd of integers. Consistency also imply that if we add any of the inferred units for the variables to the unit-equations and re-run the algorithm it will produce the same result.

For the Elec-case the most efficient way would be to solve the second equation for $u[v]$ and then the first equation for $u[i]$, but we will show that we can solve them in the given order, which means that there is no need to describe an optimal order:

substituting the known units gives

$$u[v] = u[i] \cdot \Omega \tag{13}$$

$$u[v] = V \tag{14}$$

solving the first (13) for $u[v]$ gives

$$u[v] \to u[i] \cdot kg \cdot m^2 \cdot s^{-3} \cdot A^{-1} \tag{15}$$

substitution this in (14) gives

$$1 = u[i]^{-1} \frac{kg \cdot m^2 \cdot s^{-3} \cdot A^{-1}}{kg \cdot m^2 \cdot s^{-3}} = u[i]^{-1} \cdot A \tag{16}$$

solving this gives

$$u[i] \to A \tag{17}$$

substitution this in (15) gives

$$u[v] \to A \cdot kg \cdot m^2 \cdot s^{-3} \cdot A^{-1} = kg \cdot m^2 \cdot s^{-3} \tag{18}$$

and subsitution in the two remaining equations

$$u[p_1] \to u[v] = kg \cdot m^2 \cdot s^{-3} \tag{19}$$

$$u[p_2] \to u[v] = kg \cdot m^2 \cdot s^{-3} \tag{20}$$

And thus `v.unit="kg.m2.s-3"` (or `v.unit="V"`), and `i.unit="A"`. As noted above solving (14) from $u[v]$ and then (13) for $u[i]$ is an alternative giving the same result.

For SecondOrder we can start by handling the trivial cases to get:

$$u[k] \to 1 \tag{21}$$

$$u[y] \to 1 \tag{22}$$

$$u[u] \to 1 \tag{23}$$

$$u[yd] \to s^{-1} \tag{24}$$

$$u[\alpha_1] \to u[w] \tag{25}$$

and the non-trivial equations (10) and (11) gives:

$$u[w] = u[D] \cdot s^{-1} \tag{26}$$

$$s^{-2} = u[w]^2 \tag{27}$$

the non-trivial part (26) becomes

$$u[w] \to u[D] \cdot s^{-1} \tag{28}$$

substitution in (27) and simplifying gives

$$1 = u[D]^2 \tag{29}$$

solving (29) and substituting in (28) gives

$$u[D] \to 1 \tag{30}$$

$$u[w] \to s^{-1} \tag{31}$$

DOI 10.3384/ecp21817

The key part is that the substitutions combine `D` and `w` into one equation that can be solved, something that the simple algorithm cannot do. In particular the simple algorithm does not have the unit equation (27) due to how the equations are written, but would have it for the equivalent equation `der(yd)= w^2*(k*u-y)-2*w*D*yd;`

If we remove the unit for `y` we would not infer units for `yd`, `u`, and α_1, but the inferred units for `D` and `w` would be unchanged.

An equivalent approach is to solve integer equations for the units (this is a traditional approach in dimensional analysis - and also seen in this context by Karr and Loveman III (1978), for the connection see (Kennedy 1996)) -– those integer equations correspond to taking the logarithm of the unit-equations used by Hindley-Milner. It is important to note that the equations are integer equations and treated as such -– using the same approach for solving systems of equations with real coefficients is not numerically robust.

For this case the left-hand side of the integer equations are:

$$\begin{pmatrix} +1 & & & & & & \\ & +1 & & & & & \\ +1 & -1 & +1 & & & & \\ & -1 & & +1 & & & \\ & +1 & & & +1 & -1 & \\ & & & -1 & +1 & +1 & \\ & & & +1 & & -1 & 1 \end{pmatrix} \cdot \begin{pmatrix} \log(u[k]) \\ \log(u[y]) \\ \log(u[u]) \\ \log(u[yd]) \\ \log(u[w]) \\ \log(u[\alpha_1]) \\ \log(u[D]) \end{pmatrix} \tag{32}$$

They have been arranged to be in block-lower-triangular form, where the non-trivial block corresponds to solving for a combination of variables.

3 Comparison of the approaches

We will both investigate the relationship between the two approaches in theory, and then practically investigate it for the Modelica Standard Library.

3.1 Theoretical considerations

If we only consider scalars then the Hindley-Milner approach can infer more units, and thus also detect more errors.

There are two reasons for this:

- The combination of exponents for variables can infer the unit of `x` from the unit of `x*x` and to all variants of this (the simple algorithms handles `x^2` and extending it to detect `x*x` is trivial). Disappearing variables allows deducing the unit of `T` from `T*der(x)=x` (`T` has unit "s"), and also from `r*x+x` (`r` has unit "1"); without requiring a unit for `x`. Basically we for the first case get $u[T] \cdot u[x] s^{-1} = u[x]$, and divide both sides by $u[x]$ giving $u[T] = s$.

- Substitution means that if we have a damping factor `D` and a frequency `w` and the equations `der(yd)= w*(w*(k*u-y)-2*D*yd)` and `yd=der(y)` we can find the unit for both `D` and `w`.

The similarity between the two approaches is that by a suitable ordering of the unit-equations we can view the simple inference as prematurely stopping the Hindley-Milner algorithm. Basically we stop before we get to any of the problematic cases where we would *need to* substitute a unit-variable depending on another unknown unit-variable, or combine unit-exponents. This has some implications consequences:

- The simple algorithm gives a consistent sub-set; not a different result.

- By continuing with Hindley-Milner after running the simple algorithm we clearly see which additional unit inferences (and possible errors) it gives.

There are some caveats here:

- It is only the case for a suitable ordering, the Hindley-Milner can process the unit-equations in any order, and thus first find that the unit of `z` depends on the unit of `x`, and after substituting this solve another equation giving the unit for `x`.

- It is only the case if the unit-equations are created in a suitable way, specifically to handle `z=a*b+y` the simple algorithm would be stuck if we add the equations $u[z] = u[a] \cdot u[b]$ and $u[a] \cdot u[b] = u[y]$ instead of $u[z] = u[a] \cdot u[b]$ and $u[z] = u[y]$. A solution is to introduce a new unit-variable α_1 and $u[\alpha_1] = u[z]$, $u[\alpha_1] = u[a] \cdot u[b]$, $u[\alpha_1] = u[y]$, since it ensures that as soon one of the three unknowns get a unit it can be propagated to the other two (in this case we can skip the extra variable and reuse `z` or `y`, but in general all of them could be complicated expressions). Hindley-Milner can handle these cases without new variables.

- It also requires that we don't convert known units to base-units in Hindley-Milner during that part (in order to get the benefit of the original units). In theory this could be disastrous for performance, but in practice it does not seem problematic and if it is a problem it could be limited to units with a few factors.

- Combining arrays and Hindley-Milner causes problems. This will be discussed later.

Keeping the original units (i.e., not converting to base-units) in the normal Hindley-Milner would be more complicated due to the repeated substitutions, that problem practically disappears due to the suitable ordering.

For unit-errors keeping the original units allows the error message to include both the original units and the result converted to base-units; where either may be more helpful depending on the circumstances.

Additionally, this suggest that the algorithms can be unified, i.e., first running the part of Hindley-Milner corresponding to the simple algorithm – even extended to combining variables. That will automatically preserve most of the benefits of the simple algorithm (locality and understandable unit strings), and then later running the complete algorithm for the small additional gain – but restricted to scalars.

Instead of iterating all equations for the simple algorithm one could keep track of where unit-variables are used and after updating the impacted ones for each substitution check those substitutions to check whether there is at most one unknown unit in that equation. That ensures that the simple algorithm is practically linear, similarly as Hindley-Milner.

3.1.1 Array issue

Lambert and Tidefelt (2024) indicate that there are issues for arrays by limiting it to scalars, but we want to make the issues clear. The obvious issue is that conditional equalities for arrays (for subscripting and concatenation) do not naturally fit the unification based on equality.

However, they can be handled by introducing an unknown array variable (in case the array occurred inside an expression) and after substitutions check if either that array has a unit, or the other side has homogenous units and only then add it as an equality. It is not known whether this will give good results when substituting unit-variables depending on others.

There is also a worse problem that is less obvious, consider `T*der(x)=A*x;` For scalars we can see that `A.unit="1"` (regardless of the unit of `x`), and simply applying the algorithm would give the same result for a matrix `A`. However, if we have `x.unit={"m","s"}` then we should have `A.unit=["1","m/s";"s/m","1"]` (i.e., only the diagonal has unit 1). Solutions to this would be to scalarize before performing unit inference and checking, or adapting the algorithm to handle the various kinds of array equations. In practice both variants would make the implementation considerably more complicated as one needs to ensure that scalarization does not perform any simplification that would impact unit inference or checking. Neither approach has been implemented.

Note that the problem does not occur for *all* array expressions. E.g., if `A` is a scalar then the simple result is correct regardless of whether `x` is a scalar, vector, or matrix. Similarly, for `v1*x=v2*x` or `v1.*x=v2.*x` we can eliminate `x` to conclude that `v1` and `v2` have the same unit(s).

However, even if non-homogenous arrays exist in some models having an option to test enforcing homogenous array is useful — the idea would be to discover missed units, and manually add them to the model.

3.2 Practical difference

The previous section showed that it is possible to first run the simple algorithm, and then extend it to the Hindley-Milner algorithm (for scalars). During these tests we naively keep the unconditional array equations in the second step.

This means that we can isolate the impact of the advanced parts. This has been implemented and tested on the Modelica Standard Library (MSL, specifically on (Modelica Association 2025b)) and the experience was:

- The Hindley-Milner approach is actually easier to implement and also easier to get right.
- The reason is that for the simple unit inference one needs to implement unit-inference going both "up" and "down". If there are a large number of different operations that can be substantial work, and instead Hindley-Milner only needs to go "up" and collect equations in a standardized form.
- The most complicated part of implementing Hindley-Milner is the gcd-case, and that does not occur in MSL (it is unclear if it occurs for any actual models).
- No new unit-errors were reported by the algorithm compared to the simple algorithm (there are a number of known unit-issues in the library — even if they are being reduced).
- Apart from squared variables, it detected new units for variables in seven models that could not be found by the simple algorithm; three of them were useful – three of less practical use, and in one case the results were incorrect (due to how the model was constructed – it did not indicate a problem with the algorithm).

There are two kinds of squared variables, one was propagating a unit down for `V=sqrt(re*re+im*im)` for electrical models. The other occurs if we keep array equations and gives that `T` has unit "1" from `r1=transpose(T)*T*r2` (where `r1` and `r2` are positions). The latter is the only case where naively keeping array equations made any difference. In this specific case the unit is correct (and the derivation is even correct in general for this specific array expression), but given that `T` is the orthogonal transformation matrix we even know that $T^T \cdot T = I_3$, and thus that `transpose(T)*T` does not matter in that equation. This indicates that at least for MSL skipping array equations in the second part of the full algorithm has neglible impact.

The useful results were for models such as SecondOrder which has `der(yd)= w*(w*(k*u-y)-2*D*yd)`, where we can find the unit for both `D` and `w`; even if we cannot find either unit directly.

The ones of less practical use were UniformNoise from the Clocked part which has `y = u + noiseMin + (noiseMax - noiseMin)*noise;` where we see that `noise.unit="1"`. The unit for that variable is of limited use as it is a protected variable (so no impact on other variables and not intended for plotting), and `unit="1"` isn't useful for unit conversion.

DOI
10.3384/ecp21817

The one with incorrect result was the most interesting and it is the Motor model used in the R3Robot example where we have:

```
  parameter Real w=4590 "Time constant of
      motor";
  parameter Real D=0.6 "Damping constant of
       motor";
...
  Modelica.Electrical.Analog.Basic.Inductor
       La(L=(250/(2*D*w))) ...;
  Modelica.Electrical.Analog.Basic.Resistor
       Ra(R=250) ...;
  Modelica.Electrical.Analog.Basic.Resistor
       Rd2(R=100) ...;
  Modelica.Electrical.Analog.Basic.
      Capacitor C(C=0.004*D/w) ...;
```

Just applying the algorithm gives the conclusion that $u[D] \rightarrow \Omega^{-1}$ (or Siemens), and $u[w] \rightarrow s^{-1}$, which is based on assuming that literal numbers have unit "1" inside equations (but unknown unit when they appear alone). Since the unit is seemingly correct for `w` it would be tempting to consider the entire result correct. However, looking carefully shows that 0.004=1/250, and an obvious thought is that all three uses of 250 likely have unit Ohm. Testing this by replacing 0.004 by 1/250 and then replacing those three uses of 250 by a common parameter gives the result $u[D] = 1$, $u[w] = s^{-1}$; and a damping constant 0.6 (with unit "1") make perfect sense.

The SI-standard, (BIPM 2019), recommends against "s-1" due to the ambiguity of whether it is a frequency ("Hz") or angular frequency ("rad/s"), numerically they differ by a factor of 2π. However, it is not possible to resolve this ambiguity automatically and thus the non-recommended "s-1" is safest. Manual checking (and also the name `w`) shows that it is an angulary frequency, "rad/s". The extra factor "rad" may seem to be unit-incorrect, but this supplementary unit is dimensionless, and thus the capacitance and inductance are unit-correct even if they do not contain a factor "rad".

Normally the parameter with value 250 would explicitly be declared as a resistance parameter, but that could also be inferred. A pull-request has been prepared that declares `w` and `D` with proper types with units, and also add the resistance parameter; as that is a more understandable model. Just adding the resistance parameter would still require Hindley-Milner to infer the units for `w` and `D`.

3.3 Impact of unit inference and check

The new algorithm was tested on different libraries and the variables were split into different categories. Variables in common models are counted multiple times, but the special variable `time` was excluded. Only models that passed unit check were considered for the inference statistics.

Library	MSL		MSL
Version	4.0.0		3.2.3
Scalars	336899		329763
Declared	238268	(70.7%)	239798
Simple	9171	(2.7%)	9325
Advanced	44	(0.01%)	39
Arrays	73553		55222
Homogenous	57312	(77.9%)	38384
Heterogenous	191	(0.3%)	308
Simple	4869	(6.6%)	1035
Advanced	0		0
Elements	40	(0.05%)	55
Tot. Classes	6299		5954
Inference	1083		931
Errors	67		147
Adv. Errors	0		2

And some smaller libraries for comparison.

Library	ThermoFluid-	VehicleInterfaces
Version	Stream 1.1.0	2.0.1
Scalars	43163	9692
Declared	33582	4287
Simple	1268	143
Advanced	0	0
Arrays	3198	10579
Homogenous	2774	8508
Heterogenous	0	0
Simple	15	712
Advanced	0	1
Elements	0	0
Tot. Classes	2156	252
Inference	106	28
Errors	40	14
Adv. Errors	0	0

The "simple" are scalar and homogenous arrays inferred by the simple algorithm. The "advanced" refer to the additional ones found by the algorithm proposed in this paper. For MSL the "advanced" cases are the previously mentioned squared variables in electrical models, and the seven additional cases – which are counted multiple times due to model re-use. The "elements" refer to arrays where units of some elements are inferred by the simple algorithm. The "declared" and "homogenous" and "heterogenous" are variables that are declared with units.

Although counting all variables the same is oversimplified it still gives an indication of the impact. The percentages do not fully capture the advantages, e.g., the simple algorithm only inferred the unit for 6.6% of the arrays in MSL 4.0.0, but that represents 30% of the arrays that lacked a declared unit.

The number of arrays where elements have different units (both declared as "heterogenous" and deduced as "elements") underestimates their already small number. The reason is that algorithm is not designed to find them, and e.g., state-space vectors are not detected as having elements with different units.

The "tot. class" include a large number of documentation classes, interface classes (where no inference is pos-

sible), and other classes without units and thus the classes with unit inference is more interesting. The "errors" are the number of classes containing errors, where a single error may be reported for multiple classes. As the unit-proposal is yet to be approved, the "errors" do not necessarily indicate quality problems.

MSL 3.2.1 had two models with advanced errors, they were both due to the gain `k` having unit "1" in the PI-block, which is used in the Controller in the NoiseExamples. That has later been removed in general, and did not indicate any error in the tested model. Note that the number of classes, including the classes with inferred units have increased in the later MSL version – while the number of errors has decreased.

4 Combined algorithm

The array issues imply that a combined algorithm would be to run Hindley-Milner corresponding to the simple algorithm (including arrays — both simple array equations and one-sided; and without combining unit-exponents and substituting unknown units), then combine exponents and run the full Hindley-Milner on the remaining unit equations only involving *scalar* variables. We would generally propose this as it is simple to explain and gives reasonable results.

However, a more detailed analysis shows that also simple equations involving arrays can be kept in the second phase, as long as one only combines exponents for scalars and only uses substitutions with scalars in the right-hand-side. (The reason is that if an array has a unit depending only on the unit of scalar variables multiplied by a fixed unit then it must be a homogenous array.) One could even extend this from only scalars to scalars *and homogenous arrays*, where operations such as `sum(v)`, `max(v)`, or `v*u` (where `u` is homogenous) imply that `v` is homogenous.

As the practical tests only found one case with any benefit from naively keeping array equations the difference between these variants will be minor.

One can construct cases where the difference between these algorithm matters, e.g.,

```
  Real x;
  Real v[3];
  Real w[3](each unit="m");
equation
  v*v=1; // Normalize
  v*x=w;
  ...
```

Considering equations involving arrays is needed to find the unit for `v` and `x`. Without any special logic for `v*v`, we can solve the second equation for $u[v] \to m/u[x]$, and substitute that in the first equation to give $u[x] \to m$ and thus $u[v] \to 1$.

5 Future Work

The most important future work for the Modelica Language is to get an agreement on the unit rules. For tools the important part is implementing these rules.

For libraries (not only MSL) the first part is to remove any unit errors and in some cases incorrect units. A particular consideration is to use rational exponents consistently, both in units-string (added by Modelica Association (2025a)) and in models. For models this implies one should use `sqrt(x)` instead of `x^0.5` and `nthRoot(V, 3)` instead of `V^(1/3)` (also introduced by Modelica Association (2025a)). Automatically handling some common rational exponents (with diagnostics) is possible and helpful for users, but using `sqrt` and `nthRoot` more clearly shows the intent.

Unit handling in equations involving arrays could be investigated further. An obvious improvement would be to prove that some operations require homogenous input (such as `sum` and `cross`) and view their inputs as homogenous *everywhere*.

Combining this approach with the analysis on subcomponents for partial result as by Broman, Aronsson, and Fritzson (2008) would be interesting. Having the possibility to store the inferred units in the model seems like a natural possibility for the future, but it seems redundant and if too automated it may include dubious units like “s-1” instead of either “rad/s” or “Hz”.

5.1 Proposed Extensions

Adding a generic type for the dimensionless case to `Modelica.Units.SI` would make it easy to clarify the many cases where a variable is known to be dimensionless (there are already a number of types for *specific* dimensionless quantities).

Having the possibility to disable unit-checking for a specific equation, or all equations in a model has been found useful. That basically means that no unit-equations are added for the corresponding equations. Testing of the Buildings 10.0.0 library, (Wetter et al. 2014), similarly revealed a need to disable the the new rules for literals on the package level.

For converting between different prefixed SI-units, or non-SI-unit it would be beneficial to have a function `UnitConvert(a, fromUnit, toUnit, factor)` that just returns `a*factor`, that should be the unit-conversion factor from one unit to another. (Also proposed by Lambert and Tidefelt (2024)) Having a special name it clarifies the intent of convertering between the units, and including the units and the factor it is trivial to implement for tools that don’t implement units – while still allowing tools to verify that the factor is correct. The *intent* is important as one cannot assume any parameter with such a unit is a conversion – e.g., an up-hill road could have slope of 100 mm per m.

6 Conclusions

This paper shows how to unify the simple unit-algorithm from Mattsson and Elmqvist (2008) traditionally implemented in Dymola with Hindley-Milner unit-inference. It shows that this will give the benefits of both – preserving understandable units and locality for the large num-

DOI 10.3384/ecp21817

ber of simple cases – including arrays, while also getting units for all the remaining scalar cases. The understandable units make it easy to input and interpret values for variables without a declared units, including switching to compatible units for display. Any errors will also (with rare exceptions) be localized to specific equations, where the units for all involved variables are easy to trace. Even if the number of additional variables with inferred units is small, it is still worth in terms of providing a stronger theoretical basis – especially considering the effort.

Based on experience in Dymola it is even simpler to implement than the complete simple unit-inference. The algorithm has been test-implemented in Dymola 2025x Refresh 1, and will also part of 3D Experience Platform 2026x.

It also shows that only implementing a conventional Hindley-Milner approach would require more care, both to get good unit strings and either not work for arrays or require scalarization for arrays (without any clear benefit).

Acknowledgements

This work has been supported by VINNOVA (grant number 2023-00969, OpenScaling). The authors would also like to thank the Modelica Language group for pushing improvements for units.

References

BIPM (2019). *Le Système international d'unités / The International System of Units ('The SI Brochure')*. Ninth. Bureau international des poids et mesures. ISBN: 978-92-822-2272-0. URL: http://www.bipm.org/en/si/si_brochure/.

Broman, David, Peter Aronsson, and Peter Fritzson (2008). "Design Considerations for Dimensional Inference and Unit Consistency Checking in Modelica". In: *Modelica 2008*. Vol. 2855. International Modelica Conference. The Modelica Association, pp. 134–155. URL: https://modelica.org/events/conference2008/sessions/session1a1.pdf.

Casella, Francesco (2016-12). *MCP-0027 Units of Literal Constants*. Tech. rep. Cremona: Modelica Association. URL: https://github.com/modelica/ModelicaSpecification/issues/2127.

Hilfinger, Paul N. (1998). "An ADA Package for Dimensional Analysis". In: *ACM Transactions on Programming Languages and Systems* 10.2, pp. 189–203. DOI: 10.1145/42190.42346.

Karr, Michael and David B. Loveman III (1978). "Incorporation of Units into Programming Languages". In: *Communications of the ACM* 21.5. DOI: 10.1145/359488.359501.

Kennedy, Andrew John (1996-04). *Programming languages and dimensions*. Tech. rep. Cambridge: University of Cambridge Computer Laboratory. URL: https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-391.pdf.

Lambert, Quentin and Henrik Tidefelt (2024-03). *Applying Hindley-Milner to unit-checking*. Tech. rep. Wolfram System Modeler: Modelica Association. URL: https://github.com/modelica/ModelicaSpecification/pull/3491.

Mattsson, Sven Erik and Hilding Elmqvist (2008). "Unit Checking and Quantity Conservation". In: *Modelica 2008*. Vol. 2855. International Modelica Conference. The Modelica Association, pp. 13–20. URL: https://modelica.org/events/conference2008/sessions/session1a2.pdf.

Modelica Association (2014-07). *Functional Mock-up Interface for Model Exchange and Co-Simulation Version 2.0*. Tech. rep. Linköping: Modelica Association. URL: https://fmi-standard.org.

Modelica Association (2025a-02). *Modelica – A Unified Object-Oriented Language for Systems Modeling. Language Specification Version 3.7 dev*. Tech. rep. Linköping: Modelica Association. URL: https://specification.modelica.org/master/MLS.pdf.

Modelica Association (2025b-02). *Modelica Standard Library*. Tech. rep. Linköping: Modelica Association. URL: https://github.com/modelica/ModelicaStandardLibrary/commit/7c4858b675d2442cd34c99047ca9fb672083e11a.

Wetter, Michael et al. (2014). "Modelica Buildings library". In: *Journal of Building Performance Simulation* 7.4, pp. 253–270. DOI: 10.1080/19401493.2013.765506.

Zimmer, Dirk, Michael Meißner, and Niels Weber (2022). "The DLR ThermoFluid Stream Library". In: *Electronics* 11.22. ISSN: 2079-9292. DOI: 10.3390/electronics11223790.

A All used unit rules

- Addition, $u[z] = u[x], u[y] = u[x]$ for `z = x + y`, `z = x - y`, `z = x .+ y`, `z = x .- y`, `z = min(x, y)`, `z = max(x, y)`, `z=rem(x, y)`[2], `z = mod(x, y)`[2], and `z = homotopy(x, y)`[2].
- Multiplication[3], $u[z] = u[x] \cdot u[y]$ for `z = x * y`, `z = x .* y`, and `z = cross(x, y)`.
- Division[3], $u[z] = u[x]/u[y]$ for `z = x / y`, and `z = x ./ y`.
- Trigonometic function, $u[z] = 1, u[x] = \text{rad}$ for `z = sin(x)`, `z = cos(x)`, and `z = tan(x)`.
- Inverse trigonometic function, $u[z] = \text{rad}, u[x] = 1$ for `z = asin(x)`, `z = acos(x)`, and `z = atan(x)`.
- Differentation, $u[z] = u[x]/\text{s}$ for `z=der(x)`.
- Dimensionless function, $u[z] = 1$, $u[x] = 1$ for `z = log(x)`, `z = log10(x)`, `z = exp(x)`, `z = sinh(x)`, `z = cosh(x)`, and `z = tanh(x)`.
- Propagate first argument, $u[z] = u[x]$ for `z = -x`, `z = skew(x)`, `z = fill(x, ...)`, `z=hold(x)`, `z = noEvent(x)`, `z = noClock(x)`, `z = min(x)`, `z = max(x)`, `z = abs(x)`, `z = sample(x, ...)` (clocked version), `z = superSample(x, ...)`, `z = subSample(x, ...)`, `z=backSample(x, ...)`, `z = shiftSample(x, ...)`, `z = previous(x)`, `z = scalar(x)`, `z = vector(x)`, `z = matrix(x)`, `z = promote(x)`, `z = transpose(x)`, `z = diagonal(x)`, `z = delay(x, ...)`, `z = pre(x)`, `z = actualStream(x)`[2], and `z = inStream(x)`[2].

[2] Implemented after testing

[3] With special treatment of literals

- Smooth, $u[z] = u[y]$, for `z = smooth(p, y)`.
- Reinit, $u[x] = u[y]$ for `reinit(x, y)`.
- Root, $u[z]^n = u[x]$ for `z = nthRoot(x, n)` (if we can evaluate n) and with $n = 2$ for `z = sqrt(x)`.
- Relations $u[x] = u[y]$ for `z = x > y`, `z = x >= y`, `z = x < y`, `z = x <= y`, `z = x == y`, and `z = x != y`.
- Time-units, $u[z] =$ s, for `z=time`, and `z=interval()`
- Truncated division $u[z] = 1$, $u[x] = u[y]$ for `z = div(x, y)`[2].
- Rounding $u[z] = 1$, $u[x] = 1$ for `z = floor(x)`[2], `z = ceil(x)`[2], and `z = integer(x)`[2].
- Arctan2 $u[z] =$ rad, $u[x] = u[y]$ for `z=atan2(x, y)`[2].
- Semilinear $u[z] = u[m] \cdot u[x]$, $u[x] = u[y]$ for `z = semiLinear(m , x, y)`[2].
- Integer power, $u[z] = u[x]^n$ for `z=x^n` if integer expression n (and ignored unless we can evaluate n).
- Non-integer power, $u[z] = 1$, $u[x] = 1$, $u[y] = 1$ for `z=x^y` if y is not an integer expression.
- Sample, $u[x] =$ s, $u[y] =$ s for `sample(x, y)`[2] (non-clocked version).
- Condition, $u[z] = u[x]$, $u[y] = u[x]$ for `z=`**`if`**` c `**`then`**` x `**`else`**` y` if `c` is not an evaluated expression.
- Only propagate arrays if either side has homogenous unit, $u[z] = u[x_1] = \ldots = u[x_n]$ for `z = {`x_1`, ..., `x_n`}`, `z = [`x_1`, ..., `x_n`]`, `z=[`x_1`; ...; `x_n`]`, and `z = cat(p, `x_1`, ..., `x_n`)` (ignoring `p`).
- Array subscripting `z = x[i]` if `i` is literal then treat $u[x_i]$ as scalar, otherwise only propagate from homogenous $u[x]$ to $u[z]$,
- Special case for literals and `zeros(...)` (the latter to handle **`equation`** `a.f + b.f = zeros(3);`).
- Connect-statements are handled by checking the equations generated from the connection-sets, the details will only impact diagnostics for incorrect models.

B Gcd-case

For completeness we will present the gcd-case for the algorithm, even if we haven't seen any practical models requiring this.

```
model GcdCase
  Real x;
  parameter Real y=1;
  Real area(unit="m2");
  parameter Real T(unit="s");
equation
  der(area)=x^4*y^6;
  T*area=x^4*y^(-6);
end GcdCase;
```

After some substitutions we get the unit-equations:

$$1 = u[x]^4 \cdot u[y]^6 \cdot \frac{s}{m^2} \tag{33}$$

$$1 = u[x]^4 \cdot u[y]^{-6} \cdot \frac{1}{s \cdot m^2} \tag{34}$$

for the first equation we introduce a new variable

$$u[z] \to u[x] \cdot u[y] \tag{35}$$

giving (36)

$$1 = u[z]^4 \cdot u[y]^2 \cdot \frac{s}{m^2} \tag{37}$$

$$1 = u[z]^4 \cdot u[y]^{-10} \cdot \frac{1}{s \cdot m^2} \tag{38}$$

importantly we solve the *same* equation again for y

$$u[y] \to (m^2 \cdot s)^{1/2} \cdot u[z]^{-2} \tag{39}$$

$$1 = u[z]^{24} \cdot \frac{s^4}{m^{12}} \tag{40}$$

giving

$$u[z] \to \frac{s^{1/6}}{m^{1/2}} \tag{41}$$

$$u[y] \to s^{-1/6} \tag{42}$$

$$u[x] \to m^{1/2} \tag{43}$$

The example is as contrived as it looks, but it shows that the case can be handled. The introduced variable has exponents corresponding to truncated integer division for all of the other exponents, which corresponds to one step in the normal algorithm for greatest common divisor. It stops when the new smallest exponent divides all others, and that exponent is the greatest common divisor (in this case 2).

DOI 10.3384/ecp21817

Implicit Unit Conversion in Modelica

Henrik Tidefelt[1] Quentin Lambert[1]

[1]Wolfram MathCore, Sweden, {henrikt,qlambert}@wolfram.com

Abstract

There are many situations in which a Modelica model needs to handle quantities which are not expressed in the often preferred unscaled SI units. Applying correct unit conversions is extremely important in such situations, and the risk of human error needs to be mitigated using unit-aware technology. Considering the power of unit checking mechanisms in several Modelica tools today, one can be surprised that unit conversion in Modelica still needs to be performed using error-prone user-written formulas and functions. It is demonstrated how automatic and implicit unit conversion can be introduced in Modelica, and that this can be done safely. The benefits of this approach are illustrated in a variety of examples and applications.

Keywords: unit checking, unit conversion, Modelica

1 Introduction

1.1 Why model with units?

Units of measurement, in this work simply referred to as *units*, play an important role in the modeling domains where Modelica is used for several reasons. An often cited reason is that attention to units of measurement would have avoided some spectacular engineering failures in the past due to mixing up numbers in metric units with numbers in non-metric units. Another reason is that use of units of measurement allows *dimensional analysis* of programs, allowing certain errors in expressions and equations to be detected during model translation. Units of measurement also enrich the presentation of computed results, and give hints regarding the kind of quantity that computed numbers represent. A key convenience of working with unit aware tooling is also the possibility of separating the units used in numeric computation and storage of results, from the units used to conveniently enter parameter values and display results.

1.2 The use of unit conversion in Modelica

If all units in the world of Modelica models were equivalent to unscaled SI base units, there would be no reason to have support for unit conversion. However, many Modelica models exist in a context where also other units are used, and in such tool chains there needs to be proper support for keeping track of and converting between units in order to avoid the sometimes subtle but catastrophic consequences of interpreting numeric values with an incorrect understanding of the associated units. In a complex environment where a Modelica model is interfaced with surrounding systems, it is hard to imagine a better place for performing the necessary unit conversions than inside of the Modelica model – it would be unreasonable to expect that all of the surrounding systems should be capable of adapting their interfaces to a choice of units governed by the Modelica model.

1.3 Unit conversion in Modelica today

There are two common approaches to performing unit conversions in Modelica models today, namely unit conversion factors (possibly embedded in gain blocks) and unit conversion functions. The Modelica Standard Library contains a selection of unit conversion functions as well as examples of using gain blocks with conversion factors.

The approach of using a dedicated function for every supported unit conversion does not scale well to the general task of performing conversion between any pair of conversion compatible units.

With the conversion factor approach, conversions are often created as needed instead of being provided as a collection of predefined conversions. This makes the method of using `displayUnit = "1"` a viable technique for manually setting the correct value of the conversion ratio – the correct value is 1 when expressed in the unit `"1"`, reflecting that there is no scaling in terms of value of quantity:

```
  Real x(unit = "m");
  Real y(unit = "yd");
  final constant Real m_per_yd(
    unit = "m/yd", displayUnit = "1")
    = 0.9144;
equation
  x = m_per_yd * y;
```

However, when a conversion factor is used in equations like above there is a high risk that the value will rarely be seen expressed in the display unit, making it a potential source of error.

On the other hand, when the conversion factor is embedded in a gain block, the gain block with a gain displayed as 1 may appear as unnecessary clutter to an untrained eye, or to someone who is not familiar with the need for explicit unit conversion in Modelica. If the `displayUnit` is removed, the gain block might appear less like unnecessary clutter due to no longer displaying a gain of 1, but at the cost of no longer making it obvious that the block is merely performing a unit conversion, not actually scaling the value of quantity.

1.4 State of the art

General purpose computer algebra systems with unit handling tend to allow implicit unit conversion. For instance, all of the following computer algebra systems support adding values of quantity expressed in different units: Wolfram Language, Maple (through the Units package), Python (through the Pint package), Julia (through the Unitful.jl package), and JavaScript (through the Math.js library). Further, none of the systems allow adding, say, a length to a numeric value. (*Wolfram Language – Quantity* 2025; *Maple – Tutorials Part 10: Units* 2025; *Pint – Tutorial* 2025; *Unitful.jl – Conversion/promotion* 2025; *Math.js – Units* 2025)

An interesting feature of Unitful.jl which is particularly relevant for the current work is the `FixedUnits` concept. It is a mechanism for locally taking control over the implicit unit conversion by disabling it.

Likewise, in systems for causal modeling, the widespread Simulink software offers implicit unit conversion between connected blocks. There is also a block for explicit unit conversion, but no block for getting automatic unit conversion at a particular place in a model.(*Simulink – Converting Units* 2025)

On the more formal side, Kennedy (1997) argues that one reason for annotating programs with units of measurement rather than physical dimensions is that it enables compilers to automatically inject unit conversions.

When it comes to Modelica, unit handling is less developed, explained partly by the added complexity that comes with not having prescribed causality of computations, but probably partly also due to the difficulties caused by not having a syntax for attaching units to numeric literals. Early developments include Mattsson and Elmqvist (2008) and Aronsson and Broman (2009), but to date the only thing which has been formalized in the Modelica language is the string syntax for expressing units. That said, the use of *display units* is well established, allowing values of a variable to be displayed or entered in a different unit than the unit used for the underlying numeric representation. This means that existing Modelica tools already have capabilities for unit conversion, but these capabilities are not leveraged by the Modelica language.

Several Modelica tools implement unit checking with unit inference, and even though implementation details differ, there are no major controversies regarding whether a model is using units consistently or not. It is an ongoing work to standardize unit checking in Modelica, so that a model considered consistent by one tool will also be considered consistent by other tools. To our knowledge, no Modelica tool is currently supporting a non-standard extension for allowing automatic or implicit unit conversions.

1.5 Limitations

This work does not elaborate the handling of units with offsets – typically temperature units – for which there is a difference in the interpretation of a unit depending on whether a value of quantity is relative or absolute. The limitation is a matter of presentation; the unit handling framework in which the current work has been implemented also supports the distinction between relative and absolute values of quantity, but a discussion of its design in this regard would overshadow the cetral topic of this work.

The unit symbols required to be recognized according to the Modelica specification are not even sufficient to support the Modelica Standard Library – for instance, the unit of the `Pressure_bar` type is the "non-standard" (that is, not supported by the Modelica specification) `"bar"`. In order to more closely relate to real world applications with need for unit conversion, examples in this work also make use of non-standard units such as `"lb"`. All of these and many more are supported out of the box by the computer algebra systems mentioned in subsection 1.4, and the authors believe that Modelica one way or another will need to develop in the same direction. In the meantime, there are several ways in which a Modelica tool could provide support for non-standard units. For example, the unit handling framwork used here supports all units appearing in the Modelica Standard Library and a conservative selection of additional well established units out of the box, and also supports per-library customization through vendor-specific Modelica annotations. It is a limitation of this work that the question of how to ensure a consistent selection of non-standard units across tools is not addressed, but the proposed ideas for unit conversion would remain relevant also with only current Modelica's standard units since the standard supports the use of unit prefixes (as in `"mm"`) as well as several unit symbols for time and volume. For example, `"L/min"` is a standard Modelica unit which might require conversion to, say, `"m3/s"`.

2 Notation and preliminaries

2.1 Notation

Concrete units will be written in the style of their Modelica string representation. For example, `"m"` represents the meter, and `"1"` represents the unit of a dimensionless number.

A *numeric value* refers to a real number without unit. For instance, 1.5 is a numeric value. Numeric value is not the same as dimensionless number, as the latter has a unit of `"1"`.

The combination of a numeric value and a unit is denoted a *value of quantity*. (In other contexts, this may also be referred to as a *value* or a *quantity*, but both these terms already have conflicting established meanings in the Modelica context.)

Objects with unit (values of quantity, variables, expressions, etc) belong to a *quantity space*, which is either an *affine space* or *vector space*. Values of quantity in affine space are often said to be *absolute*, while values in vector space are sometimes said to be *relative*.

DOI
10.3384/ecp21827

A numeric literal is said to have empty unit.

2.2 Simple non-standard operators for unit handling

In order to not distract from the central topic of this work in the following sections, a few simple non-standard operators for unit handling will be taken for granted.

2.2.1 Construction of value of quantity

The non-standard expression `withUnit(`x`, `u`)` constructs a value of quantity with numeric value x and unit u. The expression x shall have empty unit.

Example:

```
Real x = withUnit(1.0, "m");
```

When `withUnit` is used with literal operands, the expression is referred to as a *unitful literal*.

There are several appealing alternatives for using special syntax instead of calling the operator by its name as above. In this work, a unitful literal may be constructed by providing the unit in the form of a quoted identifier following the numeric literal:

```
Real x = 1.0'm';
```

2.2.2 Fixed unit conversion

The expression `inUnit(`v`, `u`)` is used to express the value of quantity v in the unit u. The expression v shall not have empty unit. The unit u shall be given by a constant expression and be conversion-compatible to the unit of v. The `inUnit` is an affine function of its first operand, and a tool can inline calls early to reduce the need for symbolic processing rules.

Example:

```
Real x(unit = "m") = 1.0;
Real y(unit = "yd") = inUnit(x, "yd");
```

This operation only uses the same sort of information that Modelica tools need to possess in order to support display unit conversions, with the difference that the information is needed during model translation rather than when editing values or plotting results.

2.2.3 Extraction of numeric value

The expression `withoutUnit(`v`, `u`)` obtains the numeric value of the value of quantity v expressed in the unit u. The expression v shall not have empty unit. The unit u shall be given by a constant expression and be conversion-compatible to the unit of v. The returned numeric value has empty unit.

For example, this can be used to make a unit-safe function for an empirical relation which only holds for a particular choice of units, while the function interface is free to use a more common choice of units:

```
function empirical
  input Real u(unit = "kg");
  output Real y(unit = "kg") =
    inUnit(y_lb, "kg");
protected
  Real u_lb = withoutUnit(u, "lb");
  Real y_lb =
    someEmpiricalRelationInPounds(u_lb);
end empirical;
```

3 Unit checking machinery

This section gives a very brief overview of the constraint-based approach to unit checking which is the foundation for the extended semantics proposed in this work. It is based on ideas found in Hindley-Milner type systems.(Hindley 1969; Milner 1978) The description in this section describes the machinery without support for implicit unit conversion, currently in use in Wolfram System Modeler for checking unit consistency in agreement with common (still not standardized) understanding of how units should be used in current Modelica. A more detailed description of the machinery is outside the scope of this work, but is available in the form of a pull request within the Modelica Change Proposal MCP-0027 *Unit checking* (currently in the state *In Development*).(Lambert and Tidefelt 2024) Although currently not standardized, the unit semantics resulting from the machinery presented in this section will be denoted the *current semantics*. In subsection 6.2, it will then be shown how the machinery can be extended to support automatic and implicit unit conversion.

In addition to the unit of an expression, the machinery also keeps track of the quantity space. Similar to unit checking, there are rules for quantity space checking, and the rules may be used for inference of quantity space. While details of the quantity space handling are outside the scope of this work, it should be mentioned that it is largely independent of the handling of units. In particular, the constraint-based approach to unit checking described in this work is carried out without modification also when quantity space is considered, but it must be remembered that unit inference does not consider – and hence does not determine – unit offsets. For vector quantity space, unit offsets are irrelevant, and hence units inferred by unit checking are fully accurate. For affine vector space on the other hand, unit offsets matter, and in principle it is a responsibility of quantity space checking to determine them. Alternatively – and this is by far the most common situation and corresponding to the limitations of this work – if an inferred unit has zero offset and all conversion compatible units also have zero offset, then knowing the quantity space is neither needed to conclude that the correct offset would be zero in case of affine quantity space, nor to perform correct unit conversion. Hence, it would be needlessly strict to make unknown quantity space an error in this situation, and it can be seen that quantity space checking can be safely ignored as long as units with offset are not considered.

Unit checking is carried out by solving a set of unit *equivalence compatibility* constraints. The unknowns are a combination of unit variables associated with variables in the model, and auxiliary variables introduced during the

process of collecting the constraints. A constraint is a relation between two unit *meta-expressions*, a very simple expression language with operations for multiplication, raising to a power, and differentiation.

In order to make unit inconsistency an error according to the Modelica specification, it needs to be a tool independent property of a model. To this end, the collection of unit constraints takes place at an early stage of model translation where expressions and equations still have a close and dirrect correspondence to the original Modelica input. (Alternative approaches to unit checking applied at later stages of model translation might be easier to implement, but be difficult to define in such a way that unit consistency would become a tool independent property of a model.)

The literals of the meta-expression language, denoted *meta-literals*, consist of the *concrete* units, the empty unit, and the undefined unit. The concrete units correspond to well-formed Modelica unit strings. A constraint is trivially satisfied if either side is empty or undefined, and the inference will never determine a unit variable as empty or undefined. Hence, after completed unit inference, each unit variable will either be determined as a concrete unit or left undetermined.

The notation $\texttt{var}_\texttt{u}$ is used for the unit variable associated with the model variable `var`.

4 Shift of mindset

4.1 The unit equivalence mindset

In current Modelica there is an unstated mindset based on unit equivalence. For example, it is generally expected that this model shall pass unit checking:

```
Real x(unit = "J/s");
Real y(unit = "W") = x;
```

That is, the unit of `y` (`"W"`) does not need to be written identically to the unit of its binding equation (`"J/s"`).

Adding expressions with equivalent units comes down to just adding numeric values:

```
Real x(unit = "J/s") = 1.0;
Real y(unit = "W") = 2.0;
Real z = x + y;
```

It suffices to assume that there is no unit inconsistency in order to conclude that the numeric value of `z` is 3.0. Units only serve as complementary information to the numeric values, but have no impact on the computation of numeric values. A unit for `z` can be determined by unit inference, and it is not perceived as a problem that the exact form of the unit is not uniquely defined; knowing that the inferred unit for `z` will be equivalent to both `"J/s"` and `"W"` is enough.

When all expressions have units, and units are used consistently, mathematical field axioms are fulfilled. For example, let

```
Real x(unit = "s");
Real y(unit = "s");
Real z(unit = "s");
Real u(unit = "m");
```

Then the following properties hold:

- Commutativity: `x + y = y + x`
- Associativity: `(x + y) + z = x + (y + z)`
- Distributivity: `u * (x + y) = u * x + u * y`

When empty literals are present, there is a tradeoff between convenience and possibility of detecting unit inconsistencies. Standardization of unit checking is pointing in the direction of avoiding the "wildcard effect" of literals in multiplicative expressions:

```
// OK, 1.0 effectively has unit "s":
Real a(unit = "s") = x + 1.0;
// Error, 1.0 effectively has unit "1":
Real b(unit = "m.s") = u * 1.0;
```

Unfortunately, this means that multiplication does not distribute over addition in Modelica:

```
Real c = u * (x + 1.0);    // OK
Real d = u * x + u * 1.0; // Error
```

The semantics proposed in this work will preserve the important properties of commutativity and associativity. Likewise, distributivity will be preserved when all expressions have units.

To fix the problem with general distributivity in Modelica will require another approach with is outside the scope of this work, namely broad adoption of unitful literals, so that all remaining literals written without unit can be assigned the unit `"1"`. For example, `u * (x + 1.0's') = u * x + u * 1.0's'`.

4.2 The value of quantity mindset

As long as units are used consistently, the particular choices of units become less important, and it becomes possible to view the relations in the model in terms of related values of quantity rather than the numeric values that would vary with the choice of units. This mindset requires trust in two things:

- Correctness of annotated as well as inferred units.
- That consumers of translated models pay proper attention to units, annotated as well as inferred.

It is of particular importance that a numeric value together with a unit of measurement is understood as just one of infinitely many equivalent representations of the same underlying value of quantity.

To establish the required trust is beyond the scope of this work. However, recent developments in unit checking in common Modelica tools as well as awareness of gained quality of existing Modelica libraries thanks to unit checking support in tools, is expected to build up such trust over time.

DOI 10.3384/ecp21827

5 From error to automatic resolution

This section presents a sequence of models, giving a natural progression from how an equation which is erroneous according to current semantics, eventually can be interpreted as valid thanks to an implicit unit conversion. Details of the implicit unit conversion will be given in section 6.

5.1 Unit inconsistency is an error

A prerequisite for this work is that inconsistent use of units is an error.

```
  Real x(unit = "m") = 1.0;
  Real y(unit = "yd");
equation
  // Prerequisite that this is an error:
  x = y;
```

Thanks to being an error to start with, it will be possible to define semantics without breaking backwards compatibility.

5.2 Explicit unit conversion

The inconsistency can be resolved using `inUnit` for explicit unit conversion:

```
  Real x(unit = "m") = 1.0;
  Real y(unit = "yd");
equation
  x = inUnit(y, "m");
```

5.3 Explicit automatic unit conversion

A non-standard operator named `autoUnit` will be used to denote explicit automatic unit conversion. The operator is similar to `inUnit`, except that the target unit of the conversion is determined from the context.

Let e be an expression of the form `autoUnit(`e_1`)`, and let unit(e) denote the unit of the expression e. Similar to additive operators, a unit variable is conceptually introduced for unit(e). No unit constraints are associated with the `autoUnit` expressions, but all of the following shall hold after completed unit inference:

- unit(e) is a concrete unit.
- unit(e_1) is a concrete unit.
- unit(e_1) is convertible to unit(e).

After completed unit inference, `autoUnit(`e_1`)` is replaced with `inUnit(`e_1`,` unit(e)`)`.

Applying `autoUnit` to resolve the unit inconsistency above:

```
  x = autoUnit(y);
```

5.4 Implicit unit conversion

Finally, as an alternative to explicitly introducing `autoUnit` calls to resolve unit conflicts, it only remains to let the tool implicitly introduce `inUnit` calls where a unit inconsistency between conversion-compatible units would otherwise result. Considering that different ways of resolving a unit conflict using unit conversion may result in different interpretation of empty unit expressions as values of quantity, it must be ensured that implicit unit conversion does not cause any actual or apparent ambiguity of computed values of quantity. How to ensure this is the topic of section 6.

Now, the equation originally considered to be in error is instead interpreted as having implicit unit conversion:

```
  x = y;
```

6 Well-defined values of quantity

This section describes how the unit checking machinery can be extended to support implicit unit conversion without introducing ill or even poorly defined values of quantity. In order to serve as a viable extension of current Modelica, the design will meet the following criteria:

- No change of semantics for models free of unit errors according to current semantics.
- No actual or apparent ambiguity of computed values of quantity.
- Preserved associativity and commutativity of addition.

The first of these items is obtained by only allowing implicit unit conversion where the current semantics would have detected a unit inconsistency. Therefore, existing Modelica programs free of unit inconsistency will remain unaffected by the introduction of implicit unit conversion in the language.

The last two items are related, and will be addressed in the following.

6.1 The potential source of ambiguity

When a potential unit inconsistency is resolved using `inUnit`, it is important to be aware that different choices may lead to different interpretations of empty unit expressions. Consider the following expression, without allowing implicit unit conversion:

```
1.0'cm' + 0.5 + 1.0'm'
```

The expression has a unit inconsistency which can be resolved using unit conversion. Here, the conflict will be resolved using `inUnit`. It would also be possible to use the more convenient `autoUnit`, but this might have obscured the fact that the problem is unrelated to the automatic choice of unit.

All of the following are valid ways of resolving the unit inconsistency:

```
inUnit(1.0'cm', "m") + 0.5 + 1.0'm'
inUnit(1.0'cm' + 0.5, "m") + 1.0'm'
1.0'cm' + inUnit(0.5 + 1.0'm', "cm")
1.0'cm' + 0.5 + inUnit(1.0'm', "cm")
```

Note how the interpretation of the literal `0.5` is affected by where `inUnit` is applied. This demonstrates that the explicit use of `inUnit` (or `autoUnit`) gives control over the interpretation of empty unit expressions.

This is also where the danger of implicit unit conversion lies; if the inconsistency was automatically resolved by the tool introducing `inUnit` somewhere in the expression, the interpretation of `0.5` would seem ambiguous, especially to someone who expected addition to be commutative and associative. The authors believe that even if a unique choice of unit for `0.5` could be defined without sacrificing commutativity and associativity, the rules for implicit conversion would become too unintuitive to be of practical relevance with human users in mind. Instead, implicit unit conversion needs to be constrained so that the interpretation of empty unit expressions remains as intuitive as in current Modelica.

In subsection 6.2, the rules for implicit unit conversion will be defined so that the problematic cases above can be rejected.

6.2 Extensions to unit checking machinery

This section describes how the unit checking machinery from section 3 is extended to support implicit unit conversion. The unit semantics resulting from the extended machinery is denoted the *extended semantics*.

It is observed that in almost all situations where a unit constraint is introduced according to the current semantics, at least one side of the constraint corresponds to an expression which could be wrapped in `inUnit(...)` to resolve a unit inconsistency. The two exceptions are `unit`-attributes and `connect`-equations.

The `unit`-attribute exception is not a problem, since there are good reasons anyway to constrain the unit inference by giving these constraints priority over all other constraints in the sense that a unit variable is always solved from its `unit`-attribute constraint in case one exists. It follows that these constraints are never resolved using implicit unit conversion. Besides ensuring that `unit` -attributes are obeyed exactly for the variable they belong to, this also provides an important mechanism for delimiting the potential issues associated with allowing implicit unit conversion.

A `connect`-equation constraint is an exception since neither operand of `connect` is allowed to be an `inUnit(...)` expression, and this exception will be dealt with towards the end of this section.

The extended semantics for the non-exceptional cases is that unit constraints are now introduced in the form of *directed* equivalence compatibility of two unit meta-expressions. When x is required to be equivalence compatible to y, the direction from x to y shall indicate that x corresponds to an expression that can be unit converted if needed. When a unit constraint free of unit variables is obtained during inference, it is first checked whether it is equivalence compatible. If it is, there is no need of unit conversion and the constraint can safely be dropped. Otherwise, if it is not convertibility compatible, there is a unit inconsistency that implicit conversion cannot resolve. In the remaining case, implicit unit conversion is tentatively injected on the source side of the constraint in order to establish equivalence compatibility. After completed unit inference, the tentatively introduced conversions are analyzed to ensure that they do not introduce ambiguous values of quantity. This analysis is denoted *robustness analysis* and will be described below.

Computations where all `Real` expressions have units, and the units are used consistently, have uniquely defined values of quantity as outcome – the alternative would have been a serious flaw of the current semantics. Similarly, computations where all `Real` expressions have empty unit, are interpreted as operations on real numbers, and since no units are present, there is no risk of ambiguity due to automatic unit conversions. The risk of ambiguity arises where the two kinds of computation meet, due to the Modelica convention of treating an empty unit expression as if it had a unit which fulfills a unit constraint with a concrete unit on the other side.

In the extended semantics, units determined by the unit inference machinery are only uniquely determined up to convertibility. For example, a variable could end up getting the unit `"J"` or the unit `"kN.m"` depending on how the unit constraints are processed, but the ambiguity in the unit of a variable isn't by itself causing ambiguous values of quantity. The ambiguous values of quantity arise when an ambiguous unit is used to interpret an empty unit expression as a value of quantity.

Ambiguous units originate where a tentative unit conversion is introduced, since it is generally not uniquely defined where the conversion must be introduced. Such a unit constraint is denoted an *origin of ambiguity*. The ambiguity of the constraint gets propagated to any unit variable solved from the constraint. If a unit variable is found to be ambiguous, the ambiguity gets propagated to all unit constraints where the unit variable is present. If the variable `x` has a `unit`-attribute, then x_u does not propagate ambiguity and is said to be *robust by definition*. If the propagation of ambiguity back and forth between constraints and unit variables reaches a constraint where the empty unit is on one side of the constraint, then an ambiguous value of quantity has been detected, a situation which will be denoted a *robustness conflict*.

Thus, robustness analysis comes down to showing that there are no robustness conflicts. This problem is naturally studied on the bipartite incidence graph of unit constraints and unit variables. A constraint where one side is empty or undefined is said to be an *origin of fragility*. Existence of a robustness conflict is equivalent to existence of a path in the graph from an origin of ambiguity to an origin of fragility, after excluding the variables which are robust by definition. Such a path is denoted a *robustness conflict path*.

Note that the origins of ambiguity in a robustness analysis graph depend on the order in which the unit constraints

DOI
10.3384/ecp21827

are processed during unit inference, so other implementations are likely to produce different graphs. However, if one implementation marks a constraint as an origin of ambiguity, another implementation will also mark a constraint as origin of ambiguity within reach from the former constraint. Hence, the existence of a robustness conflict path will not depend on the order in which the constraints are processed.

Returning to the exception of `connect`-equations, the only difference in the handling compared to other equations is that the introduction of `inUnit(...)` to resolve unit conflicts is delayed until the connection set equations have been generated. For instance, the robustness analysis makes no difference between unit constraints coming from `connect`-equations and unit constraints coming from other equations; the need for implicit unit conversion in the constraint of one `connect`-equation will be an origin of ambiguity, and as all the connectors in the connection set will be connected in the robustness analysis graph, the ambiguity will reach the entire connection set. At the time of elaborating the connection set equations, it must be remembered that the established unit consistency of the `connect`-equations is only for convertibility, not equivalence. For each generated flow or potential equation a common unit for the equation is selected among the ones which are already present; this will ensure that no implicit unit conversion in the connection set equation will take place when all units are equivalent. Then all terms of the flow or potential equation are conceptually converted to the common unit using `inUnit(...)` (which will be a no-op in case the term already has a unit which is equivalent to the common unit).

6.3 Examples

Examples in this section will be illustrated using the following conventions for the display of a robustness analysis graph:

- Constraint vertices are placed to the left, and variable vertices to the right.
- An origin of ambiguity is colored in blue, and an origin of fragility is colored in orange. The remaining constraint vertices have dark gray color.
- A variable which is robust by definition is colored in light gray, while other variables have dark gray color.
- All edges incident to robust variables are colored in light gray (they cannot be part of robustness conflict paths). Edges reachable from an origin of ambiguity are drawn with an arrow in the direction of increasing distance to the origin of ambiguity. The remaining edges are drawn with dashed line and without arrow.
- In a constraint, empty is presented as ∅, and undefined as ?.

With this representation, existence of a robustness conflict path corresponds to an orange vertex with an incoming arrow. A robustness conflict path can then be reconstructed by following the directed edges in reverse direction until a blue vertex is reached. Note that the robustness analysis graphs presented below correspond to one of many possible orders of processing the unit constraints, but that while other implementations are likely to find other origins of ambiguity, the existence of a robustness conflict path does not depend on the order in which the constraints are processed.

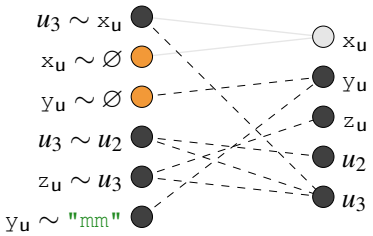


Figure 1. Robustness analysis graph without conflict path.

Consider the following simple model with explicit automatic unit conversion:

```
Real x(unit = "m") = 0.1;
Real y(min = 1.0'mm') = 10.0;
Real z = x + autoUnit(y);
```

The robustness analysis graph in Figure 1 contains origins of fragility, but no origins of ambiguity since all unit constraints are fulfilled without implicit unit conversion. Without any origins of ambiguity there cannot be any robustness conflict.

Now consider the same model again, but without `autoUnit`:

```
Real x(unit = "m") = 0.1;
Real y(min = 1.0'mm') = 10.0;
Real z = x + y;
```

The robustness analysis graph in Figure 2 contains a robustness conflict path leading to the fragile constraint for the binding of `y`.

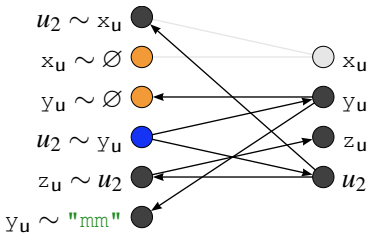


Figure 2. Robustness analysis graph with conflict path.

By adding a `unit`-attribute for `y`, the robustness conflict path is cut by the removal of the node y_u. Then, add the literal `1.0` the sum:

```
Real x(unit = "m") = 0.1;
Real y(unit = "mm") = 10.0;
Real z = x + 1.0 + y;
```

The robustness analysis graph in Figure 3 contains a robustness conflict path also this time.

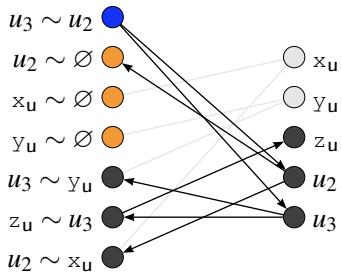


Figure 3. Robustness analysis graph with conflict path.

Now change units so that unit conversion is not needed:

```
Real x(unit = "m") = 0.1;
Real y(unit = "m") = 0.01;
Real z = x + 1.0 + y;
```

This is robustness analysis applied to existing Modelica models in a nutshell; there are plenty of origins of fragility, but no origins of ambiguity. As expected, the robustness analysis graph in Figure 4 has no robustness conflict path.

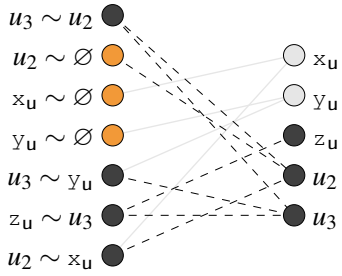


Figure 4. Robustness analysis graph without conflict path.

Robustness analysis doesn't require that all unit variables have been determined:

```
Real x = 2.0;          // Undetermined unit
Real y = 3.0;          // Undetermined unit
Real z = 1'm' + 100'cm';
Real u = x + y * z;
```

As expected, the robustness analysis graph in Figure 5 has a robustness conflict path.

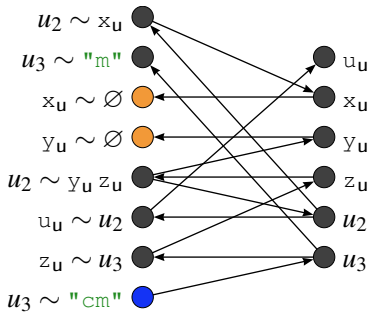


Figure 5. Robustness analysis graph with conflict path.

A consequence of the desire to avoid the "wildcard effect" of empty literals in current Modelica is that multiplication does not distribute over addition when empty literals are present:

```
Real x(unit = "m") = 2.0;
Real y(unit = "m") = 3.0;
Real u = x * (1.0 + y);    // 6.0'm2'
Real v = x * 1.0 + x * y; // Inconsistent
```

Since there is a constraint which cannot be resolved using implicit unit conversion, robustness analysis is neither needed nor applicable.

The following equation is considered consistent in units according to current Modelica, and is therefore also considered valid with the extended semantics:

```
  Real x(unit = "m");
equation
  x = 2.0;
```

Note how the literal `2.0` gets its unit from the other side of the equation. The robustness analysis graph in Figure 6 has no robustness conflict path.



Figure 6. Robustness analysis graph without conflict path.

Now try to adding the same value of quantity to both sides of the equation:

```
  Real x(unit = "m");
equation
  x + 300.0'mm' = 2.0 + 300.0'mm';
```

Looking at the right-hand side alone, the addition of 300 millimeters has changed it from 2000 millimeters to 302 millimeters, a quite unpleasant surprise. It is reassuring to see that the robustness analysis graph in Figure 7 has a robustness conflict path.

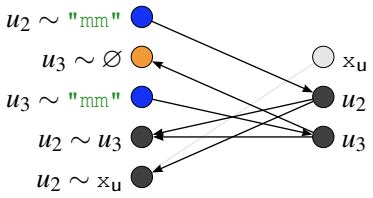


Figure 7. Robustness analysis graph with conflict path.

Consider subtracting the same literal without unit from both sides of an equation:

```
  Real x1(unit = "m");
  Real x2(unit = "m");
equation
  x1 + 1 = 1.0'mm';
  x2 = 1.0'mm' - 1;
```

The solutions to the two equations are not the same, that is, the solutions for `x1` and `x2` are poorly defined values of quantity. Again, it is reassuring to see that the robustness analysis graph in Figure 8 has two independent robustness conflict paths.

DOI 10.3384/ecp21827

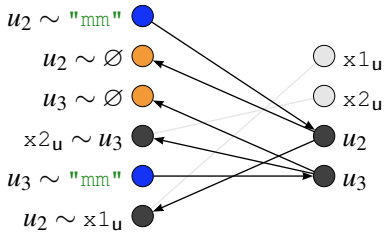


Figure 8. Robustness analysis graph with two independent conflict paths.

Ambiguity can spread through multiplicative constraints:

```
  Real x = 1'cm' + 1'm';
  Real y;
  Real z = y + 1.0;
equation
  1.0'J' = x * y;
```

The robustness analysis graph – omitted due to space constraints – contains a robustness conflict path via y_u.

Variation of the previous example:

```
  Real x = 1'cm' + 1'm';
  Real y;
equation
  1.0'J' = x * (y + 1.0);
```

The robustness analysis graph in Figure 9 has a robustness conflict path. That giving `y` a `unit`-attribute does not help is a sign of the conservativeness of the proposed semantics.

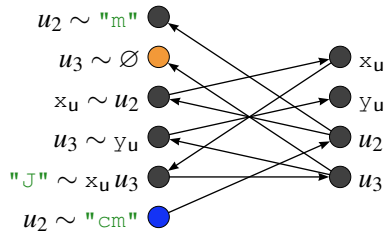


Figure 9. Robustness analysis graph with conflict path.

The trivial resolution of the conflict is to attach the intended unit to the literal:

```
  1.0'J' = x * (y + 1.0'kN');
```

When units are undefined, the possibility of implicit unit conversion must not invalidate models that are valid according to current conventions:

```
function f
  input Real u;
  output Real y = u;
end f;

Real x(unit = "m") = 2.0;
Real y(unit = "mm") = 300.0;
Real u = y + f(x);
Real v = u + 1.0;
```

Intuitively, however, `f(x)` should have had unit `"m"`, and there should be a need for unit conversion resulting in an origin of ambiguity incident to u_u, which in turn would reach the origin of fragility from `u + 1.0`. What actually happens is that `f(x)` has undefined unit, meaning that there is no origin of ambiguity in the model and that the model is accepted also according to the extended semantics.

Now take a model which is inconsistent according to current Modelica:

```
Real u = 1.0'cm' + f(2.0'm');
Real v = 1.0'cm' + u + 1.0'mm';
```

The function call's undefined unit will lead to an origin of fragility, and the need for implicit unit conversion means that there will be at least one origin of ambiguity in the robustness analysis graph. Without any robust variables in the graph, it is easy to see that a robustness conflict path exists. This is in agreement with what would happen if `f(2.0'm')` was inlined, so that there would be no undefined units in the model.

While robustness analysis of equation-based models is essential for illustrating the principles, the final example in this section is a component-based model where implicit conversion is needed in a connection set. Unlike some of the nontrivial examples above, it is representative of the simple way in which implicit conversion matters most to applications. That is, `unit`-attributes are present near the place of implicit conversion on both sides, shielding any origins of fragility in other parts of the model from being reachable from the origin of ambiguity:

```
  Sources.Constant a(k(unit = "m/h") = 1);
  Math.Gain b(u(unit = "mm/s"), k = 1);
equation
  connect(a.y, b.u);
```

7 Applications

In the following, let `Convert` be the block version of the `inUnit` operator, that is, a block for explicit conversion to a given unit. Similarly, let `Value`, `NumericValue` and `AutoConvert` be the block versions of the `withUnit`, `withoutUnit` and `autoUnit` operators.

7.1 Eliminating conversion blocks

In current Modelica, a unit conversion block can be created by careful parameterization of a gain block:

```
final Gain from_MW_to_W(
  u(unit = "MW"),
  y(unit = "W"),
  k(unit = "W/MW", displayUnit = "1") = 1e6
);
```

While the use of `displayUnit = "1"` helps entering and verifying that the value of `k` is correct for the current choice of source and target units (compare subsection 1.3), the setup is still error-prone and users who are not familiar with gain blocks used to perform unit conversion could find the use confusing.

With `inUnit` or `autoUnit`, defining a reusable unit conversion block is straight-forward, but unnecessary

when implicit unit conversion is allowed. Instead of using a unit conversion block between a source with unit `"MW"` and a sink with unit `"W"`, the source and sink may be connected directly. While this may appear strange with the unit equivalence mindset, it is the natural way of connecting the blocks with the value of quantity mindset.

Sometimes, the shift of mindset alone may be enough to eliminate a conversion block. Here, if the sink would have been an integrator for computing total amount of work based on input power, the value of quantity mindset makes it less important whether the unit of the total work is `"W.s"` or `"MW.s"`. For instance, a user might prefer to use a display unit such as `"J"` or `"kW.h"`, and then the underlying unit will not matter for the presentation.

7.2 Working with table data

When table data is expressed in different units than what is used in the rest of a model, there are several interesting options based on the proposed extended semantics. As before, the starting point is to ensure that the table component for accessing the data has connectors with correctly specified `unit`-attributes.

By putting a `Convert` block after a table output, it is possible to read table data in an explicit choice of unit. In a model made without declared units, a `NumericValue` block can be used instead of the `Convert` block to access the table data in the undeclared units assumed by the model. In a model with declared units, a `Convert` block will generally be redundant in view of implicit unit conversion, but if there is a preference of unit conversion to happen as close as possible to the table block, an `AutoConvert` block can be used to pinpoint the transition between the table's units and the units used in the rest of the model.

Similar options exist handling the need for unit conversion of table inputs.

7.3 FMI import

Although the creation of an FMU wrapper model for each imported FMU opens up the possibility of adding unit conversion blocks in the wrapper model, the recommended approach is to use FMU's units in the interface of the wrapper model. With this approach, working with FMUs is even easier than working with table data, since the FMI standard makes it a responsibility of the FMU import mechanism to set up correct units in the wrapper model interface.

7.4 Connecting with external processes

The typical way of connecting a Modelica model to an external process is to use a library providing blocks for reading and writing data over some process communication protocol. Again, the important part of the setup is to ensure that the block instances for reading and writing data have correctly configured `unit`-attributes. When supported by the communication protocol, the library should verify that the units configured for the read blocks agree with incoming data, and that the units configured for write blocks are transmitted with the written data.

7.5 Combination with unitful literals

Consider setting a length parameter to 40 feet. In current Modelica, it is not easy for a human looking at the source code to see that the parameter has a simple value in the `displayUnit`:

```
parameter Real p
  (unit="m", displayUnit="ft") = 12.192;
```

By combining constructs proposed in this work, the intent of the parameter setting can be made clear also to humans looking at the source code by writing `40'ft'` directly in the declaration equation. Further, if the preferred unit for plotting the result is not feet, the `displayUnit` can be changed independently of the unit used to set the parameter value.

It is observed that adding dedicated graphical user interface support for this way of expressing declaration equations would provide a clean solution to the problem of how many digits to present when display unit conversion is lossy due to finite precision, such as in the common case of having degrees as display unit for angles in radians.

7.6 Comparing a ratio to a threshold

Consider some variants of (attempting to) compare a ratio to a threshold:

```
Real x = 1.0'm';
Real y = 1.0'cm';
Boolean b1 = x / y > 50;
Boolean b2 = inUnit(x / y, "1") > 50;
Boolean b3 = x / y > 50'1';
```

In the first case (`b1`) one gets the quite surprising result that one meter is not more than 50 times longer than one centimeter. The second variant is the most verbose, but gives the desired result and does not rely on implicit conversion. The third variant is the most compact thanks to relying on implicit conversion.

8 Future work

It might come as a surprise that the proposed extended semantics considers the following a robustness conflict:

```
Real x1 = 1.0'm' + (1.0'mm' + 1.0);
```

That is, not even parentheses can be used to remove ambiguity. One idea for future work is therefore to make the robustness analysis aware of the parentheses.

Depending on how the methods proposed here for locally taking control over the implicit unit conversions are received, there could be reasons to explore other options as well. This would include the `FixedUnits` approach taken by Unitful.jl, but the Modelica setting also offers the possibility of specifying an annotation on the level of a class or equation for disallowing implicit unit conversion.

The relation between evaluable variability and unit conversion needs clarification. Here, more work is required

DOI
10.3384/ecp21827

to analyze the tradeoff between complexity of interleaving parameter evaluation and unit inference, and the limitations that would come with more restrictive approaches with less implementation effort.

There are currently several tool-specific mechanisms for defining additional unit symbols beyond the ones required by the Modelica specification, but a standardization effort is needed to make use of these units portable across tools. Alternatively or as a complement, the need for customization could easily be mitigated by adding many more well established units to the Modelica specification, and having well established units in the specification would also reduce risk of conflicting custom definitions or multiple symbols being introduced for the same unit.

When understanding of unit checking for units without offset in Modelica has matured, the time would be ripe for a discussion about units with offset, that is, the topic of quantity space checking. With quantity space checking, it becomes possible to detect an inconsistency when computing the difference between two absolute temperatures in degree Celsius and kelvin. When combined with implicit unit conversion, the inconsistency is automatically resolved by converting one operand to the unit of the other before carrying out the subtraction.

In the long run, it would be interesting to pursue a development in another direction, namely to support moving the Modelica ecosystem away from the legacy of allowing expressions with empty unit to be interpreted as values of quantity based on units taken from the context. This would align unit handling in Modelica with unit handling in other popular languages, and could eventually completely eliminate the need for the robustness analysis presented in this work. By only keeping the possibility of interpreting empty unit expressions as values of quantity with unit `"1"`, the threshold comparison in subsection 7.6 could finally be correctly expressed simply as `x / y > 50`.

9 Summary and conclusions

It has been demonstrated how implicit unit conversion can be safely allowed in Modelica. The crucial difference to other languages and software for unit aware computation is the Modelica legacy of allowing expressions without unit to be interpreted as values of quantity based on units taken from the context. When implicit unit conversion is allowed, this creates a risk of interpreting such expressions based on units which are not uniquely determined. A simple and conservative condition for rejecting programs in risk of ambiguously determined values of quantity has been presented. The condition will allow implicit conversion in typical situations where it is useful, as well as never reject any Modelica program which is valid according to current unit semantics. In addition to implicit unit conversion, a set of basic operators for working with units and values of quantity has been proposed. These operators add value on their own, and can also be used to locally take control over the implicit unit conversions.

The importance of understanding a model in terms of related values of quantity rather than merely seeing units attached to variables as decorative ornaments has been stressed. Without this understanding, implicit unit conversion in Modelica would make as little sense as it would in any of the other languages and softwares that allow implicit unit conversion today.

The usefulness of the proposed unit semantics has been demonstrated in several real world examples of Modelica use, and the safety of the approach has been illustrated with numerous minimal test cases.

Acknowledgements

This work has been supported by Wolfram MathCore.

References

Aronsson, Peter and David Broman (2009-09). “Extendable Physical Unit Checking with Understandable Error Reporting”. In: *Proceedings of the 7th International Modelica Conference*, pp. 890–897. DOI: 10.3384/ecp09430027.

Hindley, J. Roger (1969). “The Principal Type-Scheme of an Object in Combinatory Logic”. In: *Transactions of the American Mathematical Society* 146, pp. 29–60. DOI: 10.2307/1995158.

Kennedy, Andrew J. (1997-01). “Relational Parametricity and Units of Measure”. In: *Proceedings of the 24th Annual ACM Symposium on Principles of Programming Languages*. ACM.

Lambert, Quentin and Henrik Tidefelt (2024). *Applying Hindley-Milner to unit-checking*. Tech. rep. Wolfram MathCore. URL: https://github.com/modelica/ModelicaSpecification/pull/3491.

Maple – Tutorials Part 10: Units (2025). URL: https://www.maplesoft.com/support/help/Maple/view.aspx?path=MaplePortal/Tutorial10 (visited on 2025-04-24).

Math.js – Units (2025). URL: https://mathjs.org/docs/datatypes/units.html (visited on 2025-07-27).

Mattsson, Sven Erik and Hilding Elmqvist (2008-03). “Unit Checking and Quantity Conservation”. In: *Proceedings of the 6th International Modelica Conference*. Vol. 1, pp. 13–20. URL: https://modelica.org/events/conference2008/proceedings/volume_1.pdf.

Milner, Robin (1978). “A Theory of Type Polymorphism in Programming”. In: *Journal of Computer and System Sciences* 17.3, pp. 348–374. DOI: 10.1016/0022-0000(78)90014-4.

Pint – Tutorial (2025). URL: https://pint.readthedocs.io/en/stable/getting/tutorial.html (visited on 2025-04-24).

Simulink – Converting Units (2025). URL: https://se.mathworks.com/help/simulink/ug/convert-units.html (visited on 2025-04-24).

Unitful.jl – Conversion/promotion (2025). URL: https://painterqubits.github.io/Unitful.jl/stable/conversion/ (visited on 2025-04-24).

Wolfram Language – Quantity (2025). URL: https://reference.wolfram.com/language/ref/Quantity.html (visited on 2025-04-24).

DOI
10.3384/ecp21827

Model Disambiguation Technology in MWORKS.Sysplorer

Zhipeng Chen[1] Zhichao Huang[1] Chong Zhou[1] Yinqi Chen[1] Qi Liu[1] Fanli Zhou[1] Liping Chen[2]

[1]Suzhou Tongyuan Software & Control Technology Co., Ltd., Suzhou, China , {chenzhipeng, huangzc, zhouc, chenyq, liuq, zhoufl}@tongyuan.cc
[2]Huazhong University of Science and Technology, Wuhan, China , chenlp@hust.edu.cn

Abstract

Modelica models exhibit excellent cross-platform compatibility (they can be compiled and simulated on any platform supporting Modelica). However, experiments have revealed that simulation results of the same Modelica model may vary across different platforms (under identical simulation algorithm configurations). The root cause of such discrepancies lies in model translation uncertainty introduced by improper modeling practices, such as insufficient initial constraints or ambiguous state variables selection. Different Modelica tools may have different translation strategies. Therefore, model disambiguation should be performed to ensure consistent simulation results. It can be addressed by three parties: through language improvements, such as the proposal of relevant annotations, by vendor tools; and by the modelers through manual intervention. This paper presents a model disambiguation technology in MWORKS.Sysplorer that enables modelers to automatically correct model text based on translation information, eliminating uncertainties and ensuring model portability across Modelica platforms.

Keywords: Model Disambiguation, Cross-platform, Modelica, MWORKS.Sysplorer

1 Introduction

Modelica's declarative and object-oriented modeling approach lowers the barrier to modeling complex systems. Specialized model library vendors, research teams, and open-source communities provide foundational model components, which users then leverage to build their own research objects. These foundational components often come with default configurations, such as default start values and state variables selection (Mattsson and Söderlind 1993). Modelers must adjust these settings according to their specific application scenarios, including designing necessary initial constraint equations to ensure their Modelica model satisfies well-defined mathematical constraints.

Although Modelica platforms can still translate and simulate models based on default configurations, such models may yield different simulation results across different platforms. These discrepancies can arise from platforms selecting different state variables or applying non-equivalent initial constraint conditions. Additionally, most modern Modelica platforms support nonlinear equation tearing techniques, but implementation differences may lead to the selection of distinct nonlinear iteration variables. When nonlinear equations have multiple solutions, variations in the chosen iteration variables and their initial estimates can further affect simulation outcomes.

To ensure Modelica model portability across platforms, potential translation ambiguities must be resolved during the modeling phase. However, most modelers have limited knowledge of Modelica's translation and solving theory, making it difficult to construct ideal models. Moreover, for complex models (e.g., those with hundreds or thousands of state variables), relying on modelers to eliminate uncertainties is neither reliable nor cost-effective.

The paper is organized as follows. Section 2 uses a series of extremely simple examples to systematically dissect the sources of model translation uncertainty. Section 3 presents MWORKS.Sysplorer's model disambiguation technology and proposes standardized Translation annotation semantics for Modelica specifications. Section 4 empirically demonstrates the practical effects of model disambiguation through experimental validation.

2 Model Translation Uncertainty

Model translation uncertainty primarily stems from three aspects: lack of initial equations, selection of state variables, and nonlinear tearing. We will provide a detailed explanation using a series of extremely simple examples (though lacking engineering significance) in separate subsections, demonstrating how it leads to inconsistent simulation results.

2.1 Lack of Initial Equations

An initial system in Modelica remains valid even with missing but consistent equations (Pantelides 1988), as tools automatically supplement them. Modelica specifies that start values should be used to supplement initial values. If no start value is available, default values are applied (0 for Real and Integer types, false for Boolean types). When selecting m equations from n candidates with $0 < m < n$, uncertainty arises. Different tools have significant discretion in how they perform this completion, and the underlying mechanisms they employ are generally not the same. Vendors should report supplemental information to alert modelers.

2.2 Start Value Recommended Priority

As "Start Value Recommended Priority"(see §8.6.2 (Modelica Association 2023)) states, the start values from models closer to the top level are considered more reliable. However, some vendor tools may not follow this recommendation for various reasons. Subsequent tool upgrades adopted this specification-recommended strategy for selecting start values, resulting in incompatibility with older versions.

Listing 1. An example of model translation uncertainty caused by lack of initial equations with multiple start value

```
model Case1
  model M
    Real x(start = 1);
  equation
    der(x) = 1;
  end M;
  M m;
  Real y(start = 2) = m.x;
end Case1;
```

In Listing 1, tools adopting Modelica's recommended start value selection strategy will set the initial value of x to 2. However, older versions may initialize x with 1 instead. Considering that not all modeling tools support this feature uniformly, MWORKS.Sysplorer's will elevate start values to initial conditions by adding "fixed=true" attributes when performing model disambiguation.

2.2.1 Lack of initial equations for state variables

Listing 2. An example of model translation uncertainty caused by lack of initial equations for state variables

```
model Case2_1
  Real x;
  Real y;
equation
  x + y = 1;
  der(x) - der(y) = 0.5;
end Case2_1;
```

Noted that not all differential variables will be identified as state variables. In Listing 2, both x and y are differential variables, and they are subject to the algebraic constraint $x+y=1$. As a result, the system has only one degree of freedom, meaning only one of them can be selected as a state variable. The system does not provide additional information to determine which variable x or y is the better choice as the state variable.

Listing 3. A mathematically equivalent example to Listing 2

```
model Case2_2
  Real x;
  Real y;
equation
  y + x = 1;
  der(x) - der(y) = 0.5;
end Case2_2;
```

Changing $x+y=1$ to $y+x=1$ in Listing 2 yields Listing 3, which are mathematically equivalent, but get different result in Dymola 2024x trivial version. This is because Dymola selects y as the state variable in Listing 2, while choosing x as the state variable in Listing 3. As a result, the initial value $(x,y)=(1,0)$ in Listing 2, while $(x,y)=(0,1)$ in Listing 3. The root cause is that the original model lacks an initial condition, and the selection of state variables affects how default conditions are supplemented.

Listing 4. An alias example to Listing 2

```
model Case2_3
  Real x;
  Real u;
equation
  x + u = 1;
  der(x) - der(u) = 0.5;
end Case2_3;
```

Similarly, by renaming y to u in Listing 2, we obtain Listing 4. When simulating with MWORKS.Sysplorer, y is selected as the state variable in Listing 2, while x becomes the state variable in Listing 4. The default values of the state variables are used as the initial conditions, resulting in completely different outcomes.

To eliminate this uncertainty, modelers can: (1) Assign a start value to either x or y in Listing 2 to explicitly determine the state variables selection. However, if both are given start values, they return to an equally weighted condition, providing no additional information to prioritize one over the other. (2) Use the StateSelect attribute (see §4.8.7.1 (Modelica Association 2023)), such as StateSelect.prefer, to explicitly specify the preferred state variable. But if both variables are marked with this attribute, the ambiguity persists.

Since the fixed mechanism alone cannot reliably resolve such conflicts (particularly when multiple variables carry additional constraints), Sysplorer adopts the following strategy when performing model disambiguation: (1) Automatically applies "fixed=true" to elevate start values to initial conditions. (2) Uses "StateSelect.always" to enforce unambiguous state variables selection. Additional note: In cases of dynamic state variables selection, MWORKS.Sysplorer will only automatically process a subset of differential variables with "fixed=true" when performing model disambiguation.

2.2.2 Lack of initial equations for discrete variables

Listing 5. An example of model translation uncertainty caused by lack of initial equations for discrete variables

```
model Case3
  Integer x;
  Integer y;
equation
  when sample(0, 0.1) then
    x = pre(x) + 1;
    y = pre(y) + x;
  end when;
initial equation
  y = x + 1;
end Case3;
```


DOI
10.3384/ecp21839

In Listing 5, both x and y are discrete variables with only one initial equation provided, requiring an additional initial condition to be supplemented. The key issue lies in whether to set $x = 0$ or $y = 0$ as the supplemental condition. Although mathematically equivalent, when the initial equation $y = x + 1$ is reformulated as $x = y - 1$, both MWORKS.Sysplorer and Dymola2024x exhibit changes in their automatically supplemented conditions. This ultimately leads to inconsistent simulation results.

To eliminate this uncertainty, modelers can similarly assign start values to either x or y in Listing 5, which explicitly uses the start value as the initial condition. However, when both variables have start values specified, the system returns to an equally weighted condition where no additional information exists to determine which start value should take precedence as the initial condition. The fixed attribute mechanism alone cannot reliably resolve this conflict when both variables carry additional constraints. To address this, MWORKS.Sysplorer automatically applies "fixed=true" to elevate the start values to initial conditions.

2.3 Necessity of Start Values for Nonlinear System

The uncertainty in model solving arises from the existence of multiple solutions in nonlinear systems. Currently, the specifications do not provide clear recommendations for the selection of nonlinear iteration variables (i.e., nonlinear tearing), which may lead to inconsistencies in nonlinear tearing across different versions. Consequently, the initial iteration values may vary, resulting in different solutions or even cases where one version succeeds in finding a solution while another fails.

Listing 6. An example of necessity of start values for nonlinear system

```
model Case4_1
  Real x;
  Real y = x - 2;
equation
  (x - 0.78) * (x - 1.57) * (x - 3.14) = 0;
end Case4_1;
```

In Listing 6, the direct simulation yields a result of 0.78. However, when x's start value is set near 1.5, the simulation result becomes 1.57 and when x's start value is set near 3, the result changes to 3.14.

This demonstrates that the tool cannot autonomously determine which solution the modeler intends to obtain, and consequently performs iterations based on the provided start value. This observation underscores the critical necessity of specifying start values as initial estimates for nonlinear systems.

2.4 Nonlinear Tearing

Listing 7. An example of nonlinear iteration variables selection

```
model Case4_2
  Real x;
  Real y;
equation
  x - y = 2;
  (x - 0.78) * (x - 1.57) * (y - 1.14) = 0;
end Case4_2;
```

Listing 6 and Listing 7 are mathematically equivalent, but yield different simulation results in MWORKS.Sysplorer: x=0.78 in Listing 6 versus x=1.57 in Listing 7. This discrepancy occurs because x is explicitly determined as the nonlinear iteration variables in Listing 6 while both x and y qualify as potential nonlinear iteration variables, and MWORKS.Sysplorer defaults to selecting y In Listing 7.

Providing a start value for x forces its selection as the iteration variables. However, when both x and y have start values, the system reverts to the original ambiguous state. Nonlinear iteration variables has no *StateSelect* attributes like *state variables*. So MWORKS.Sysplorer innovatively employs annotation markup to explicitly specify which variable should serve as the nonlinear iteration variables. This annotation-based approach effectively eliminates the ambiguity, and the entire process can be automatically handled by the model disambiguation technology in MWORKS.Sysplorer.

2.5 More Words about Nonlinear Tearing

Note that, it is possible that some variables appear in both the nonlinear system and the initial nonlinear system. For instance, variable y in Listing 8 falls into this category

Listing 8. An example of nonlinear iteration variables selection

```
model Case5
  Real x;
  Real y;
  Real z;
equation
  der(x) = sin(time);
  x^2 + y^2 = z^2;
  z = x * sin(y) + y * cos(x);
initial equation
  x + y + z = 1;
end Case5;
```

2.6 More Words about State Variables Selection

Even when the initial equations have sufficient initial conditions, the selection of state variables can still introduce uncertainty. In Listing 9, if y is chosen as the state variable, then x and z form a nonlinear system. However, if x is selected as the state variable, no nonlinearity exists. This discrepancy leads to non-negligible differences in simulation results between platforms that choose x as the state variable and those that select y.

Listing 9. An example of model translation uncertainty caused by state variables selection

```
model Case6
  Real x(start = 0);
  Real y(start = 0.5);
```

```
  Real z;
equation
  der(x) + der(y) = 1;
  z = sin(x)^2 + 1;
  z = x + 2 * y + time;
initial equation
  x = 0;
end Case6;
```

In Listing 9, if the "stateSelect=StateSelect.always" attribute is added to x and y separately, and simulations are performed for each case, the results are as shown in Figure 1.

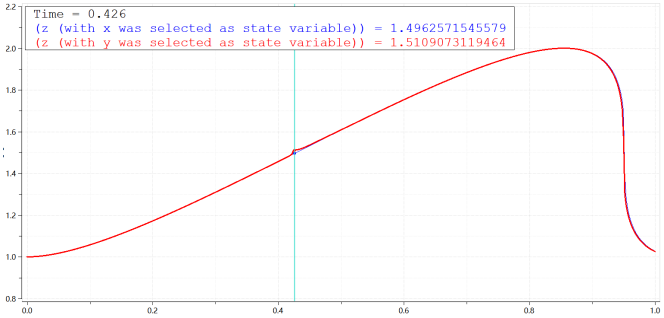


Figure 1. An example of inconsistency caused by State Variable Selection

2.7 A graphic example about State Variables Selection

We provide a simplified circuit model (which may lack physical significance) to demonstrate that even when using a drag-and-drop component assembly method, inconsistent selection of state variables can lead to divergent simulation results.

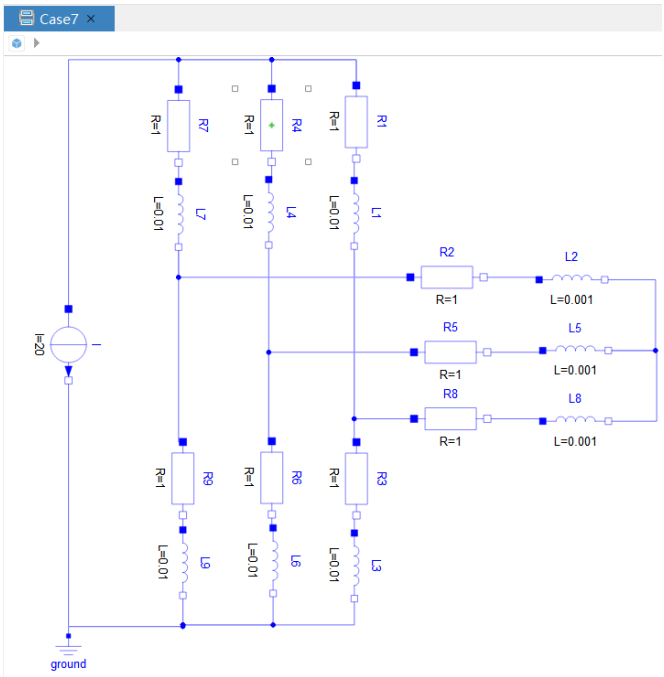


Figure 2. A graphic example about State Variables Selection

The circuit model Figure 2 contains 9 inductors with algebraic constraints, but only 4 of them can have their currents (which must be linearly independent) selected as state variables. MWORKS.Sysplorer selected L6.i, L7.i, L8.i, and L9.i as state variables, while OpenModelica chose L1.i, L4.i, L5.i, and L8.i. This difference in selection leads to inconsistent initial conditions between the two platforms, for instance, L1.i has an initial value of 20A in MWORKS.Sysplorer, but 0A in OpenModelica.

3 Model Disambiguation Technology

MWORKS.Sysplorer utilizes the results of the first simulation to perform the following actions during model disambiguation in the second translation:

1. Add the "fixed=true" attribute to variables that were supplemented with initial values in the report.

2. Add the "stateSelect=StateSelect.always" attribute to state variables listed in the report.

3. Record vendor-customized annotations for nonlinear iteration variables and initial nonlinear iteration variables, and assign them corresponding start values.

4. If the above modifications involve protected variables, vendor annotation could be used, otherwise, they would conflict with Chapter 4.1 of (Modelica Association 2023).

The following model Listing 10 represents the disambiguated version of all models mentioned in the previous section. Specifically, Case2_1_protected, Case4_2_protected, and Case5_protected correspond to modified versions of Case2_1, Case4_2, and Case5 respectively, where all model variables have been set as protected variables.

Listing 10. An example of disambiguated model

```
model DisambiguationModel
  Case1 m1(m(x(stateSelect=StateSelect.
      always,fixed=true)));
  Case2_1 m2(y(stateSelect=StateSelect.
      always,fixed=true));
  Case2_1_protected m2_p;
  Case3 m3(x(fixed=true));
  Case4_1 m4(x(start=0.78));
  Case4_2_protected m4_p;
  Case5 m5(x(stateSelect=StateSelect.always
      ),y(start=0.349110989362788),z(start
      =0.4190691332521359));
  Case5_protected m5_p;
  Case6 m6(y(stateSelect=StateSelect.always
      ),x(start=0));
  Case7 m7(L7(i(stateSelect=StateSelect.
      always,fixed=true)),L9(i(stateSelect=
      StateSelect.always,fixed=true)),L6(i(
      stateSelect=StateSelect.always,fixed=
      true)),L8(i(stateSelect=StateSelect.
      always,fixed=true)));
```


DOI
10.3384/ecp21839

```
  annotation(__MWORKS(Translation(m2_p(y(
      stateSelect=StateSelect.always,fixed=
      true)),m5_p(x(stateSelect=StateSelect
      .always),y(start=0.349110989362788),z
      (start=0.4190691332521359)),m4_p(y(
      start=-0.43000000000000005)),
      INonLinearVariables={m5.y, m5.z, m5_p
      .y, m5_p.z},MNonLinearVariables={m4.x
      , m4_p.y, m6.x, m5.y, m5_p.y})));
end DisambiguationModel;
```

The following are keynotes when doing model disambiguation:

1. If the original model is read-only, you can create a new model that references it, then perform disambiguation in the new model.

2. If the model fails to solve in the first place, disambiguating its state becomes rather meaningless, and the modelers need to modify the model themselves.

3. After Performing model disambiguation, modelers can still manually fine-tune the model by editing the text. Regardless of whether the original model is modifiable, it is recommended to create a new model for disambiguation to clearly track all modifications.

4. After Performing model disambiguation, providing explicit start values will accelerate the solving of the initial value system.

5. MWORKS.Sysplorer uses vendor-specific annotations to explicitly define the selection of nonlinear variables.

6. A variable may appear in both the nonlinear system and the initial nonlinear system, so MNonLinearVariables and INonLinearVariables are used to control them separately. MNonLinearVariables means nonlinear iteration variables during model simulation, while INonLinearVariables means nonlinear iteration variables during Initialization.

7. Unlike Modelon (Kari 2022), which considers both iteration variables and residual equations to resolve ambiguity caused by nonlinear tearing, we focus solely on iteration variables for the following reasons: configuring annotation is difficult when equations aren't at the simulation model's top level; different tools process iteration and residual equations maybe inconsistently; and our practical experience demonstrates that restricting iteration variables alone sufficiently eliminates model ambiguity.

8. If the modeler modifies the model after model disambiguation, causing variables in the annotations to become invalid, a warning message must be issued. There are two typical handling methods: directly ignoring the annotations or striving to satisfy the annotations. The tool may offer an option to let the modeler choose the preferred approach.

9. While other platforms don't recognize MWORKS.Sysplorer's vendor annotations, the model disambiguation provides deterministic initial values and relatively accurate compatible initial guesses for nonlinear iterations - making it highly probable to maintain result consistency across other platforms.

3.1 Annotation of Translation

Since the minimal tearing problem is NP-hard (Karp 2009) (Elmqvist and Otter 1994), most simulation platforms prioritize translation efficiency by employing greedy or heuristic algorithms to approximate minimal tearing. Even if vendors develop improved heuristic strategies, concerns over version compatibility often discourage modifications, hindering tool advancement.

Generally, better heuristics should facilitate nonlinear system solving. A typical scenario where models solvable in the original tool fail in new tools occurs because when initial modeling in the original tool, users would supplement appropriate start values as nonlinear iteration initial guesses if missing values caused solving failures, after switching to new tools with different tearing strategies, some models may lack required start values due to altered variable selection, necessitating manual re-supplementation.

Therefore, we propose that the Modelica specification incorporate Translation Annotations to govern nonlinear/-linear tearing and initial-value system tearing, thereby eliminating strategy-induced inconsistencies for modelers. For tool vendors, the implementation priority should be:

1. First satisfy the tearing requirements specified in the annotations.

2. Then prioritize variables based on start value hierarchy (top-level variables take precedence).

3. Finally pursue minimal tearing under equal conditions.

This strategy ensures tearing consistency across all compliant platforms when users provide comprehensive annotation specifications. The benefits are threefold:

1. Better interoperability between different vendors' tools.

2. Enhanced control for modelers over system solving.

3. Optimal utilization of each tool's translation and solving capabilities.

Ultimately, this advancement moves Modelica closer to the ideal of "Write once, run on any Modelica tool" - significantly improving model portability and reducing maintenance overhead. However, even with identical model

descriptions and unified nonlinear/linear tearing through translation annotations, achieving simulation results that differ only by rounding errors across platforms under identical settings remains a significant challenge. Key persistent discrepancies include:

1. Inconsistent precision tolerance interpretations across platforms
2. Divergent implementations of nonlinear/linear solvers
3. Algorithmic variations in implicit integration methods
4. Discrepancies in variable-step size adjustment strategies
5. Interpolation method variations
6. Event handling policy differences
7. Integrator restart strategy implementations

While writing this paper, we observed that both OpenModelica (Fritzson et al. 2022) and Modelon (Kari 2022) employ vendor-specific annotations for tearing, which exhibit subtle differences not only between themselves but also when compared with MWORKS.Sysplorer. This phenomenon demonstrates that such requirements are relatively common. Therefore, we hope the Modelica Association could provide an official annotation standard to eliminate inconsistencies across different implementations.

4 Experimental Results

When simulating the model "LoadTestExpRecovery" from the model library OpenIPSL-3.0.1 (Vanfretti et al. 2016), it was found that the results from MWORKS.Sysplorer and OpenModelica were completely inconsistent, for example, the variable "pwLine2.P12". After investigation, the inconsistency was traced to differences in nonlinear tearing. Since it was hard to control OpenModelica's nonlinear selection at the modeling level, we instead used custom annotation in MWORKS.Sysplorer to control nonlinear tearing, as shown in the Listing 11.

Listing 11. An example of control nonlinear tearing using Translation annotation

```
model LoadTestExpRecovery_Disambiguated
  extends LoadTestExpRecovery;
  annotation (experiment(Interval=0.03,
      StartTime=0,StopTime=15), __MWORKS(
      Translation(INonLinearVariables={
      order3_Inputs_Outputs1.e1q,
      order3_Inputs_Outputs1.vd,
      order3_Inputs_Outputs1.vq,pwLine2.n.
      ii,pwLine2.n.ir,pwLine2.n.vi,pwLine2.
      n.vr,pwLine2.p.ii,pwLine2.p.ir,
      pwLine3.n.ii,pwLine3.n.ir,pwLine3.n.
      vi,pwLine3.n.vr,pwLine3.p.ii,pwLine3.
      p.ir,pwLine4.n.ii,pwLine4.n.ir,
      pwLine4.p.ii,pwLine4.p.ir},
      MNonLinearVariables={
      order3_Inputs_Outputs1.vd,
      order3_Inputs_Outputs1.vq,pwLine2.n.
      ii,pwLine2.n.ir,pwLine2.n.vi,pwLine2.
      n.vr,pwLine2.p.ii,pwLine2.p.ir,
      pwLine3.n.ii,pwLine3.n.ir,pwLine3.n.
      vi,pwLine3.n.vr,pwLine3.p.ii,pwLine3.
      p.ir,pwLine4.n.ii,pwLine4.n.ir,
      pwLine4.p.ii,pwLine4.p.ir})));
end LoadTestExpRecovery_Disambiguated;
```

The final simulation results of Listing 11 are consistent with OpenModelica, see Figure 3 below.

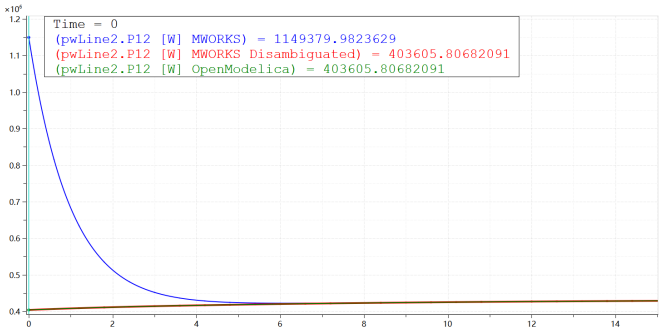


Figure 3. An example of inconsistency caused by State Variable Selection

References

Elmqvist, Hilding and Martin Otter (1994). "Methods for tearing systems of equations in object-oriented modeling". In: *Proceedings ESM*. Vol. 94, pp. 1–3.

Fritzson, Peter et al. (2022). "The OpenModelica integrated environment for modeling, simulation, and model-based development". In: Mic.

Kari, Oskar (2022). "Improving Tearing in a Modelica Compiler". In: *LU-CS-EX*.

Karp, Richard M (2009). "Reducibility among combinatorial problems". In: *50 Years of Integer Programming 1958-2008: from the Early Years to the State-of-the-Art*. Springer, pp. 219–241.

Mattsson, Sven Erik and Gustaf Söderlind (1993). "Index reduction in differential-algebraic equations using dummy derivatives". In: *SIAM Journal on Scientific Computing* 14.3, pp. 677–692.

Modelica Association (2023-03). *Modelica – A Unified Object-Oriented Language for Systems Modeling. Language Specification Version 3.6*. Tech. rep. Linköping: Modelica Association. URL: https://specification.modelica.org/maint/3.6/MLS.html.

Pantelides, Constantinos C. (1988). "The Consistent Initialization of Differential-Algebraic Systems". In: *SIAM Journal on Scientific and Statistical Computing* 9.2, pp. 213–231. DOI: 10.1137/0909014.

Vanfretti, Luigi et al. (2016). "iTesla Power Systems Library (iPSL): A Modelica library for phasor time-domain simulations". In: *SoftwareX* 5, pp. 84–88.

Data Reconciliation for Industrial Experiments

Baptiste Mazurié[1] Audrey Jardin[1] Pascal Borel[1] Frans Davelaar[1] Didier Boldo[1] Luis Corona Mesa-Moles[1]

[1]R&D, Electricité de France, France, firstname.lastname@edf.fr

Abstract

This paper presents the use of data reconciliation to improve the characterization of the operating state of experimental installations in an industrial context. It focuses on the development of a data reconciliation approach using the OpenModelica prototype to study: the detection of the defects in an experimental hydraulic test loop; and to validate measurement data of an HVAC testing facility. Data reconciliation is shown to be effective for the most pronounced defects intentionally introduced in the system. Regarding the characterization of HVAC measurements, data reconciliation identified two initially unnoticed invalid experiments.

Keywords: data validation, data reconciliation, default detection, uncertainties, measurements, Modelica, model reuse, industrial experiments

1 Introduction

Control of measurement equipment and associated acquisition chains is a key element for monitoring and diagnosis of industrial installations, in particular in the domain of electricity production, where constraints related to safety and resilience for securing the supply of energy networks require a sound and accurate knowledge of the true operating state of the system. To that aim, several experimental installations exist in premises of EDF Lab to improve the characterization of uncertainties associated with the examination of the main circuits of interests by testing new methods, using either direct measurements (via new sensor technologies) or indirect measurements (via the complementary use of knowledge models).

This article falls into this second category showing how a physical simulation model combined with statistical assessments can be used to:

- detect deviations from nominal operation and alert operators of the occurrence of possible faults
- validate measurement campaigns.

In practice, the work focuses on the use of a technique known as Data Reconciliation (DR). The aim is to obtain an estimate of the most likely state of the system by correcting the measured values on the actual installation so that the reconciled values are consistent with the equations governing the behaviour of the system.

While data reconciliation exists for several decades and its application to energy systems is ruled by a German standard (VDI 2048 2017), it is finding a renewed interest in the nuclear field for Measurement Uncertainty Recapture (MUR) and potential associated power uprates (EPRI 2020).

Dedicated tools (VALI, PROCESSPLUS, Thermal System Monitoring Enterprise, BILCO ...) exist to perform data reconciliation, but they require the physical models to be specifically developed for that purpose. They use their own proprietary language and are sometimes restricted to a specific domain (e.g. material processing industry). This makes data reconciliation costly and difficult to use. A possible answer to that problem is to apply data reconciliation on existing Modelica general purpose models, developed and validated for system operation.

This is why DR has been implemented in OpenModelica enabling the reuse of simulation models by automatically extracting the equations that should be considered for binding the measured variables together.

The principle and use on a simple example has been documented in (Bouskela et al. 2021). This paper presents first Data Reconciliation algorithm (Section 2) and then shows its applicability on two industrial experiments: one to control liquid flow rate measurements (Section 3) and a second to measure air flows (Section 4).

2 Data Reconciliation Theory

Industrial processes are usually monitored with a number of measurement equipments which can be more or less isolated on different sections of the circuit and of different technologies leading to differences in terms of representativeness, accuracy and reliability. The problem is to determine whether all these measurements, which are necessarily subject to error, are physically and statistically consistent with each other and whether they are a reliable mirror of the installation.

Data reconciliation is an optimization process that allows numerical correction of measurement values so that they satisfy the physical equations of the system (the model). If the corrections are too significant, the measurements are invalidated: in that case reconciliation alerts to a possible instrumentation defect (e.g., poor calibration or a faulty sensor), a process defect (e.g., an unmodeled leak) or a poor knowledge of sources of measurements uncertainties (e.g., lack of correlation effects). If the measure-

ments are validated, the reconciliation exploits the redundancy of information to reduce uncertainties and obtain an estimate as close as possible to the true state of the system.

2.1 Mathematical Foundations

Data reconciliation aims at improving the accuracy of measurements by reducing the effect of random errors in the data. The main difference between data reconciliation and others techniques for propagating uncertainties such as Quadratic Combination or Monte Carlo simulation (Baudin et al. 2017; EPRI 2020) is that data reconciliation uses a model to express the physical constraints on the variables of interest. The measured values of these variables are then adjusted such that the estimates satisfy the constraints: the variables are thus reconciled.

Let $\{X_0^t,\ldots,X_d^t\}$ be a vector of d physical quantities constrained by a set C of r equations or auxiliary conditions derived from physical laws. These quantities must satisfy Equation 1 to ensure consistency.

$$\begin{aligned} C &: \mathbb{R}^d \to \mathbb{R}^r \\ 0 &= C(X^t) \end{aligned} \tag{1}$$

Measurements $\hat{X}$ of X^t inevitably have uncertainties, which are commonly modeled by a random variable X with a multivariate Gaussian distribution characterized by its mean vector μ and its variance-covariance matrix Σ.

$$\pi(X) = \mathcal{N}(\mu,\Sigma) \tag{2}$$

The contradiction vector U of a given state $\{X_0,\ldots,X_d\}$ is given by:

$$U = C(X) \tag{3}$$

The objective of data reconciliation is to define a conditional probabilistic distribution of measurements, called the reconciled distribution, such that Equation 1 is satisfied by the random variable $X|C(X)=0$. This reconciled distribution is a multivariate Gaussian characterized by a mean vector μ' and a variance-covariance matrix Σ'.

$$\pi\left(X|C(X)=0\right) = \mathcal{N}(\mu',\Sigma') \tag{4}$$

(Bouskela et al. 2021) present a detailed methodology for solving this optimization problem using Lagrange multipliers as stated in (VDI 2048 2017). The optimization problem consists in minimizing the following objective function in order to determine the set of reconciled values μ':

$$J(\mu') = (\mu'-\mu).\Sigma.(\mu'-\mu)^\top \tag{5}$$

The variance-covariance matrix of reconcilied values derive from the general formula of uncertainty propagation:

$$\Sigma' = \partial_\mu \mu'.\Sigma.\partial_\mu {\mu'}^\top \tag{6}$$

Relevancy of the reconciled distribution can be assessed using local and global test : χ^2, reflecting the relevancy of the estimated reconciled distribution based on prior measurements uncertainty knowledge. The global test involves verifying the following statistical hypothesis, where r denotes the number of degrees of freedom and p represents the significance level associated with the χ^2 distribution:

$$J(\mu') \leq \chi^2_{r,p} \tag{7}$$

The improvement vector v is defined as:

$$v+\mu = \mu' \tag{8}$$

Local tests, measurement by measurement, can help to identify measurements with a too large improvement pushing the most probable reconciled value to be out of its confidence range defined by the prior measurement knowledge as presented in Equation 9.

$$|\mu'_i - \mu_i| < \Sigma^v_{ii}\lambda_p \tag{9}$$

where μ_i is the i^{th} measurement, μ'_i is the i^{th} reconciled measurement which is also the mean of the i^{th} marginal of the reconciled distribution and λ_p is the confidence interval width, usually taken as 1.96 corresponding to 95% of the possible values. Σ^v is the variance-covariance matrix of the improvement v defined as:

$$\Sigma^v = \partial_\mu v \Sigma \partial_\mu v^\top \tag{10}$$

2.2 Measurement Uncertainty: GUM

The measurement of physical quantity inevitably has an uncertainty which has to be estimated. Let X_i^t the true value of physical quantity to measure, $\hat{X}_i$ the corresponding measurement with an uncertainty ε_i

$$X_i = \hat{X}_i + \varepsilon_i \tag{11}$$

ε_i is a random variable with a null mathematical expectation which can by decomposed in elementary uncertainty sources as follows:

$$\varepsilon_i = \varepsilon_i^0 + \ldots + \varepsilon_i^n \tag{12}$$

One can follow the guide (ISO/IEC GUIDE 98-3 2008) to establish a model of the random variable ε_i by proposing a multivariate statistical distribution $\pi(\varepsilon_i)$. Typically, there are five sources of uncertainty: sensor calibration, sensor environment impact, sensor acquisition chain, measurement methodology, and repeatability uncertainties. The first four are epistemic uncertainties, which must be determined a priori, usually based on sensor provider documentation or calibration reports. The last source, if it exists, is a stochastic uncertainty that must be evaluated by empirical samples collected during measurement.

When the uncertainty sources are assumed to be Gaussian, it is necessary to specify the variance-covariance matrix of the uncertainty sources, denoted as Σ. Estimating the correlation is a challenging aspect that can significantly impact uncertainty propagation. In practice, without empirical samples, the correlation is often assumed to be either null or total.

DOI
10.3384/ecp21845

Complex measurements of quantities Y can be based on several measurements of elementary quantities that are combined with a function g.

$$Y = g(X) \tag{13}$$

Uncertainty propagation with methods such as first-order Taylor series expansion or Monte Carlo simulation aims to estimate the statistical distribution of Y. In the context of data reconciliation, estimating the multivariate distribution of Y, which is assumed to be a multivariate Gaussian distribution, is a crucial step that must be performed meticulously. This includes accurately estimating the correlation between the uncertainties of different measurements. The construction of this multivariate distribution can be achieved using OpenTURNS (Baudin et al. 2017), as illustrated in Figure 1.

2.3 Data Reconciliation in OpenModelica

The main added-value of performing DR in OpenModelica lies in the reuse of models initially developed for other purposes than data validation or diagnosis.

In particular the equational nature of the Modelica language is used to automatically extract from a complete simulation model (i.e. a square system of equations with the same numbers of equations than unknowns) a minimal subset of auxiliary equations that will bind the measured variables together (i.e. here a non-square system of equations since the redundancy of measurements is exploited for the reconciliation).

The extraction algorithm has already been fully described in (Bouskela et al. 2021) and is now available in the standard release of OpenModelica.

For the sake of clarity, Figure 1 summarizes the main steps required to perform a complete data reconciliation process in OpenModelica. For more details and illustration on the simple example of a splitter, the user is invited to read the documentation accessible online (*OpenModelica Documentation: Data Reconciliation* 2021).

On Figure 1 light blue refers to steps implemented in OpenModelica while dark blue highlights steps performed in a third-party tool dedicated to uncertainty quantification (in this case OpenTURNS (Baudin et al. 2017) but other tools can be chosen). Basically, the user :

1. states the reconciliation problem by specifying which variables should be reconciled and which equations should be voluntary not be considered as an auxiliary condition (typically the boundary conditions). This is done by decorating the Modelica model with dedicated annotations;

2. defines the set of measurements and quantifies their corresponding uncertainties (potentially with the help of a dedicated tool). The numerical information to be reconciled can be provided in the form of a csv file;

3. launches the DR functionality which will extract the auxiliary conditions and computes the reconciled data. Based on the results, a report is generated in the form of an html file with reconciled outputs and statistical tests.

4. analyses the validity of reconciled data with respect to the initial assumptions.

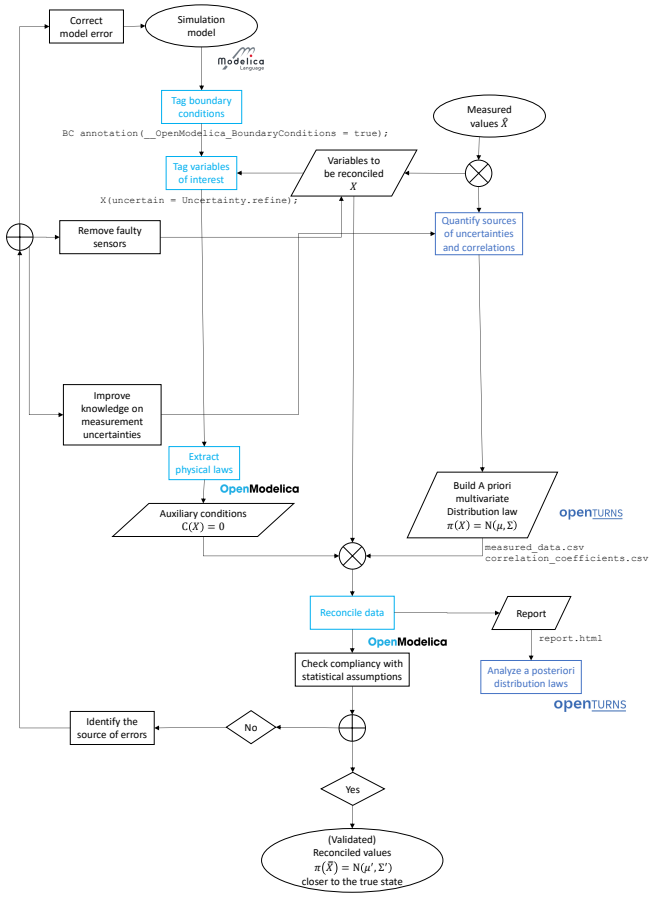


Figure 1. Data reconciliation workflow in OpenModelica

3 Flowrate Measurement with EVEREST Facility

3.1 Description of the test loop

3.1.1 Global Description

EVEREST (Thibert et al. 2019) is a closed loop facility with a liquid flow rate regulation (Table 1). The fluid is clean demineralized water. EVEREST consists of three parts: a flow metering reference section, an operation section and a modular flow metering test section (Figure 2). Table 1 shows the EVEREST design specifications.

The test bench provides a steady water stream with Reynolds numbers as low as 5.10^4 up to 2.10^6. Flow circulation is provided by a variable centrifugal pump (315

Prime features	Details
Calibration test type	Closed regulated loop
Flow meter reference	Master meters method
Test section pipe size	From DN40 up to DN350
Pipe compositions	Stainless steel
Fluid	Demineralised water
Pressure range	From 1 up to 10 bar
Fluid temperature range	From 20°C up to 40°C
Environment Temperature	From 10°C up to 35°C
Flow rate range	From 5 $m^3.h^{-1}$ up to 1200 $m^3.h^{-1}$
Flow metering uncertainty	0.2% (volume and mass flow rate)

Table 1. Overview of the design specification of EVEREST

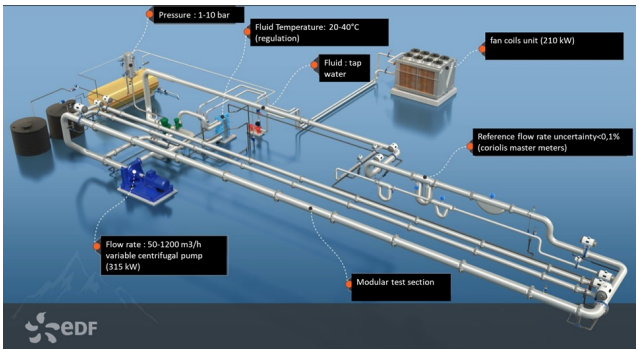


Figure 2. EVEREST test facility layout design

kW). Depending on the geometrical configuration of the test section, the flow can range from 25 $m^3.h^{-1}$ up to 1100 $m^3.h^{-1}$. Thanks to regulation systems, a steady flow rate is maintained in the reference and the test sections. The heat produced by the pump is absorbed by the pumped liquid flowing in the pipes. Due to the EVEREST loop configuration, this phenomenon consequently increases the fluid temperature. To compensate this phenomenon, the water temperature is regulated thanks to a plate heat exchanger (215 kW). The cooling system including the fan coils unit is located at the outside of the experimental hall. The fluid temperature can thus be maintained at a specific value according to the desired test conditions, generally between 20°C and 40°C. The fluid pressure is regulated at a set-point between 1 bar up to 10 bar thanks to a pressuriser.

The EVEREST loop is designed to minimise its environmental impact by managing efficiently its water consumption. Three water storage tanks are used to fill and empty the circuits between two test campaigns requiring different pipe configurations.

3.1.2 Reference Flow Rate

The reference flow rate is calculated from four different master meters based on the Coriolis technology. Two meters are used for low flow rate calibration tests, from $50m3.h^{-1}$ up to $350m3.h^{-1}$. They are installed on a DN100 stainless steel pipe. The other two devices operate under high flow rate calibration tests conditions, from $150m3.h^{-1}$ up to $1200m3.h^{-1}$. They are installed on a DN250 stainless steel pipe. This configuration aims at decreasing uncertainty over the whole loop flow rate rangeability, so each meter is used in its nominal operative flow rate.

For each reference line, one Coriolis meter is used as the reference flow rate while the other is used as a "drift detection" meter. This choice allows monitoring and detecting any reference sensor drift between two consecutive calibrations.

The EVEREST reference flow rate can be calculated either as a volume or as a mass flow rate. The EVEREST test loop has been ISO17025 accredited with a volume (resp. mass) flow rate uncertainty of 0.2% from 25 $m^3.h^{-1}$ (resp. $t.h^{-1}$) up to 1100 $m^3.h^{-1}$ (resp. $t.h^{-1}$).

3.1.3 Modular Test Section

The flow metering test section is located at the end of EVEREST (see Figure 3) and resides completely inside the building. Devices to be tested are mounted on stainless steel pipes, they can be invasive as well as non-invasive according to the sensor technology. This section constitutes an available space with a length of 25 m, a width of 5 m and with a high ceiling of 5 m.



Figure 3. EVEREST test facility test section

This test section is designed to perform calibration tests not only under ideal thermo-hydraulic conditions (fully developed and swirl-free velocity profile thanks to a long straight pipes at the upstream of the tested meter) but also calibration tests under non-ideal thermo-hydraulic conditions due to the presence of various fittings located upstream or downstream of the tested meter.

Summerizing, the test section of the loop allows the recreation of numerous process pipe geometries or to install a reduced scale mock-up of typical industrial components for metrological purposes. This flexibility is a key feature of the loop as it allows EDF R&D to investigate the impact on the accuracy of flowmeters installed in real industrial conditions.

3.2 Detection of Defects

As described in §3.1, the EVEREST test loop is a representative and reliable experimental test facility. Therefore, it is an interesting use case for Data Reconciliation

DOI 10.3384/ecp21845

in order to improve the reliability of the facility's state estimation. Indeed, the test loop being modular, deliberately machined defects can be introduced into the system to impact its behaviour. The defects are introduced upstream the mass flow rate measurement and are shown in Figure 4. They are representative of what could be found in thermal hydraulic installations, such as weld bead defects (trapezoidal or rectangular) or changes in diameter at the junction of two pipes (referred to as step defects). Those defects have more or less impact on the flow and are sorted by order of impact in Figure 4, with the one having no defect showing no impact, and the change in diameter having the greatest impact on the flow. Thus, the goal is to determine whether data reconciliation will allow the detection of these defects.

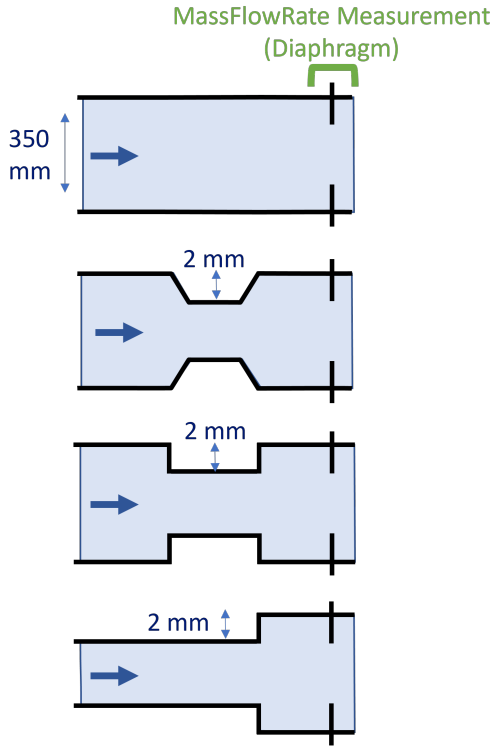


Figure 4. Diagram of the different defects introduced in the EVEREST test loop: no defect, trapezoidal weld bead defect, rectangular weld bead defect, change of diameter (also refered as step defect) sorted by order of impact from the one having the less impact (no defect added), to the one having the more impact on the flow (step defect).

A data reconciliation approach using the OpenModelica prototype is thus developed to study the detection of these defects thanks to an experimental test campaign on the loop. This workflow is inspired from (EPRI 2020). First a Modelica model of the facility is developed using manufacturer's data of the different equipment and calibrated by inversion of `n` parameters based on `n` historical measurements from the facility. Initial tests of data reconciliation are run to make sure that the data reconciliation problem is well-posed. Those first steps ensure a good numerical representativity of the installation and that the data reconciliation problem is well-posed. Once the initial phase completed, test campaigns are conducted, both with and without defects, to allow comparison with a control test. The data reconciliation algorithm is run and challenges the prior uncertainties with the posterior results. If the results are significantly different (in the sense of a χ^2 test as described in §2.1), it can be concluded that an inconsistency has been detected: either the model is incorrect or the measurements are false.

A readily available ThermoSysPro (El Hefni and Bouskela 2019) model of EVEREST test loop shown Figure 5 was adopted and calibrated. Test campaigns are then

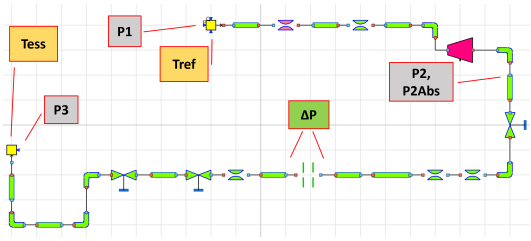


Figure 5. ThermoSysPro model of the EVEREST test loop with the different measurements used (Pressure, Temperature and MassFlowRate).

conducted, starting with a characterisation test, followed by one defect at a time. For each test campaign (each defect), different thermal hydraulic conditions are tested (variation of mass flow rates) both with and without the defect. A test campaign for one defect thus consists of 105 measurement sets (with and without the defect). Effectiveness of defect detection with data reconciliation can be evaluated in many ways. For instance, the global and local tests for data reconciliation can be used in order to identify if a test is detected as faulty by comparing it to a threshold λ as described by Equation 9. If so, the confusion matrix for such a case can be written as shown in Table 2. For industrial use case defect detection, it is desirable to have as few *False Positive (FP)* as possible.

Confusion matrix	**Test with a defect**	**Test with no defect**
Detected with a defect	True Positive (TP)	False Positive (FP) *False alarm*
Detected with no defect	False Negative (FN) *No detection*	True Negative (TN)

Table 2. Confusion matrix for defect detection

With numerous measurements to evaluate the effectiveness of defect detection using data reconciliation, new indicators can be computed. Sensitivity and Specificity are defined. Sensitivity (or true positive rate) is the probability of detecting a defect for a test containing a defect (Equation 14). Specificity (or true negative rate) is the

probability of not detecting any defect for a test that does not contain any (Equation 15). In a perfect scenario, Sensitivity and Specificity should be equal to 1 as all defects would be detected and no false alarms would appear.

$$Sensitivity = \frac{TP}{TP+FN} \tag{14}$$

$$Specificity = \frac{TN}{TN+FP} \tag{15}$$

For each test campaign, the data reconciliation algorithm is applied to each measurement set. As the result of the Sensitivity and Specificity for each set depends on the threshold λ, it is plotted with respect to λ. Starting with the defect having the greatest impact on the flow, Figure 6 shows the Sensitivity and Specificity against the threshold for the step defect. It can be seen that with the default value of $\lambda = 1.96$, this defect can be detected perfectly as Sensitivity and Specificity are equal to 1.

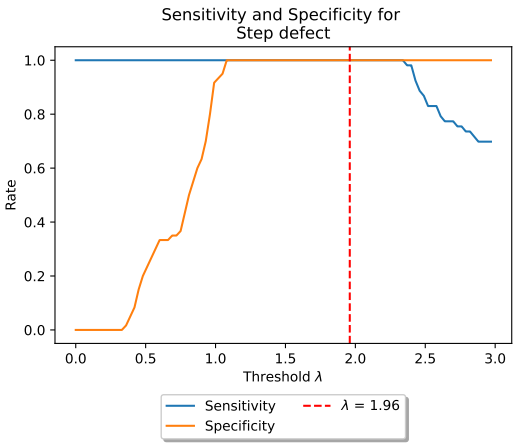


Figure 6. Sensitivity and Specificity against threshold λ for the Step defect. The defect can be detected perfectly with $\lambda_{default} = 1.96$ as Sensitivity and Specificity are equal to 1.

However, the default value of λ does not always lead to a perfect diagnosis. Indeed as shown on Figure 7, the rectangular weld bead defect can be detected perfectly, but not with $\lambda_{default} = 1.96$. Indeed with this default value, the Sensitivity is equal to 0.35. λ should be in the interval $[1.05; 1.35]$ so that the diagnosis is perfect. This difference has an impact on the diagnosis as more *False Negatives* would have been found.

Finally, for the trapezoidal weld bead defect, which has the smallest impact on the flow, it is seen on Figure 8 that the defect can not be perfectly detected as no range of threshold λ gives a Sensitivity and Specificity equal to 1. However the threshold could be reduced so that the Specificity is still high, but that some *True Positives* have been found.

Thus, data reconciliation can detect defects that were intentionally introduced in the system. On the most pronounced defects as the step defect, the impact is such that global and local tests fail with the default threshold value

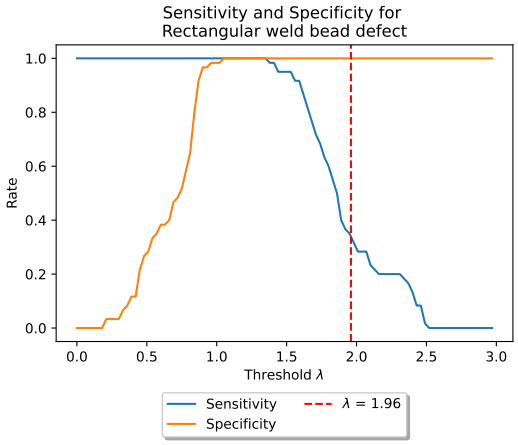


Figure 7. Sensitivity and Specificity against threshold λ for the Rectangular Weld Bead defect. The defect can be detected perfectly, but not with $\lambda_{default} = 1.96$, but with $\lambda \in [1.05; 1.35]$.

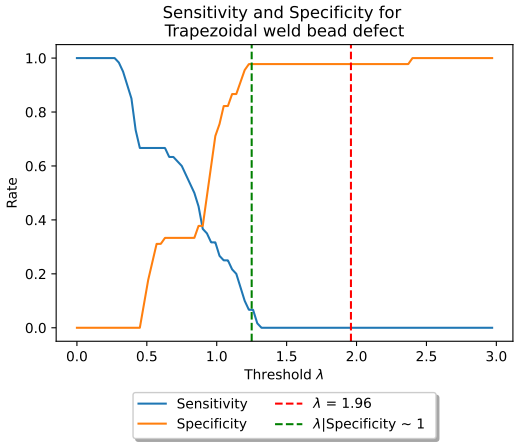


Figure 8. Sensitivity and Specificity against threshold λ for the Trapezoidal Weld Bead defect. The defect can not be detected perfectly.

of $\lambda_{default} = 1.96$. However with less serious defects, $\lambda_{default}$ is not always appropriate. The optimal threshold $\lambda_{optimal}$, depends on the defect that is studied and should be fixed to a value minimising the number of false alarms.

4 ZEPHYR Facility for Testing HVAC Cooling Coil with Latent Exchanges

4.1 In search for good measurement data

The modular test facility ZEPHYR provides air at a precise flow and temperature. On hot days, the air is cooled by a heat exchanger and under humid conditions, condensation can occur. To predict the performance of this heat exchanger at very high temperature and humidity, a validated simulation model is needed. While several theoretical models are available, all of them need "good" input data to calibrate their parameters. Therefore, an experimental test campaign on the cooling coil of the main Air Handling Unit of ZEPHYR and illustrated on Figure 9,

DOI
10.3384/ecp21845

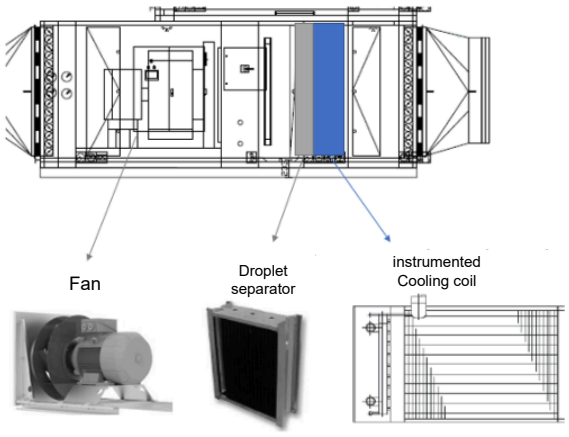


Figure 9. Instrumented cooling coil of Air Handling Unit of ZEPHYR laboratory

was developped to create a set of validated measurement data.

4.2 Experimental campaign

Sensors were installed on the (cool) water side and the (hot) air side of the heat exchanger. On the water side, input and output temperatures as well as the flow are measured. Air measurements at the inlet include temperature and humidity, and at the outlet, temperature and flow. For latent heat, the quantity of condensate is measured. Additionally, the uncertainty of each calibrated sensor has been determined, primarily from data sheets and measured data points. The uncertainty distribution of each sensor can be considered Gaussian. The outlet temperature at one point can be considered representative, because the uniformity of the outlet air temperature was confirmed using an optical fiber installation with approximately 60 measurement points evenly distributed across the entire cross-section.

Over fifteen steady-state experiments were conducted over two days, with varying flows and inlet temperatures (coolant and air). One of these experimental data points is shown in Table 3.

Label	Description	Value	σ	Units
Q_w	Coolant flow	7.91	0.09	kg/s
ΔT_w	Coolant temperature difference	1.62	0.03	°C
Q_a	Air flow	5.82	0.09	kg/s
T_{ai}	Inlet air temperature	29.0	0.30	°C
T_{ao}	Outlet air temperature	20.0	0.47	°C
RH_{ai}	Inlet air relative humidity	42.9	1.5	%
Qc	Condensate flow	1.62	0.03	kg/h

Table 3. Example of sensor data : values and uncertainty (k=1)

It can be observed that the uncertainty on the air side is higher than on the coolant side. One reason for this is that the inlet air was taken from outside the building, causing the temperature and humidity to change gradually throughout the day.

While the amount of condensate at the exit can be measured precisely, the phenomena related to condensation on the plates inside the heat exchanger are complex. Discrepancies, such as the transport time and the thickness of the condensate layer, may lead to invalid experimental data. The idea was to apply data reconciliation to eliminate these invalid experiments and simultaneously improve the accuracy of the measurement set. Since this set contained some level of redundancy, data reconciliation based on basic conservation laws was possible.

4.3 Driving equations

The purpose of this paragraph is to model the thermal exchanges inside an HVAC exchanger that cools humid air coming from outside, as illustrated in Figure 10. The air temperature inside the exchanger can drop below the dew point, caused by the relatively cold surface temperature of the coils, and produce condensation. The main energy balance equations are :

$$\begin{cases} P_w = Q_w(h_{wi} - h_{wo}) \\ P_a = Q_{ai}h_{ai} - Q_c h_c - Q_{ao}h_{ao} \\ 0 = P_w + P_a \\ Q_{ai} = Q_{ao} + Q_c \end{cases} \tag{16}$$

where h denotes the fluid enthalpy, Q the massflowrate and P the thermal power exchanged between the two fluids within the exchanger. The subscript $_a$ denotes air, $_w$ denotes water, $_c$ denotes condensate, $_i$ denotes inlet, and $_o$ denotes outlet. Psychrometric charts of humid air are

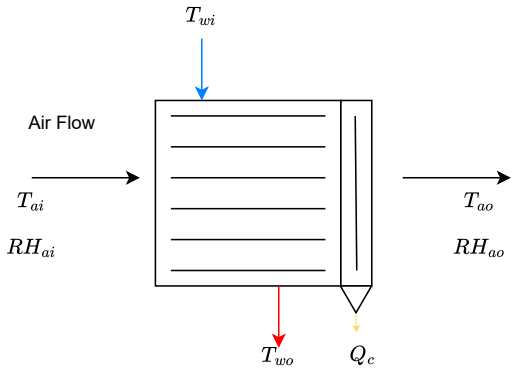


Figure 10. Scheme of the studied HVAC cooling coil with latent exchanges

typically used to visualize the thermodynamic processes occurring within the HVAC cooling coil, as shown in Figure 11.

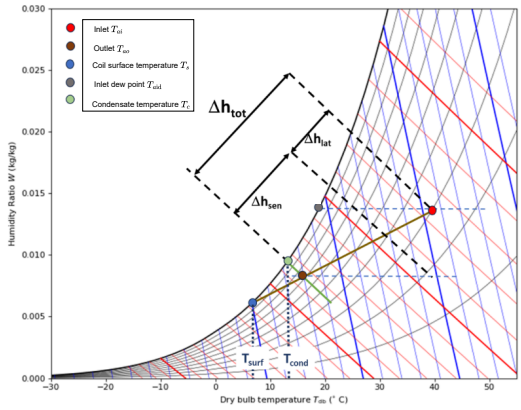


Figure 11. Typical thermodynamic process within an HVAC cooling coil

4.4 Data Reconciliation

Using the conservation equations Equation 16 and the experimental measurements listed in Table 3, it becomes evident that one measurement is redundant and can be deduced from the others. The enthalpy of the humid air and water, assuming the total pressure is equal to atmospheric pressure, is computed using T and RH for air, and T for water, utilizing the media package of the `Modelica Standard Library` as declared below:

Listing 1. HVAC Modelica media

```
replaceable package MediumAir = Modelica.
    Media.Air.MoistAir;
replaceable package MediumWater = Modelica.
    Media.Water.StandardWater;
```

Condensate rate is computed from the other measurements, solving a non linear system to respect mass and energy balances involving condensate rate. Measurements are declared as presented in Listing 2

Listing 2. Declare measurment for data reconciliation

```
/*Measurement*/
Modelica.SIunits.MassFlowRate Qa_mes(
    uncertain=Uncertainty.refine, start=1)
    "Air flowrate [kg/s]";
Modelica.SIunits.MassFlowRate Qe_mes(
    uncertain=Uncertainty.refine) "water
    flowrate [kg/s]";
Modelica.SIunits.Temperature Tai_mes(
    uncertain=Uncertainty.refine) "air
    inlet temperature [K]";
Modelica.SIunits.Temperature Tao_mes(
    uncertain=Uncertainty.refine) "air
    outlet temperature [K]";
Modelica.SIunits.Temperature Twi_mes(
    uncertain=Uncertainty.refine) "water
    inlet temperature [K]";
Modelica.SIunits.TemperatureDifference
    DeltaTw_mes(uncertain=Uncertainty.
    refine) "water temperature rise [K]";
Real RH_ai_mes(uncertain=Uncertainty.refine
    ) "air inlet relative humidity [-]";
Modelica.SIunits.MassFlowRate Q_c(uncertain
    =Uncertainty.refine) "condensate rate [
    kg/s]";
```

In this case, the number of obtained auxiliary conditions r is equal to one. The results of the data reconciliation for two typical experiments are presented on Figure 12 produced by interfacing `OpenModelica` data reconciliation with an OpenTURNS (Baudin et al. 2017) `https://openturns.github.io/` module as illustrated in Listing 3.

Listing 3. Perform data reconciliation with OpenTURNS and OpenModelica

```
covarianceMatrix = priorDistribution
    .getCovariance()
priorMeasurements =
    priorDistribution.getMean()
dataR = otDataR.DataReconciliationOM
    (modelName, VarNames,
    priorMeasurements,
    covarianceMatrix, casePath,
    simuMatPath)
dataR.reconcileData()
reconciledDistribution = dataR.
    getPosteriorDistribution()
```

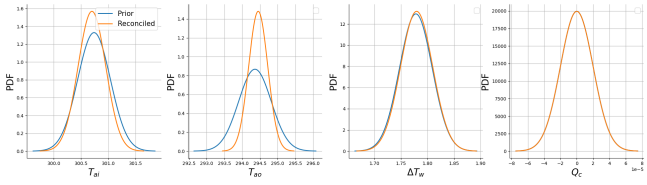
For the first experiment, the data is considered consistent, the result of the local being positive, which is qualitatively in agreement with the prior and posterior Gaussian distribution of the measurements. For the second experiment depicted on Figure 12, the prior and posterior distribution are clearly inconsistent. One can notice that the results of the data reconciliation process indicate that the air temperature measurement may be potentially incorrect, while the condensate rate measurement shows little variation between the prior and posterior distributions. This result is strongly driven by the specified prior distribution. The condensate rate measurement, relying on the mass of the condensate collected over a period, is specified as a measurement with a very low prior uncertainty while some uncertainty sources can have been misidentified due to the complex condensation phenomena occurring within the exchanger. It underlines that output of data reconciliation process has to be carefully analysed to identify correctly defectuous sensors in case of measurements set declared as non consistent.

Measurement with uncertainty before and after reconciliation are illustrated in Figure 13 and Figure 14. It can be observed that experiment number 8 leads to inconsistent measurement after data reconciliation, as the cooling coil outlet state is above the saturation curve. Therefore, this experiment has been discarded.

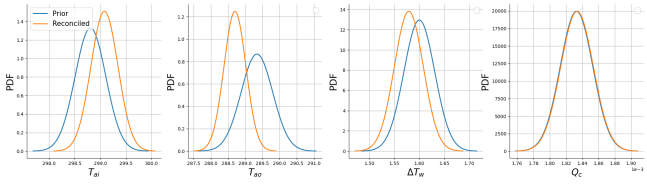
5 Conclusions and future work

The implementation of data reconciliation with OpenModelica has been successfully tested on two industrial experimental installations. For the EVEREST test loop

DOI
10.3384/ecp21845



(a) Positive local test and improve measurements



(b) False local test for some measurements

Figure 12. Example of reconciled data leading to: improve outlet air temperature measurement (a) - inconsistent measurements (b).

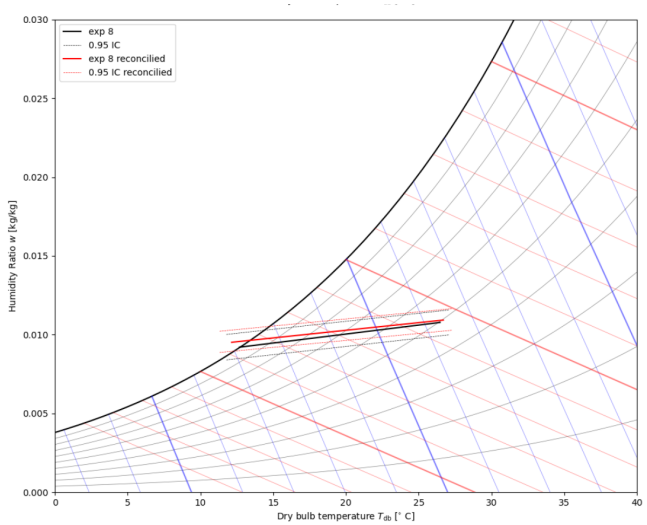


Figure 13. Measurements with uncertainty before and after data reconciliation for experiment number 8

aiming at controlling measurements of liquid flow rates, some defects were intentionally introduced and data reconciliation detected efficiently the most pronounced defects. The detection threshold λ should however be carefully chosen as it has a great impact on the detection results: for less serious defects such as the trapezoidal weld bead defect, the detection is not perfect and *False alarms/No detection signals* will be returned. Further work could be to integrate data reconciliation as a monitoring tool to study sensor drift of the facility in addition to other defect detection methods like comparing a measurement set to a validated reference characterisation test campaign to observe the relative difference through time.

Data reconciliation on the HVAC application for measuring air flows not only improved the accuracy and uncertainty of the measurements but it also identified two initially unnoticed invalid experiments. This high confidence reconciled data set, contains thus "good" measurements that can be used as inputs for parameter calibration of simulation models.

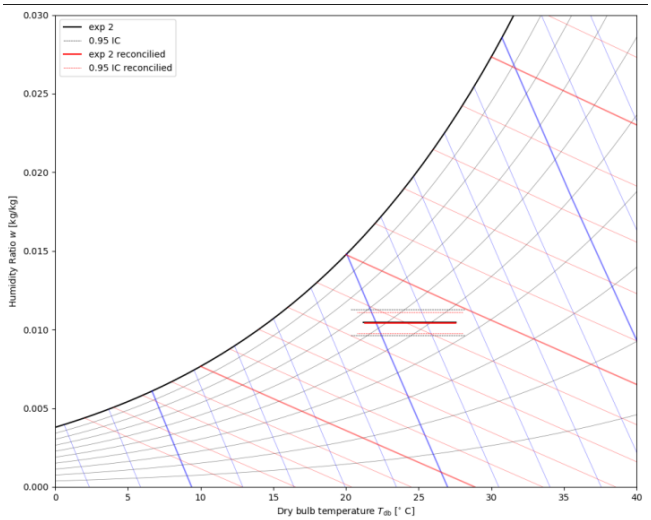


Figure 14. Measurements with uncertainty before and after data reconciliation for experiment number 2

Data reconciliation tools exist and have already proven their worth in the industrial sector. However, they are limited to a restricted modelling domain and often use a dedicated, proprietary modelling environment. In practice, this restricts the scope of reconciliation: apart from use cases where reconciliation is used as a real-time monitoring tool to gain operating margins, few use cases can afford this costly environment. Other applications can nevertheless be imagined: support for experimental installations as presented in this article, support for the design of systems to assist in the positioning and selection of sensors and provision of rigorous proofs to justify the control of uncertainties of sensitive quantities, etc. The integration of data reconciliation into OpenModelica opens up new possibilities: users no longer need to create a dedicated reconciliation model in a dedicated environment; they can derive it from pre-existing models, such as those used for classical simulation studies. The Modelica language is used there as a sole and standardized format for modelling the physical phenomena of the system hence bridging the gap between design tools and operation tools.

From the software standpoint, although the current implementation of data reconciliation in OpenModelica is still in the prototype stage, the article demonstrates its ability to handle industrially realistic models. The prototype documentation will be updated based on the application cases described here, particularly with regard to the methodology to be followed when using third-party tools to calculate uncertainties for pre- and post-processing of data reconciliation. Further improvements to the prototype are also expected to enable a better scaling for larger or more complex models. This will include improvements to the user interface for providing inputs to be reconciled - especially for linking model variables to uncertainty data- and for optimising the visualisation of reconciled outputs.

It will also include adjustments of the extraction algorithm to handle models embedding algebraic parts which cannot be easily inlined and symbolically processed (e.g. if-then-else clauses, function with external codes...).

Acknowledgements

The authors would like to thank Arunkumar Palanisamy, Lennart Ochel and Adrian Pop for their implementation of data reconciliation in OpenModelica. This functionality has been initially supported by DGE (France) and Vinnova (Sweden) in the scope of the ITEA3 OPENCPS project and then improved within the Open Source Modelica Consortium thanks to EDF funding.

References

Baudin, Michaël et al. (2017). "Openturns: An industrial software for uncertainty quantification in simulation". In: *Handbook of uncertainty quantification*. Springer, pp. 2001–2038.

Bouskela, Daniel et al. (2021). "New Method to Perform Data Reconciliation with OpenModelica and ThermoSysPro". In: *Modelica Conferences*, pp. 453–462. DOI: https://doi.org/10.3384/ecp21181453.

El Hefni, Baligh and Daniel Bouskela (2019). *Modeling and Simulation of Thermal Power Plants*. Springer.

EPRI (2020). *Use of Data Validation and Reconciliation Methods for Measurement Uncertainty Recapture*. Last accessed 24 March 2025. URL: https://www.nrc.gov/docs/ML2105/ML21053A030.pdf.

ISO/IEC GUIDE 98-3 (2008). *Uncertainty of measurement - Part 3: Guide to the expression of uncertainty in measurement (GUM:1995)*.

OpenModelica Documentation: Data Reconciliation (2021). URL: https://openmodelica.org/doc/OpenModelicaUsersGuide/latest/dataReconciliation.html (visited on 2025-04-29).

Thibert, Emmanuel et al. (2019). "EDF R&D new test bench for liquid industrial flow meters calibration". In: *FLOMEKO 2019. The 18th International Flow Measurement Conference*.

VDI 2048 (2017). *Blatt 1 "Control and quality improvement of process data and their uncertainties by means of correction calculation for operation and acceptance tests*.

Chemical 2.0

(Free open-source Modelica library)

Marek Mateják[1,2]

[1]Institute for Clinical and Experimental Medicine, Czech Republic

[2]First Faculty of Medicine, Charles University, Czech Republic

marek@matfyz.cz

Abstract

Free open-source Modelica library called Chemical 2.0 (https://github.com/MarekMatejak/Chemical) provides expressions between chemical substances and processes. These robust and unified definitions allow users to choose whether define processes or substances in their dynamic (electro-)chemical models. Propagation of substance definition and chemical solution through connected components simplify configuration. Chemical pathways can start even with unknown substances. Chemical kinetics was rewritten.

The possibilities and performance of chemical pathways modeling are increased using a new type of connectors based on inertial electro-chemical potential. Chemical processes can be directly connected without need to add unsignificant states. Parameterization of chemical reactions is also streamlined, e.g. using forward rate and dissociation coefficient.

Keywords: Chemical 2.0, inertial electro-chemical potential, Modelica library, physical chemistry, thermodynamics equilibria, electrochemical potential, electrochemical cell, internal energy, semipermeable membrane, chemical kinetics, chemical pathways

1 Introduction

Motivation to improve Chemical library from version 1.4 (Mateják et al. 2015) to version 2.0 comes from following Modelica libraries:

A) Media.IdealGases from Modelica Standard Library (Casella et al. 2006): Chemical substances and chemical processes can be described by the selected extensive physical quantities.
B) ThermofluidStream (Zimmer 2020): Inertial connectors can be generalized for chemical domain.

As selection of physical quantities for each substance and process definition we chose a similar approach used in Modelica.Media.IdealGases, which comes from (McBride, Zehe, and Sanford 2002). Having this definition as a record of constant coefficients, there is a possibility to calculate molar heat capacity, free molar formation energies, free molar entropy and many other thermodynamic properties in dependence on given substance activity and state of chemical solution (temperature, pressure, electric potential, etc.). Moreover, unlike the previous version, there is no need to have separate calculations for different states of matters (phases), because phase is also included in extensive physical quantities of the definition records. Therefore, every chemical equilibrium of each process can be expressed only using the definitions of included substances and processes (Peter Atkins, De Paula, and James Keeler 2018). And because we chose extensive quantities, they can be easily added or subtracted as in common chemical processes notations (e.g. mass of substrates = mass of products; charge of substrates = charge of products; enthalpy of substrates + enthalpy of reaction = enthalpy of products, entropy of substrates + entropy of reaction = entropy of products, etc.).

The connectors from ThermofluidStream library are revolutionary (Zimmer, Bender, and Pollok 2018). This library still allows user to define very complex thermodynamic behavior of predefined media in the same level of detail as in Modelica.Fluid and Modelica.Media does (Zimmer, Meißner, and Weber 2022). However, ThermofluidStream library permits connections between processes without unnecessary accumulations of media, simultaneously eliminating a number of non-linear systems in the model (Meißner and Zimmer 2022). By eliminating numerous middle-state components accumulating and mixing medium the model becomes more readable. And during the simulation run there is no longer need to solve implicit equations for many non-linear systems. All this results in significant improvement of the performance and readability, with minimal compromise in the model details (Zimmer, Meißner, and Weber 2021).

Using the inertial approach of ThermofluidStream in chemical domain promises the analogical improvements for Chemical 2.0. As a result, the processes can be connected directly together to define complex chemical

pathways such as in biochemistry or physiology. Additionally, allowing to define better chemical kinetics (Sauro 2009), complex processes and more detailed calculations of any part of the model.

The inertial approach is connected with following basic composition rules:

- Electro-chemical pathways shall start and end at a boundary component. Enabled substance ports must be connected.
- Connecting more than two substance ports together should use splitters and junctions from package Topology. Splitting directly using the connector is only recommended for the use of sensors.
- Each cyclic flow must be broken by at least one Substance component.

2 Definition

The previous version of Chemical library has various definitions of substances. In version 2.0, all these structures for all type of substances and even for all type of chemical processes are unified into one type of operator record called Chemical.Interfaces.Definition. This record can define each substance and process used in the previous version. In addition, the user can set their values independently on internal representation using suitable constructors or the function processData. This function creates process definition based on dissociation constant (molar-based) and consumed heat of the process (free molar enthalpy change) (**Figure 1**). Using relation between this process definition and its substances definitions, the user can easily evaluate new definition of the new chemical substance or new chemical process using expressions with '*', '+' and '-' operators.

For example, a definition of aqueous O2 can be set from gaseous O2 and water dissolution process with tabulated Henry's coefficient (0.0013, 1500K). For another example, a definition of H2O formation reaction from H2 and O2. Each new definition can be set as an algebraic equation between reaction, products and substrates (**Listing 1**).

Listing 1. Example of definitions

```
import Chemical.Interfaces.Definition;
import Chemical.Substances.Gas;
import Chemical.Substances.Liquid;
import Chemical.Interfaces.processData;
constant Real R = Modelica.Constants.R;

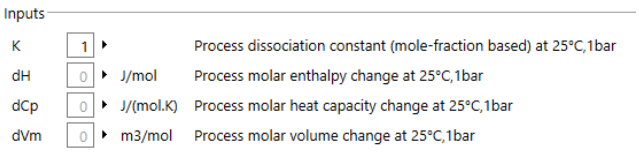
constant Definition O2_aq =
  Gas.O2 + processData(
              K = 0.0013,
             dH = -1500*R);

constant Definition H2O_formation =
  Gas.H2O - (Gas.H2 + 0.5*Gas.O2);

constant Definition Hemoglobin =
  Liquid.Unknown;
```

Each chemical pathway can start with Unknown substance. And next substance can be defined relatively by process definition. The default definition of Unknown substance is molar mass 1kg/mol and molar heat capacity 1 J/(mol.K), with zero free formation Gibbs energy and zero free formation enthalpy. These values are used if the user does not specify the substance. Because free formation energies are always defined in a relative manner, unknown substances can be used, such as any base substance with unknown formation process. If they play a role in the solution properties, at least definition data in relation with the changing solution properties should be set. E.g. molar mass and molar volume must be set if they play a role in solution composition; molar heat capacity must be set if it plays a role in heat accumulation in the chemical solution during simulation.

Figure 1. Inputs of processData function



3 Chemical solution

There are two options for how to define a chemical solution. The first involves the Chemical.Solution component and connecting all substances through its solution connector. This was the only option in the previous version, which was retained in this version. If a chemical solution is defined this manner, then all its properties are dynamically calculated during the simulation.

Listing 2. Example of fixed solution state

```
import Chemical.Interfaces.SolutionState;
import Chemical.Interfaces.Phase;

constant SolutionState SATP =
  SolutionState(phase = Phase.Gas,
                T = 298.15);

SolutionState heatingSolution =
  SolutionState(phase = Phase.Gas,
                T = 273.15+time);
```

However, there is not always a need to change the solution properties. If solution properties, such as temperature, pressure, total mass, total volume, total amount of particles, and electric potential, are assumed as constant during simulation, then the solution state (**Listing 2**) can

DOI
10.3384/ecp21855

be set as a parameter. If the user did not set a solution property, then its default value (**Figure *2***) will be used.

Figure 2. Default chemical solution properties

T	20	°C	Temperature of the solution
p	1.01325	bar	Pressure of the solution
m	1	kg	Mass of the solution
V	1	l	Volume of the solution
n	1	mol	Amount of the solution
v	0	V	Electric potential in the solution
G	0	J	Free Gibbs energy of the solution
Q	0	C	Electric charge of the solution
I	0	1	Mole fraction based ionic strength of the solution

4 Properties of chemical processes

Each process has different properties (e.g. dissociation coefficient, free Gibbs energy or free enthalpy) in dependence on its definition and on the current state of chemical solution (e.g. temperature, pressure, etc.). The user needs to connect the substance definition and solution to model Chemical.Interfaces.ProcessProperties to evaluate these properties.

Listing 3. Example of process properties

```
import Interfaces.ProcessProperties;

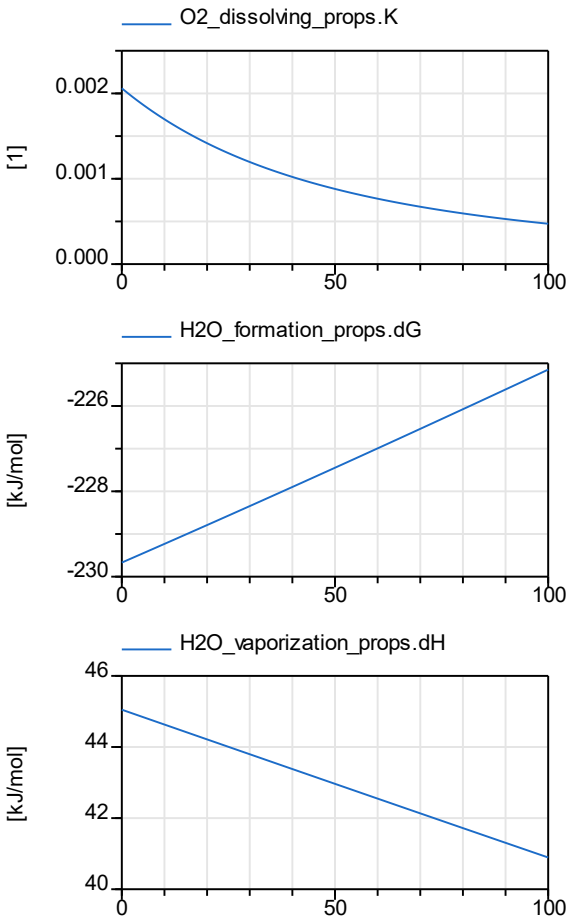
ProcessProperties O2_dissolving_props
(definition = processData(0.0013,-1500*R),
 solutionState = heatingSolution);

ProcessProperties H2O_formation_props
(definition = H2O_formation,
 solutionState = heatingSolution);

ProcessProperties H2O_vaporization_props
(definition = Gas.H2O - Liquid.H2O,
 solutionState = heatingSolution);
```

After the simulation of code (**Listing 1-3**) for 100 seconds, the temperature of the solution changed from 0°C to 100°C, and we can see the recalculated properties that resulted from temperature change such as dissociation constants (Henry's coefficient), free Gibbs energy change of reaction or consumed heat by process (free enthalpy change) – e.g. **Figure *3***.

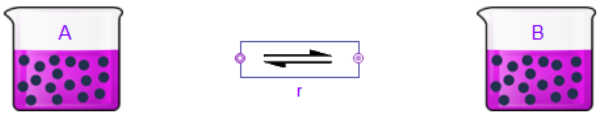
Figure 3. Results of simulation (**Listing 1-3**) per temperature [°C]



5 Chemical process

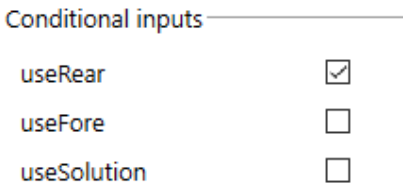
To create a model of a chemical reaction using Chemical 2.0, drag and drop component Substance from package Chemical.Boundaries as substance ***A*** and substance ***B***. In the same way, put into the model Reaction as process ***r*** from package Chemical.Processes (**Figure 4**).

Figure 4. Components of simple chemical reaction



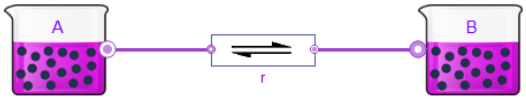
The default setting for the substances does not provide any connector, so it is necessary to setup which connectors will be used. This is done in parameter dialog by selecting the ***useFore*** checkbox for substance ***A*** and ***useRear*** for substance B (**Figure 5**).

Figure 5. Configuration of substance connectors



Having all necessary connectors, it is possible to connect substances with chemical reaction ***r*** **(Figure 6)**.

Figure 6. Connections



This model could be simulated, but everything remains constant because the dissociation coefficient is equal to one and both substances are the same (defined as Liquid.Unknown). However, if we change the dissociation coefficient ***K*** to 2 (as shown in **Listing 4**) then we can see dynamic behavior in the default chemical solution. And after 5s of simulation time, the reaction reaches almost the equilibrium where the amount of product ***B*** is two times higher than the amount of substrate ***A***. We can change reaction speed by setting forward reaction rate coefficient ***r.k_forward*** lower than its default value. Backward reaction rate is not possible to set as a reaction parameter because it is defined as ***k_forward/K***. This reaction can have as many substrates (***nS***) and as many products (***nP***) as needed, and its index is defined by the order of drawn connections. Note, that default chemical solution in this example is set to **Figure 2** and remains constant during simulation.

Listing 4. Generated code for a reaction example

```
model SimpleReaction
 extends Modelica.Icons.Example;

 import Chemical.Boundaries.Substance;
 import Chemical.Processes.Reaction;
 import Chemical.Interfaces.processData;

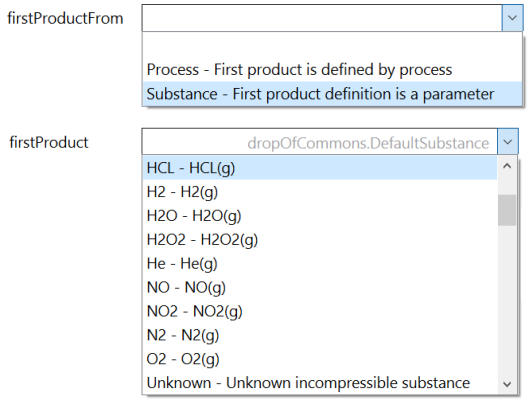
 Substance A( useFore = true );
 Substance B( useRear = true );

 Reaction r( nP = 1, nS = 1,
    process = processData(2) );

equation
  connect( A.fore , r.substrates[1] );
  connect( r.products[1] , B.rear );
end SimpleReaction;
```

If the user prefers products definition before definition of process, then it is possible to set process option ***firstProductFrom*** into ***"Substance"*** selection. This enables the parameter ***firstProduct*** (the first product definition) in the same dialog as the process setting. With knowledge of all products and all substrate definitions the process definition can be evaluated, so the process definition parameter is disabled during this selection. Please note, that each product is by default defined by process, so it is not necessary to define it again in following process or boundary component.

Figure 7. Process defined by substances



In addition, stoichiometric coefficients can be set as parameter ***s*** for substrates and parameter ***p*** for products as in previous version. The new extension for processes allows to change chemical kinetics as redefinition of relation between electro-chemical gradient and process molar rate called ***uLoss***. By default, ***uLoss*** is set to traditional chemical kinetics based on forward rate coefficient, but the user can define their own type of chemical kinetics constrained by proposed interface for potential loss functions.

Listing 4. Example of reaction defined by product with uncommon chemical kinetics

```
import Chemical.Utilities.Types.
         FirstProductChoice;
import Chemical.Processes.Internal.Kinetics;

Chemical.Processes.Reaction H2_burning (
    firstProductFrom =
        FirstProductChoice.Substance,
    firstProduct = Gas.H2O,
    nS=2, s = {2,1},
    nP=1, p = {2},

    redeclare function uLoss =
         Kinetics.fastPotentialLoss,

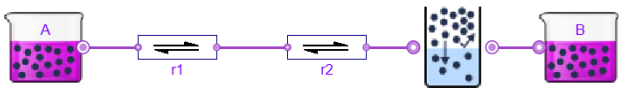
)
```


DOI
10.3384/ecp21855

6 Data propagation

The version 2.0 of the Chemical library is based on a special type of substance ports. These connectors propagate the state of substance without inertial part in flow direction and generate equality of inertial electro-chemical potential, instead of direct equality of the total electro-chemical potential. The equilibria and even the dynamic simulation reach almost the same values if inertia is fast enough. As proven by (Zimmer 2020) this type of connector has physical background and better performance because it eliminates non-linear systems of equations where solvers typically spend the most of time. Moreover, this design allows users to connect more processes together without any accumulation of substance (or just a few Substance components). As a result, it is possible to define long chemical pathways as are usual in physiology.

Figure 8, Example of possible pathway connection



To have a chemical pathway such as in **Figure *8*** there is a need to have each substance definition between each chemical process. This is done automatically using a propagation pattern from forward connectors (called fore) into rearwards connectors (called rear). Please note that substances definition propagation is not dependent on flow direction. Each substance definition remains constant during simulation, so the direction of definitions propagation does not take any role during simulation and vice versa. In **Figure *8*** only the substance ***A*** can be defined by user. The definition of a product of reaction ***r1*** is evaluated from the definition of this reaction and its substrates. So generally, each process can define the first product from its other products, substrates, and process definition. Even the substance ***B*** does not have to be explicitly defined, because its definition can come from PhaseTransition process.

In the same way, the fixed state of the chemical solution can be propagated. If the state of chemical solution is constant, it remains the same for each substance in the solution during each step of the simulation. In the model shown in **Figure *8*** it means that chemical solution is specified only in substance ***A*** and propagated trough connections to each other components as the same constant values.

To model the situation when processes cross different solutions, there is an option to break the solution propagation. In some processes, such as Diffusion, Membrane or PhaseTransition, the solution states are not propagated by default because they change the chemical solution between substrates and products. As a result, the new solution must be defined for the process and its product to propagate into next processes. Please note, that "next" here is not defined by molar flow but by fore-rear connections.

Listing 5. Example of process changing chemical solution

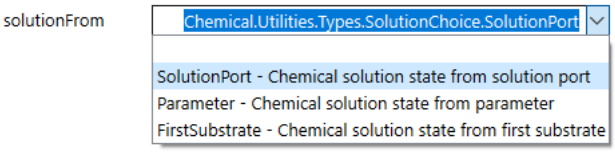
```
import Chemical.Processes.PhaseTransition;
import Chemical.Interfaces.SolutionState;
import Chemical.Interfaces.Phase;
import Chemical.Substances.Gas;

PhaseTransition PhaseTransition(
     solutionParam =
        SolutionState( phase = Phase.Gas ),
     Product = Gas.CO2 );
```

Diffusion and Membrane process have the same substance on both sides. PhaseTransition changes the substance. This allows to use it even for solvation (gas dissolution in liquids and gas volatility from liquids) or for phase transition (vaporization, condensation, sublimation, deposition, melting and freezing). Note that the definition of each chemical compound is dependent on phase. For example, gaseous water was defined differently from liquid water. Phase transition of a substance is a chemical process, which even consumes/releases heat (has non-zero free enthalpy change).

Please note, that if a chemical substance significantly changes the chemical solution properties, then it must be connected to Solution component using ***SolutionPort*** instead of fixed parametrized solution propagation. This situation is selected by choice ***solutionFrom*** (**Figure *9***).

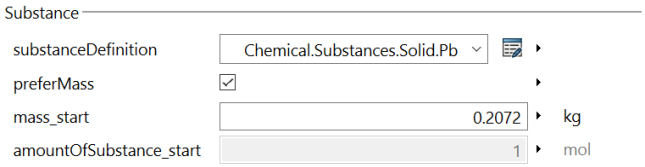
Figure 9. If substance change the solution



7 Initialization

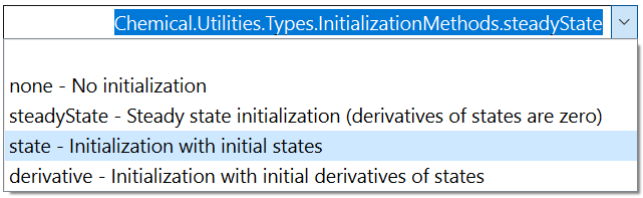
Substance component is initialized by initial mass or amount of base substance molecules. Recalculation from number of base molecules and mass is obtained through molar mass, which is part of the substance definition. Switching between mass-based and molar-based accumulations is based on ***preferMass*** checkbox (**Figure *10***). If mass is preferred, ***amountOfSubstance_start*** is disabled and ***mass_start*** is enabled as an initial value of substance.

Figure 10. Substance amount initialization



A principle of the inertial connectors affects the initialization of the model processes. Having acceleration of molar flows on each connection, each flow has a state. Therefore, the simplest process initialization is to set values of molar flows (**Figure *11***). From these values the gradients of electro-chemical potentials on process components are calculated as a relations of substance states between boundaries.

Figure 11. Molar flow initialization



Alternatively, it is possible to set the initial process state to steady state. This will take inertial electrochemical potential change ***r*** to zero. If each process has this type of initialization, then the solver is looking for the non-accelerated state of whole system, where derivation of each flow is zero.

8 Discussion

ThermofluidStream is a proof of concept that even a complex models could improve usability, performance, details and interfaces. We hope that the update of Chemical library from version 1.4 to version 2.0 marks also this kind of a progression.

Despite the complexity of this code containing many physical equations and parameters, the user interface is simplified significantly. It is permitting users to start with basic models without any significant alterations to the default component settings. And it works, because common (electro-)chemical assumptions are hidden as these default values. Starting with a very simple model and extending significant details during each modeling step leads to optimal model development with minimal effort. This development strategy can be reversed if something does not work – model can be simplified and results should be also well interpretable.

Theory behind Chemical library at the level of physical chemistry is summarized in documentation found at: Chemical\Resources\Documentation\Chemical.pdf.

Acknowledgements

Special thanks to Taras (Terry) Yavorskyy and Dr. Jason Brown, who contributed by proof reading this article for English grammar and syntax.

References

Casella, Francesco, Martin Otter, Katrin Proelss, Christoph Richter, and Hubertus Tummescheit. 2006. “The Modelica Fluid and Media Library for Modeling of Incompressible and Compressible Thermo-Fluid Pipe Networks.” In *Proceedings of the 5th International Modelica Conference*, 631–40.

Mateják, Marek, Martin Tribula, Filip Ježek, and Jiří Kofránek. 2015. “Free Modelica Library of Chemical and Electrochemical Processes.” In *11th International Modelica Conference, Versailles, France*, 118:359–66. Linköping University Electronic Press, Linköpings universitet.

McBride, BJ, MJ Zehe, and G Sanford. 2002. “Glenn Coefficients for Calculating Thermodynamic Properties of Individual Species.” *Report: NASA/TP* 211556.

Meißner, Michael, and Dirk Zimmer. 2022. “Robust Modeling of Volumes for Dynamic Simulations of Thermo-Fluid Stream Networks.” *IFAC-PapersOnLine* 55 (20): 265–70.

Peter Atkins, P, J De Paula, and James Keeler. 2018. *Atkins’ Physical Chemistry*. OUP Oxford.

Sauro, Herbert M. 2009. “Network Dynamics.” *Computational Systems Biology*, 269–309.

Zimmer, Dirk. 2020. “Robust Object-Oriented Formulation of Directed Thermofluid Stream Networks.” *Mathematical and Computer Modelling of Dynamical Systems* 26 (3): 204–33.

Zimmer, Dirk, Daniel Bender, and Alexander Pollok. 2018. “Robust Modeling of Directed Thermofluid Flows in Complex Networks.” In *Proceedings of the 2nd Japanese Modelica Conference*, 39–48. Linköping University Press.

Zimmer, Dirk, Michael Meißner, and Niels Weber. 2021. “Robust Simulation of Stream-Dominated Thermo-FluidSystems: From Directed to Non-Directed Flows.” *SNE Simulation Notes Europe* 31 (4): 177–84.

Zimmer, Dirk. 2022. “The DLR Thermofluid Stream Library.” *Electronics* 11 (22): 3790.

Pharmacolibrary - Free Library to Model Pharmacology.

Tomáš Kulhánek[1,3,4] Filip Ježek[2,3] Jiří Kofránek[3,4] Marek Mateják[3,5] Stef Rommes[1]

[1]VITO, Mol, Belgium
[2]University of Michigan, Ann Arbor, USA
[3]First Faculty of Medicine, Charles University, Prague, Czechia
[4]Creative Connections, s.r.o., Prague, Czechia
[5]Institute for Clinical and Experimental Medicine, Prague, Czechia

Abstract

Mechanistic modeling of drug behavior and response is essential for rational drug development and personalized therapy, yet constructing, maintaining, reusing and customizing complex pharmacokinetic–pharmacodynamic and physiologically based pharmacokinetic models can be error-prone when implemented solely via equations or code. We introduce Pharmacolibrary, a free Modelica library offering standardized acausal components for pharmacokinetics, pharmacodynamics, toxicokinetics/toxicodynamics and pharmacogenomics from compartmental and physiologically based templates to effect models and genotype–phenotype records—to simplify model reuse, customization, and interoperability. Its utility is showcased with gentamicin, midazolam, and fentanyl case studies, including pharmacogenomics-driven clearance adjustments and pharmacodynamics simulations.
Keywords: Modelica, Pharmacology, Pharmacokinetics, Pharmacodynamics, Pharmacogenomics, Modeling

1 Introduction

Pharmacology investigates how drugs modulate biological functions—spanning their origin, mechanisms of action, therapeutic benefits, and adverse effects—while toxicology focuses on chemical-induced harm. Core subfields include pharmacokinetics (PK; absorption, distribution, metabolism and elimination processes of a drug in organism) and pharmacodynamics (PD; effect, concentration–response relationships), with analogous toxicokinetics/toxicodynamics (TK/TD) in the toxicity domain. Pharmacogenomics (PGx) quantifies how genetic variation shapes PK and PD. Physiologically based pharmacokinetics (PBPK) augments classical PK with anatomical and physiological detail to predict drug distribution(Rosenbaum 2016). Quantitative systems pharmacology (QSP) further integrates PK/PD with systems-biology to capture drug interactions within complex molecular networks(Azer et al. 2021).

A variety of commercial and academic tools support these methods: Certara's SimCyp(Jamei et al. 2009) and Bayer's PK-Sim/MoBi for PBPK and Open-system pharmacology community(Lippert et al. 2019); MediWare's MWPharm++ for pharmacogenetically informed therapeutic drug monitoring(Schön et al. 2022); MATLAB SimBiology® and its gPKPDSim extension for PK/PD workflows(Hosseini et al. 2018); and Modelica-based libraries (Biochem, Physiolibrary, Chemical) for pathway, physiological, and physicochemical modeling. However, these platforms either specialize in a single aspect or blend modeling with simulation in proprietary environments.

Some of the existing Modelica libraries can be currently used to mimic e.g pharmacokinetics like the Biochem library for biochemical pathways(Nilsson and Fritzson 2005), Physiolibrary for human physiology models(Mateják et al. 2014) or Chemical for using electrochemical potential and physical chemistry principles to model chemical reaction and behavior of various substances(Matejak et al. 2015). However, these libraries focus on different aspects of biology, physiology and physical chemistry.

Here, we introduce *Pharmacolibrary*, a reusable Modelica library unifying PK, PD, PBPK, TK/TD, and PGx constructs with standardized pharmacological terminology, units, and acausal connectors. By decoupling model definition from simulation, our library leverages Modelica's auto-generation of correct equations from diagrams, supports FMU export for integration into Python/Julia/-MATLAB/WebApp workflows, and interoperates with existing Modelica libraries to enable advanced QSP applications. In the following sections, we describe the library's components and demonstrate exemplar models that reproduce experimentally observed drug behaviors.

2 Pharmacolibrary

As a foundational concept for pharmacology modeling, we first define the following connectors, which hold and exchange drug mass and concentration via various types between components.

The ConcentrationPort connector (table 1) contains MassFlowRate as a flow variable and MassConcentration as a non-flow variable that is usually used to model chemical domain. The ConcentrationPort is intended to be used in compartment based pharmacokinetics model where the volumetric flow is not taken into account or is neglected and ideal mixing can be assumed among connected components.

The FlowPort connector (table 2) contains Volume-

	**ConcentrationPort** is acausal connector that contains mass flow rate quantity q_m and concentration c defined as follows:
	
	`flow MassFlowRate qm;`
	`MassConcentration c;`
	
	When a component is connected using such a connector, standard Modelica tool will ensure that flow variable q_m is not accumulated in connected connectors by ensuring that $\sum_{i=0}^{n} q_{m_i} = 0$ and also that all non-flow variables - in this case concentration c are equilibrated i.e. $c_1 = c_2 = ... = c_n$ (n is number of connected components via a connector).

Table 1. Graphical connectors for drug-mass and concentration exchange.

FlowRate as a flow variable and MassConcentration as a stream variable and Pressure as a non-flow variable. The combination of VolumeFlowRate and Pressure is usually used to model pressure-volume hydraulic domain, we additionally add a concentration. The FlowPort is intended to be used in physiology based pharmacokinetics model (PBPK) where the substance (e.g., drug) transport is resolved through branching and merging conduits, automatically enforcing mass-balanced mixing at junctions according to the instantaneous flow rates.

	**FlowPort** is acausal connector that contains flow quantity q_v, stream quantity c and non-flow quantity p defined as follows:
	
	`flow VolumeFlowRate qv;`
	`stream MassConcentration c;`
	`Pressure p;`
	
	Using such definition, when a component is connected using a connector, standard Modelica tool will ensure that flow variable q_v is not accumulated in connected connectors by ensuring that $\sum_{i=0}^{n} q_{v_i} = 0$ and additionally all variables in stream connector (in this case concentration c) are ensured to be correctly and physically consistently mixed based on the flow quantity. All non-flow (potential) quantities (in this case pressure p) are equalized among connected components.

Table 2. Graphical connectors for drug-mass and concentration exchange in case of volumetric flow and mixing is taken into account.

By employing these two connector types, we can distinguish between pure compartment based models (where active flow and compartmental distribution can be simplified to diffusion kinetic) and PBPK models (where drug particles are exchanged via active distribution, i.e. volume flow rate and compartment are matched with anatomical and/or physiological tissues or systems).

2.1 Pharmacokinetics - compartment models

Pharmacokinetics studies how drug is absorbed, distributed, metabolized, and excreted (the “ADME” processes). The compartmental pharmacokinetic (PK) model abstracts the organism as one, two, or several interconnected compartments through which the drug distributes, with exchange between compartments generally described by first-order kinetics.(Fleishaker and Smith 1987) A compartment need not map to a specific anatomical tissue or organ; its defining parameter—the apparent volume of distribution V_d—is usually obtained by numerically fitting the time-course of blood-concentration data after drug administration.

By measuring drug concentrations in blood, tissues, or other biological fluids, pharmacokinetics compartmental analysis seeks to characterize the parameters of the components involved in ADME processes. For absorption the known parameters are dose mass, dose duration, administration route and unknown is bioavailability F. For distribution the volume distribution V_d of compartments. For Metabolism and Excretion the kinetic rates and clearance rates.

For modeling mainly the ’Distribution’ part of ADME processes in pharmacokinetics, we define the reusable compartment components as in table 3. For modeling the rest of the ADME processes, we define additional components as in table 4.

	**NoPerfusedTissueCompartment** expresses basic theoretical volumetric distribution of a drug in a hypothetical compartment, that does not necessarily be equivalent to an anatomical tissue or organ. With the following equations: $$\frac{dM}{dt} = q_m$$ $$C = \frac{M}{V_d}$$ Where derivation of M is mass flow rate q_m and concentration C is mass of drug in compartment M divided by compartment’s volume of distribution V_d.
	**SystemicTissueCompartment** express volumetric distribution of a drug via arterial or venous bloodstream through 2 flowports and 1 concentration port using these main equations: $$\frac{dM}{dt} = \sum q_{vxi} \times \text{actualStream}(c_{xi}) + q_m$$ $$c_a = c_b = \frac{M}{V_d}$$ where M is drug total mass in compartment, parameter V_d is volume of distribution, $q_v a$ and $q_v b$ are distribution of volume flow rates in ports a and b and q_m is mass flow rate that may happen directly in the blood tissue. c_a and c_b are actual concentrations on connected ports a and b respectively.
	**TissueCompartment** expresses not only distribution of a drug through bloodstream, but additional a simple constant tissue-blood concentration ratio k_{TB} as $$c = \frac{c_t}{k_{TB}}$$

Table 3. Compartment components.

DOI
10.3384/ecp21861

	**TransferFirstOrderNonSym** express basic first order non-symetric transfer between compartments defined by intercompartmental clearance parameters Cl_a and Cl_b.
	**SingleDose** expresses a source that contributes to concentration change by administering a certain amount of drug during administering duration into the 1 concentration port. $$q_m = \frac{\text{adminMass}}{\text{duration}}$$
	**FlowGround** The component FlowGround express the ground of concentration within the stream from the flowport connector. Note, that this component does not affect the model behavior, as no volume normally flows in or out, yet is required for mathematical completeness. An error is raised when any flow leaks or is added inadvertently.
	**ClearanceDrivenElimination** expresses simplified elimination that is driven by the clearance parameter by the following equation: $$q_m = CL \times c$$ $$\frac{dM_{exc}}{dt} = q_m$$ Where M_{exc} is excreted drug mass, CL is clearance parameter usually in liters per minute, c is drug concentration and q_m is massflowrate of the drug that is eliminated.
	**FixedFlow** express enumerated flow that is forced to be through this component. It can be used e.g. to mimic cardiac output or partial flow within a specific organ or tissue. It's parameter is prescribed flow Q which is then given to the connected ports a and b as: $$q_{va} = -q_{vb} = Q$$

Table 4. Additional components

2.2 1-compartment models

The single-compartment pharmacokinetic (PK) model is a theoretical model where whole organism is represented as a single, well-mixed compartment. In this model, the drug is absorbed according to its bioavailability, distributed uniformly throughout the apparent volume of distribution, and eliminated at a constant rate. Such a PK model can be described by the following integral equation of concentration c over time t where initial concentration c_0 and clearance rate of elimination Cl:

$$c = c_0 \times e^{-Cl \cdot t} \tag{1}$$

or in derivative form

$$\frac{dc}{dt} = -Cl \times c \tag{2}$$

When a mass of a drug is taken into account M, a theoretical parameter "volume distribution" V_d is in equation:

$$c = \frac{M}{Vd} \tag{3}$$

So the PK model can be described as

$$\frac{dM}{dt} = -Cl \times c \tag{4}$$

In Modelica language, the equations (3) and (4) can be implemented directly as seen in model implementation fig.1 (A). However the component NonPerfusedTissueComponent (implementing equation (3)) is connected with ClearanceDriveElimination which implements and correctly connect equation (4) so the diagram mode Fig.1 (B) is equivalent to the text mode of implementation.

(A)

```
within Pharmacolibrary.Examples;

model PK_1C_GentamicinEquations
  extends Modelica.Icons.Example;
  Pharmacolibrary.Types.Mass M(start=0.000400) "Mass of
drug [kg]";
  Pharmacolibrary.Types.MassConcentration C
(displayUnit="mg/l") "Concentration of drug";
  parameter Pharmacolibrary.Types.Volume Vd = 0.0175
"volume of distribution [m3]";
  parameter Pharmacolibrary.Types.VolumeFlowRate Cl = 0.1/
(1000*60) "elimination clearance [m3/s]";
equation
  C = M / Vd; //(12)
  der(M) = - Cl * C; //(13)
  annotation( ...);
end PK_1C_GentamicinEquations;
```

(B)

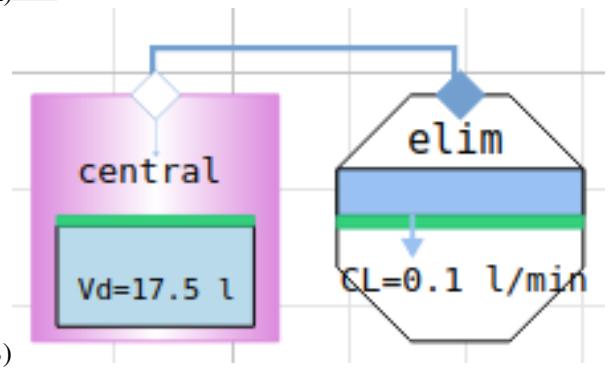


Figure 1. Equivalent representations of the single-compartment model. (A) Text mode, in which the governing equations (3) and (4)—are written out explicitly. Text mode is ideal for simple models or for defining individual components. (B) Diagram mode, which links predefined component icons into a acausal component diagram. Diagram mode is preferred when domain-specific components are already well defined and can be reused and composed in different settigns. Example parameters of PK of gentamicin with an initial dose of 400 mg, a clearance of 0.1 L/min, and a volume of distribution of 17.5 L. In text mode, the parameter values are declared in basic SI units. In diagram mode, tools such as OpenModelica(Fritzson et al. 2020) allow user to chose the prefered unit and automatically recalculate and display all units in SI within the component icons, dialogs and plots.

This model requires setting the initial drug mass at time zero. To incorporate single-dose absorption, a SingleDose component can be connected as shown in the modified diagram in Fig 2.

Single-compartment models describe the PK of a drug whose distribution via the cardiovascular arterial tree system can be neglected—typically because they are absorbed and distributed rapidly and do not significantly bind to tissues.

For example, gentamicin is an antibiotic administered intravenously, usually once daily, to treat serious bacterial infections. Sample parameter values of gentamicine's PK model are given in Figure 1 and its caption. A primary concern in drug administration is efficacy; for gentamicin, the minimum inhibitory concentration (MIC) represents the lowest concentration that prevents bacterial growth. Another concern is toxicity: gentamicin can cause nephrotoxicity (kidney toxicity) and ototoxicity (ear toxicity). The toxicity for gentamicine should not reach a $C_{tox,peak}$ value for short term toxic effect and minimal concentration before second dose should be bellow $C_{tox,valley}$ or sometimes dedicated as $C_{tox,trough}$. The dosing goal for clinical pharmacologist can be described as simplified rule to maintain drug concentrations above the MIC for a sufficient portion of the dosing interval while keeping peak level and trough level below toxic thresholds. For gentamicin the MIC and toxicity thresholds are in Fig.2(C).

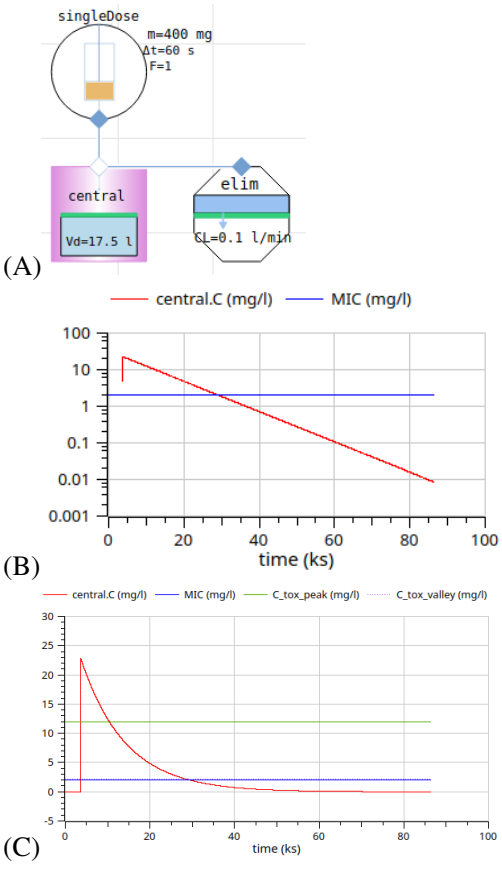


Figure 2. (A) Single-compartment model with a SingleDose component, a central compartment, and an elimination component. (B) Simulated concentration plotted on a logarithmic scale, illustrating the model's constant log-linear decline. (C) Example simulation configured for gentamicin pharmacokinetics—initial dose 400 mg, volume of distribution 17.5 L, clearance 0.1 L/min—showing the central compartment concentration (red), the minimum inhibitory concentration (MIC)(Hodiamont et al. 2022), and observed peak and valley (trough) toxicity levels (green) and (dotted blue).

2.3 2-compartment models

If the experimental concentration–time curve is biphasic, a two-compartment model usually fits better: the central compartment represents blood and well-perfused tissues, while the peripheral compartment represents tissues that transiently store the drug. Figure 3(A) shows this arrangement—administration and elimination occur from the central compartment, and drug moves between compartments by first-order transfer.

Midazolam—a fast-acting benzodiazepine used for sedation, anxiolysis, and anesthesia induction—exemplifies a drug whose intravenous pharmacokinetics are well described by a two-compartment model. Sample PK parameters are in Figure 3 caption. The plasma concentration required for anesthesia induction must remain above a defined minimal effective threshold C_{min}.

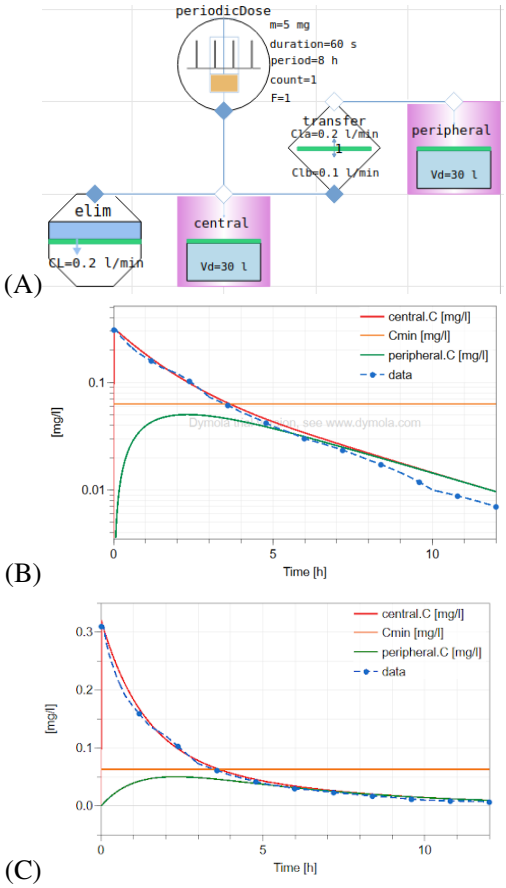


Figure 3. (A) Diagram view of a two-compartment pharmacokinetic model, showing drug administration and elimination in the central compartment with bi-directional transfer to the peripheral compartment. (B) Simulation on a logarithmic scale, illustrating the non-constant log-linear decline of concentrations in both compartments. validated against data (blue dots) from Heizmann et al. (1983)(Heizmann, Eckert, and Ziegler 1983). (C) Simulation of midazolam concentration in the central (red) and peripheral (green) compartments, alongside the minimal effective concentration (blue). A 9.6 mg IV dose is administered; both compartments have an apparent volume of distribution $V_d = 30l$, inter-compartmental transfer rate constants $k_{12} = 0.2h^{-1}$ and $k_{21} = 0.1h^{-1}$ and a systemic clearance $Cl = 0.2l.min^{-1}$

DOI
10.3384/ecp21861

2.4 Multicompartmental models

When a drug distributes into multiple tissue types, its post-administration concentration decline exhibits more complex behavior, making multi-compartment models more appropriate than single- or two-compartment models.

Fentanyl is a potent synthetic opioid used to manage severe pain and as an anesthetic adjunct during surgery. After intravenous administration, it rapidly distributes into highly perfused organs—such as the brain, heart, lungs, and kidneys—while also partitioning more slowly into poorly perfused tissues like fat and muscle. Over time, both tissue groups release fentanyl back into the plasma. The pharmacokinetics of fentanyl can be captured by the three-compartment model shown in Figure 4, in which the "peripheral" and "peripheral2" compartments represent highly and poorly perfused tissues, respectively.

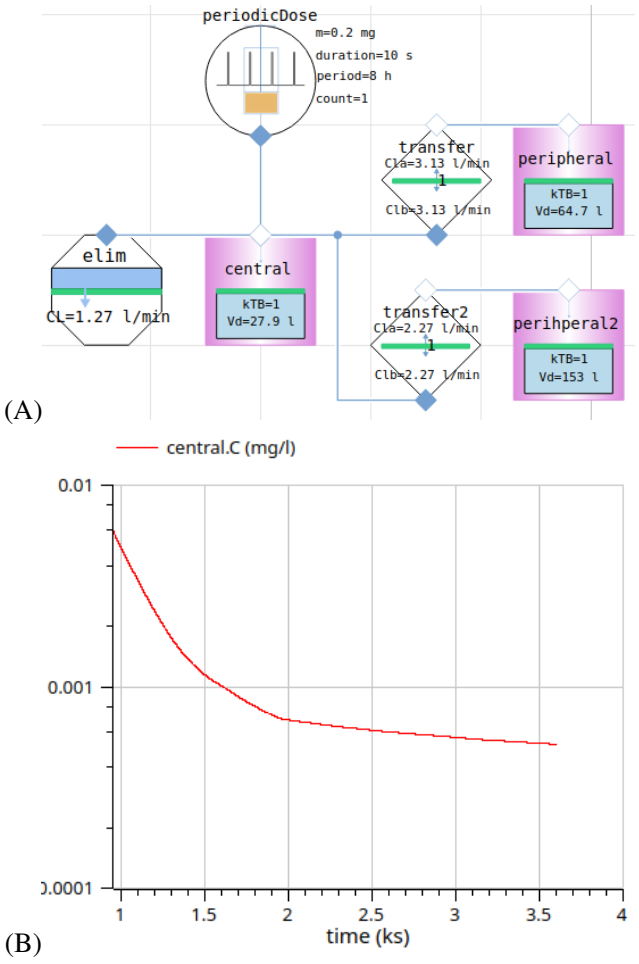


Figure 4. (A) Three-compartment PK model diagram. (B) Log-scale simulation of fentanyl plasma concentration 10–60 min after a 0.2 mg bolus. Parameters: V_d=27.9L (central), 64.7L (peripheral$_1$), 153L (peripheral$_2$); Cl=1.27, 3.13, 2.27L min^{-1}, respectively(Kaneda and Han 2009). The curve inflects as both peripheral pools return drug to the central compartment.

2.5 Physiology based pharmacokinetics models - component models

Compartmental PK models fit empirical compartments to concentration–time data but lack anatomical meaning. Physiologically based PK (PBPK) models instead assign compartments to real organs, enabling organ-level and inter-individual simulations when parameters are available.

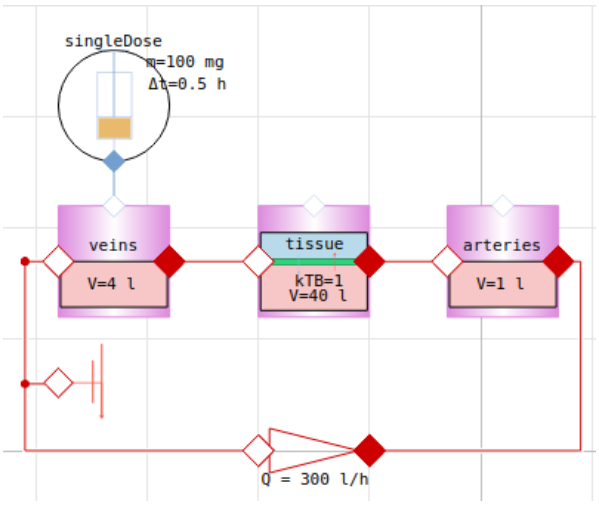


Figure 5. Model diagram of three-component PBPK model. Linked red FlowPorts enforce mass flow rate and resolve stream variables by flow direction. ConcentrationPorts join SingleDose→Veins (IV 100 mg in 0.5 h). A dedicated block sets unidirectional flow at a cardiac output of 5 L min^{-1} (300 L h^{-1})

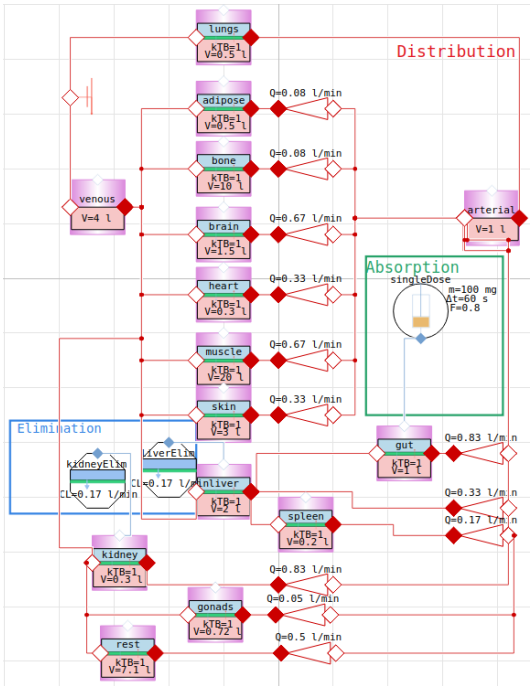


Figure 6. Whole-body PBPK model, utilizing main tissues with normal blood flow distribution. This is suitable for extension by a particular drug.

```
322  equation
323    kidney.cport.c = kidneyElim.cport.c;
324    liver.cport.c = liverElim.cport.c;
325    singleDose.cport.c = gut.cport.c;
326    gut.cport.qm + singleDose.cport.qm = 0.0;
327    venous.cport.qm = 0.0;
328    lungs.port_a.qv + flowGround.port_a.qv + venous.port_a.qv = 0.0;
329    lungs.cport.qm = 0.0;
330    arterial.port_b.qv + lungs.port_b.qv = 0.0;
331    arterial.cport.qm = 0.0;
332    restFlow.port_a.qv + testesFlow.port_a.qv + fixedFlow9.port_a.qv + fixedFlow8.port_a.qv + fixedFlow7.port_a.qv + fixedFlow6.port_a.qv +
   fixedFlow5.port_a.qv + fixedFlow4.port_a.qv + fixedFlow3.port_a.qv + fixedFlow2.port_a.qv + fixedFlow1.port_a.qv + fixedFlow.port_a.qv +
   arterial.port_a.qv = 0.0;
333    adipose.port_b.qv + fixedFlow.port_b.qv = 0.0;
334    adipose.cport.qm = 0.0;
335    rest.port_a.qv + gonads.port_a.qv + kidney.port_a.qv + liver.port_a.qv + skin.port_a.qv + muscle.port_a.qv + heart.port_a.qv + brain.port_a.qv +
   bone.port_a.qv + adipose.port_a.qv + venous.port_b.qv = 0.0;
336    bone.port_b.qv + fixedFlow1.port_b.qv = 0.0;
337    bone.cport.qm = 0.0;
338    brain.port_b.qv + fixedFlow2.port_b.qv = 0.0;
339    brain.cport.qm = 0.0;
340    heart.port_b.qv + fixedFlow3.port_b.qv = 0.0;
341    heart.cport.qm = 0.0;
342    muscle.cport.qm = 0.0;
343    fixedFlow4.port_b.qv + muscle.port_b.qv = 0.0;
344    skin.cport.qm = 0.0;
345    fixedFlow5.port_b.qv + skin.port_b.qv = 0.0;
346    fixedFlow6.port_b.qv + gut.port_b.qv = 0.0;
347    spleen.cport.qm = 0.0;
348    liverElim.cport.qm + liver.cport.qm = 0.0;
349    fixedFlow7.port_b.qv + liver.port_b.qv + spleen.port_a.qv + gut.port_a.qv = 0.0;
350    fixedFlow8.port_b.qv + spleen.port_b.qv = 0.0;
351    kidney.port_b.qv + fixedFlow9.port_b.qv = 0.0;
352    kidneyElim.cport.qm + kidney.cport.qm = 0.0;
353    gonads.cport.qm = 0.0;
354    testesFlow.port_b.qv + gonads.port_b.qv = 0.0;
355    rest.cport.qm = 0.0;
356    restFlow.port_b.qv + rest.port_b.qv = 0.0;
357    singleDose.cport.qm = if singleDose.adminTime <= time and time < singleDose.adminTime + singleDose.duration then -singleDose.F * singleDose.adminMass /
   singleDose.duration else 0.0;
358    der(singleDose.TotalCumulativeMass) = -singleDose.cport.qm;
359    der(venous.M) = smooth(0, venous.port_a.qv * (if venous.port_a.qv > 0.0 then $OMC$inStreamDiv(($OMC$PositiveMax(-flowGround.port_a.qv,
   1.6666666666666666e-15) * flowGround.port_a.c + $OMC$PositiveMax(-lungs.port_a.qv, 1.6666666666666666e-15) * lungs.port_a.c) / ($OMC$PositiveMax(-
   flowGround.port_a.qv, 1.6666666666666666e-15) + $OMC$PositiveMax(-lungs.port_a.qv, 1.6666666666666666e-15)), venous.port_a.c) else venous.port_a.c)) +
   smooth(0, venous.port_b.qv * (if venous.port_b.qv > 0.0 then $OMC$inStreamDiv(($OMC$PositiveMax(-adipose.port_a.qv, 1.6666666666666666e-15) *
   adipose.port_a.c + $OMC$PositiveMax(-bone.port_a.qv, 1.6666666666666666e-15) * bone.port_a.c + $OMC$PositiveMax(-brain.port_a.qv, 1.6666666666666666e-15)
   * brain.port_a.c + $OMC$PositiveMax(-heart.port_a.qv, 1.6666666666666666e-15) * heart.port_a.c + $OMC$PositiveMax(-muscle.port_a.qv,
   1.6666666666666666e-15) * muscle.port_a.c + $OMC$PositiveMax(-skin.port_a.qv, 1.6666666666666666e-15) * skin.port_a.c + $OMC$PositiveMax(-liver.port_a.qv,
   1.6666666666666666e-15) * liver.port_a.c + $OMC$PositiveMax(-gonads.port_a.qv, 1.6666666666666666e-15) * gonads.port_a.c + $OMC$PositiveMax(-
   kidney.port_a.qv, 1.6666666666666666e-15) * kidney.port_a.c + $OMC$PositiveMax(-rest.port_a.qv, 1.6666666666666666e-15) * rest.port_a.c) /
   ($OMC$PositiveMax(-adipose.port_a.qv, 1.6666666666666666e-15) + $OMC$PositiveMax(-bone.port_a.qv, 1.6666666666666666e-15) + $OMC$PositiveMax(-
   brain.port_a.qv, 1.6666666666666666e-15) + $OMC$PositiveMax(-heart.port_a.qv, 1.6666666666666666e-15) + $OMC$PositiveMax(-muscle.port_a.qv,
   1.6666666666666666e-15) + $OMC$PositiveMax(-skin.port_a.qv, 1.6666666666666666e-15) + $OMC$PositiveMax(-liver.port_a.qv, 1.6666666666666666e-15) +
   $OMC$PositiveMax(-gonads.port_a.qv, 1.6666666666666666e-15) + $OMC$PositiveMax(-kidney.port_a.qv, 1.6666666666666666e-15) + $OMC$PositiveMax(-
   rest.port_a.qv, 1.6666666666666666e-15)), venous.port_b.c) else venous.port_b.c)) + venous.cport.qm;
360    venous.port_a.c = venous.CB;
361    venous.port_b.c = venous.CB;
362    venous.port_a.qv + venous.port_b.qv = 0.0;
363    venous.C = venous.M / venous.VNonZero;
364    venous.CB = venous.C / venous.kTB;
365    venous.freeTissueConc = venous.fu * venous.C;
366    venous.freeBloodConc = venous.fu * venous.C / venous.kTB;
367    venous.cport.c = venous.freeTissueConc;
368    flowGround.port_a.c = 0.0;
369    assert(abs(flowGround.port_a.qv) < 1e-15, "some flow is lost");
370    der(lungs.M) = smooth(0, lungs.port_a.qv * (if lungs.port_a.qv > 0.0 then $OMC$inStreamDiv(($OMC$PositiveMax(-flowGround.port_a.qv,
   1.6666666666666666e-15) * flowGround.port_a.c + $OMC$PositiveMax(-venous.port_a.qv, 1.6666666666666666e-15) * venous.port_a.c) / ($OMC$PositiveMax(-
   flowGround.port_a.qv, 1.6666666666666666e-15) + $OMC$PositiveMax(-venous.port_a.qv, 1.6666666666666666e-15)), lungs.port_a.c) else lungs.port_a.c)) +
   smooth(0, lungs.port_b.qv * (if lungs.port_b.qv > 0.0 then arterial.port_b.c else lungs.port_b.c)) + lungs.cport.qm;
371    lungs.port_a.c = lungs.CB;
372    lungs.port_b.c = lungs.CB;
```

Figure 7. Excerpt from the flattened model—the text form automatically generated from the PBPK diagram by a standard Modelica tool. At line 332 the flows from all connected tissues sum to zero, and at line 359 the derivative of venous drug mass equals the flow-weighted mix of incoming concentrations, as dictated by the stream-connector logic. In a purely equation-based implementation these relations would have to be coded manually.

The basic three-component PBPK model, which accounts for cardiac output and drug distribution through the arterial and venous systems to target tissues, is illustrated in Figure 5.

Complex PBPK model follows cardiovascular distribution across multiple tissues (Figure 6). Linked red FlowPorts connect venous and arterial pools to fourteen organ/tissue compartments, automatically enforcing mass balance and setting stream variables by flow direction. A SingleDose block drives gut absorption (oral 100 mg, F = 0.8), while hepatic and renal clearance components (Cl=0.17 L min^{-1} each) handle elimination. A dedicated component sets cardiac output at 5 L min^{-1} (300 L h^{-1}). Figure 8 shows simulated tissue concentrations. An equivalent text model must state every equation and link, as illustrated by the flattened equation-based model generated by the Modelica tool in Figure 7.

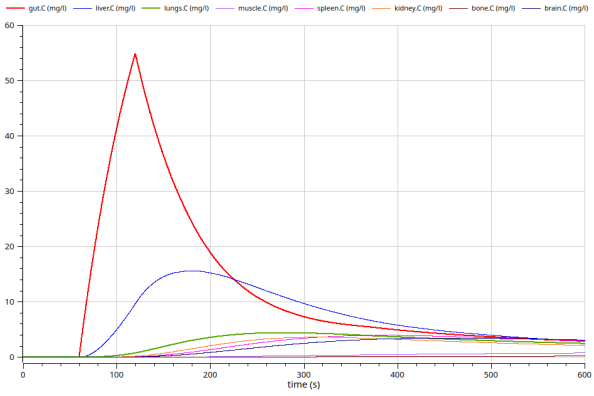


Figure 8. Simulation of the first 10 minutes of a generic drug using physiology-based pharmacokinetic model from Fig.6. Simulation reveals that majority of the drug administered orally goes first through liver and then it is subsequently distributed to other tissues.

DOI
10.3384/ecp21861

2.6 Pharmacodynamic models

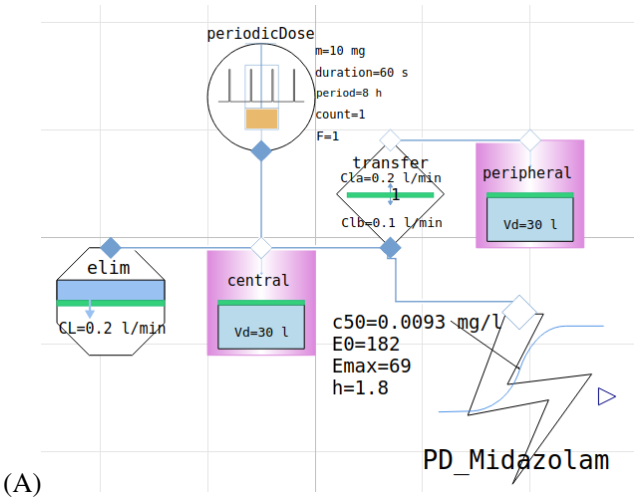
Pharmacodynamics (PD) is the bridge that converts "how much drug is present" into "how much effect is produced." In its direct or static form, the instantaneous drug concentration at the site of action is passed through an algebraic dose-response relationship. We present several such direct-effect components (Table 5): a linear effect component for low-dose proportionality, a hyperbolic E_{max} effect component for simple saturation, and a sigmoid-E_{max} effect component whose Hill exponent tunes the steepness of the transition. These components assume the effect is reversible: when the concentration falls, the response follows it back to baseline with no memory.

	**LinearEffect** depicted as equation $$E = E_0 + S \times C$$ Where E is effect, E_0 is baseline effect without drug influence, S is slope (effect per concentration), C is drug concentration.
	**EmaxEffect** Depicted by the equation $$E = E_0 + \frac{E_{max} \times C}{C_{50} + C}$$ where E is the final effect, E_0 is the baseline effect in the absence of drug, C is the drug concentration, C_{50} is the concentration producing 50% of the maximal effect E_{max}, and E_{max} is the maximal effect.
	**SigmoidEmaxEffect** Depicted by the equation $$E = E_0 + \frac{E_{max} \times C^h}{C_{50} + C^h}$$ where E is the final effect, E_0 is the baseline effect in the absence of drug, C is the drug concentration, C_{50} is the concentration producing 50% of the maximal effect E_{max}, E_{max} is the maximal effect, and h is the Hill exponent.

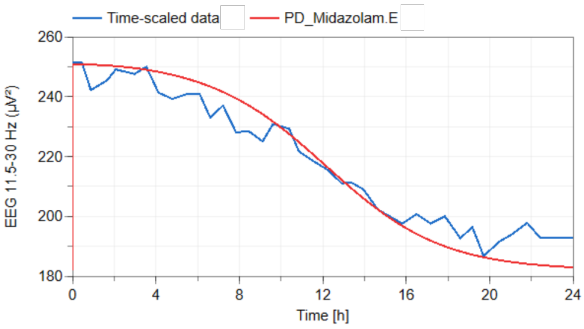
Table 5. Graphical connectors for drug-mass and concentration exchange.

Some drugs act irreversibly—e.g., covalent enzyme inhibition, receptor down-regulation, or DNA alkylation—so their effect depends on cumulative target inactivation rather than current concentration; these indirect, irreversible mechanisms are not yet covered in the current library.

Midazolam—whose intravenous disposition is captured by the two-compartment PK model in Figure 3—rapidly partitions into the highly perfused central nervous system, where it depresses electroencephalographic (EEG) activity, producing sedation, anxiolysis, and anaesthetic hypnosis. Cleton et al.(Cleton, Voskuyl, and Danhof 1998) showed that the concentration–effect relationship for the EEG β-ratio follows a sigmoid E_{max} curve. In the library, the midazolam PK–PD model couples the two-compartment PK model to a sigmoid-E_{max} PD component (Figure 9).



(A)



(B)

Figure 9. (A) PKPD midazolam model extension from Figure 3. (B) Comparison of simulation of the midazolam effect represented as EEG activity to the data reported in (Cleton, Voskuyl, and Danhof 1998).

2.7 Pharmacogenomic models

Pharmacogenomic (PGx) modeling seeks to quantify how inherited genetic variation modulates both drug pharmacokinetics and pharmacodynamics. We propose the following components to integrate genotype–phenotype–specific parameters—such as transporter activity, metabolic capacity, and receptor sensitivity—directly into established PK/PD frameworks. We define six commonly used metabolizer status classes:

```
type MetabolizerStatus =
    enumeration(
      Unknown      "No mapping implemented / uncertain",
      Poor         "Little or no functional activity",
      Intermediate "Reduced activity",
      Normal       "Reference activity",
      Rapid        "Increased activity",
      Ultra        "Greatly increased activity");
```

Next, we define two generic record types Genotype and Phenotype to encapsulate an individual's genotype (allelic composition), corresponding phenotype, and simplified functional modifications of PK/PD effects:

```
partial record Genotype
  parameter String allele[2] = {"*1","*1"};
  replaceable parameter PGx.Phenotype ph
      constrainedby PGx.Phenotype
      annotation( ...);
end Genotype;
```

```
record Phenotype
  parameter String[:] poor = fill("", 0);
  parameter String[:] intermediate = fill("", 0);
  parameter String[:] rapid = fill("", 0);
  parameter String[:] ultra = fill("", 0);
  parameter Real[6] CLscale = {1.0,0.7,0.85,1.0,1.2,1.3};
  parameter Real[6] Fscale  = {1,1,1,1,1,1};
end Phenotype;
```

Additional helper functions are defined as follows:

`phStatus(ph, allele1, allele2)` Maps a pair of star-alleles (`allele1`, `allele2`) and a `PhenotypeTemplate` instance (`ph`) to a `MetabolizerStatus`, using the allele lists specified in `ph`.

`phScaleCL(ph, allele1, allele2)` Returns the clearance-scaling factor corresponding to the combined genotype (`allele1`/`allele2`) and phenotype instance `ph`.

Based on these record templates, a modeler can define sample genotypes and phenotypes that may affect, for example, the renal clearance of a drug. For instance, the SLC22A2 gene encodes organic cation transporter 2 (OCT2), a key renal transporter responsible for the uptake and clearance of many endogenous substances and drugs from the blood into renal tubular cells, thereby influencing their urinary excretion (Koepsell 2013).

Sample implementation of SLC22A2 genotype holds default normal alleles denoted as $*1*1$ which can be changed later in runtime as a common parameter in a model:

```
record SLC22A2Genotype
  extends PGx.Genotype(
   allele = {"*1","*1"},
   redeclare parameter Phenotypes.OCT2_Phenotype ph );
end SLC22A2Genotype;
```

Phenotype OCT2 defines an intermediate metabolizer with at least one allele denoted as 808T and according altered clearance by factor 0.85:

```
record OCT2_Phenotype
  extends PGx.Phenotype(
    intermediate = {"808T"},
    CLscale = {1.0, 0.7, 0.85, 1.0, 1.2, 1.3},
    Fscale = {1, 1, 1, 1, 1, 1}
  );
end OCT2_Phenotype;
```

In order to integrate genotype–phenotype influences into the gentamicin PK model, we must adjust the clearance parameter—either by directly invoking the function phScaleCL() or by delegating the adjustment to a specialized ClearanceModifier model. An example of the gentamicin PK model, implemented in equations and augmented with PGx influence via ClearanceModifier, is shown in the following listing:

```
model PKPG_1C_GentamicineEquation
extends Modelica.Icons.Example;
  Pharmacolibrary.Types.Mass M(start = 0.000400);
  Pharmacolibrary.Types.MassConcentration C(displayUnit = "mg/l");
  parameter Pharmacolibrary.Types.Volume Vd = 0.0175;
  parameter Pharmacolibrary.Types.VolumeFlowRate Cl_nom = 0.1/(1000*60);
  parameter Boolean usePGx = true "true = apply genotype scaling";
  Pharmacolibrary.Pharmacogenomics.Modifiers.ClearanceModifier clMod(
    base = Cl_nom,
    redeclare type Gtype = Pharmacolibrary.Pharmacogenomics.Genotypes.SLC22A2Genotype,
    redeclare parameter Pharmacolibrary.Pharmacogenomics.Genotypes.SLC22A2Genotype g(
      allele = {"*1", "*1"}/* <- change one entry to "808T" for Intermediate */)
    );
  Types.VolumeFlowRate Cl_eff "effective clearance";
equation
  Cl_eff = if usePGx then clMod.modified else Cl_nom;
  C = M/Vd;
  der(M) = -Cl_eff*C;
  annotation( ...);
end PKPG_1C_GentamicineEquation;
```

PGx simulation of such PK model with normal allele and with variant 808T with altered clearance is seen in the following Figure 10:

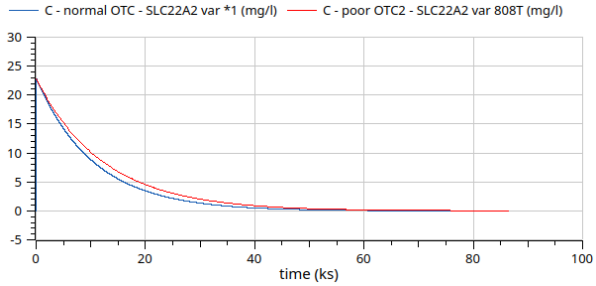


Figure 10. comparison of PGx simulation of PK model of gentamicine with normal SLC22A2 alleles compared to allele '808T' with altered clearance.

To support PGx modeling using diagram view, we define the components in Table 6.

With these components, we can extend PK models with additional PGx influence by applying clearance modifications appropriately and cascading the influence of multiple genes—e.g., CYP2C19 and SLC22A2—as illustrated in the model diagram in Figure 11.

	**ClearanceModifier** q_{vin} is baseline volume flowrate value, u is modification factor from an outside component, e.g. from genotype-phenotype component, q_{vout} is modified output volume flowrate value by equation: $$q_{vout} = u \times q_{vin}$$
	PhenotypeGenotype component joins genotype with it's phenotype and outputs the scale factor that can be used to modify other quantities $$\text{scale} = \text{phScaleCL}(p_h, a_1, a_2)$$ Where p_h is phenotype, a_1 is first allele and a_2 second allele.

Table 6. Components for diagram view to support pharmacogenomic modeling.

DOI 10.3384/ecp21861

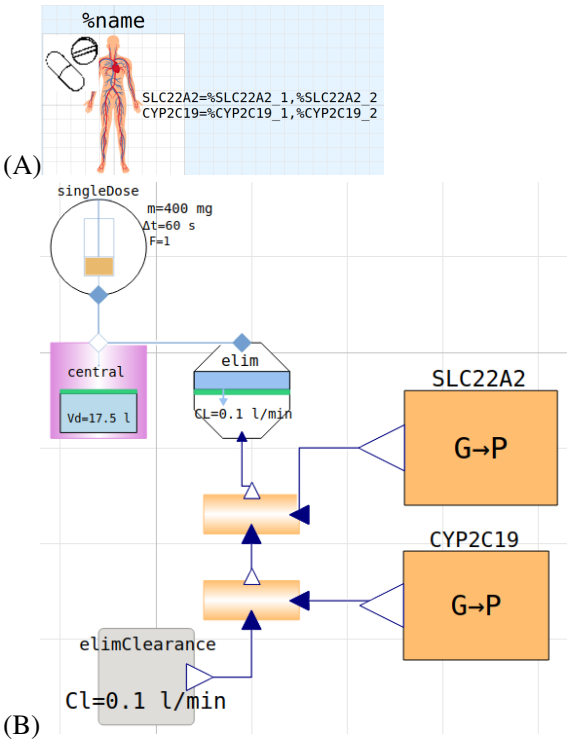


Figure 11. PKPGx model icon (A) and diagram (B) of 1-compartment PK of gentamicine where elimination clearance rate is influenced by PGx of CYP2C19 genotype and SLC22A2 genotype.

Theoretical influence of all combination of the variants of gene mutation of SLC22A2 and CYP2C19 on periodical administration of gentamicine is visible in model and simulation of 4 variants in Figure 12.

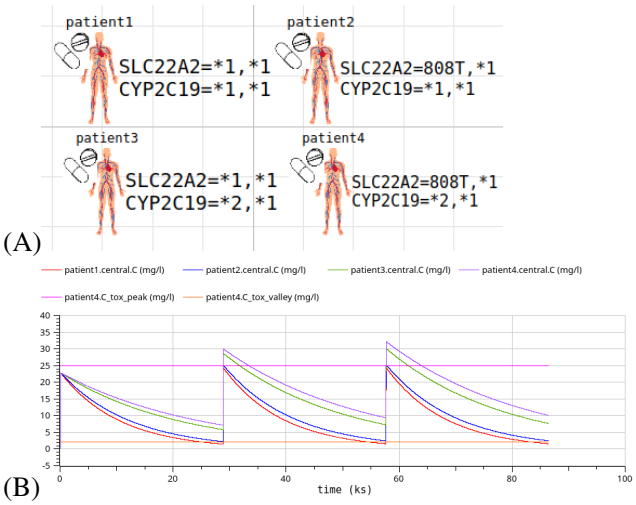


Figure 12. PGx of 4 variants of patients with various gene mutation of SLC22A2 and CYP2C19. Model diagram (A) showing the instance of PKPGx model with specific SLC22A2 alleles and CYP2C19 alleles (patient1 has normal alleles, patient2 with mutation 808T, patient3 with mutation *2 and patient4 with both mutations 8080T and *2). Simulation (B) of PK of gentamicine on all these patients during 3 days, gentamicin is administered every day. Toxicity peak and trough (valey) levels are shown too.

Note that while standard dosing in a normal patient keeps drug concentrations below toxic peak and trough thresholds, rare variants in the cited genes slow elimination; in patient3 and patient4 this raises levels above the toxicity limit, markedly increasing the risk of side effects.

The models in Figures 11 and 12 are illustrative only; the real impact of pharmacogene mutations on drug pharmacokinetics and pharmacodynamics is likely more complex and lies beyond the scope of this paper.

2.8 Integration to simulation workflows

Models exported as FMUs can be instantiated and customized via runtime parameters without requiring recompilation. The following figure illustrates a demo workflow that integrates FMU export into a Python environment using the FMPy library, retrieves experimental data from an external PK database, overlays the data on the simulation chart, and supports interactive parameter updates.

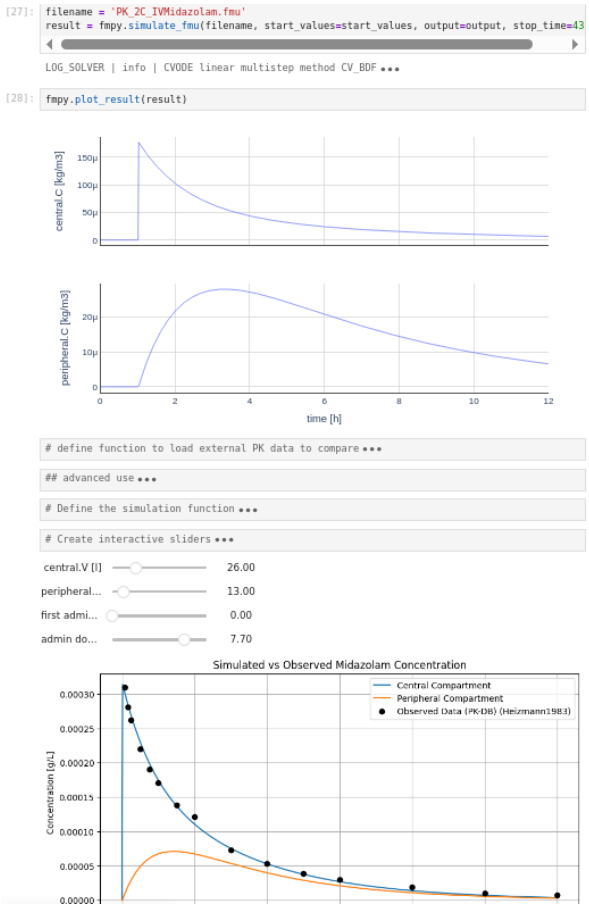


Figure 13. 2-compartment PK model of Midazolam exported to FMU and instantiated in Python using fmpy library. Jupyter notebook is used to visualize interactively model simulation and data of specific drug concentration time series retrieved from PK-DB (Grzegorzewski et al. 2020).

2.9 Integration to other Modelica libraries

Pharmacolibrary can be integrated with other Modelica libraries. The acausal connectors are unique for Phar-

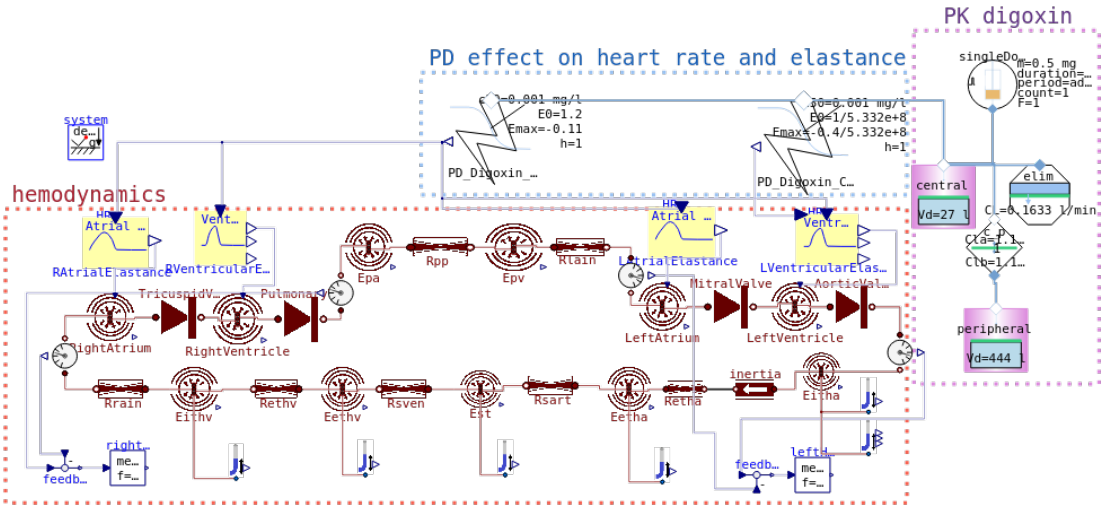


Figure 14. Example integration of PKPD model with Physiolibrary 3.0 (Mateják 2023) 1. hemodynamics of cardiocascular system as published by (Van Meurs 2011), 2. example of 2-compartment PK model of digoxin and PD model of influence of variable elastance during cardiac cycle as published by (Hornestam et al. 2003). PD effect on elastance/compliance just illustrative.

macolibrary, however causal connectors that holds only one variable can be directly connected to any appropriate component. Following the common convention in Modelica libraries, connector nominal values use SI units: compliance is expressed in $m^3\,Pa^{-1}$ (converted from $mL\,mmHg^{-1}$), and heart rate in s^{-1} (Hz) rather than beats min^{-1}. Digoxin is a drug used to treat e.g. acute atrial fibrilation as it has combined effect on inotropy (ventricular elastance) and heart rate. Example of PK PD integration of digoxin to model of hemodynamics is visible in Fig.14 with PD effect parameters in basic SI units and illustrative simulation is at Fig. 15.

3 Discussion

Implementing multicompartmental pharmacokinetic/pharmacodynamic models can be cumbersome and error-prone when using generic programming language, or modeling language and methods using equation-based or block-diagram approaches. While simple one- and two-compartment models (Fig. 1 A, B) can be implemented either in diagram mode or in equivalent equation text mode, more complex architectures quickly become unwieldy in text mode (e.g., Fig. 6, and 7). In contrast, Modelica tools automatically generate consistent textual representations from acausal diagram views—translating connected variables and component definitions into correct equations. Combination of acausal and stream connectors (connectors exchanging flow, non-flow and stream variables) ensure that non-flow (equality) and flow variables equilibrate appropriately, while stream connectors enforce mass balance by mixing based on flow direction. This component-oriented approach preserves the precision of underlying equations while retaining an intuitive, schema-like structure plausible for non-programmers.

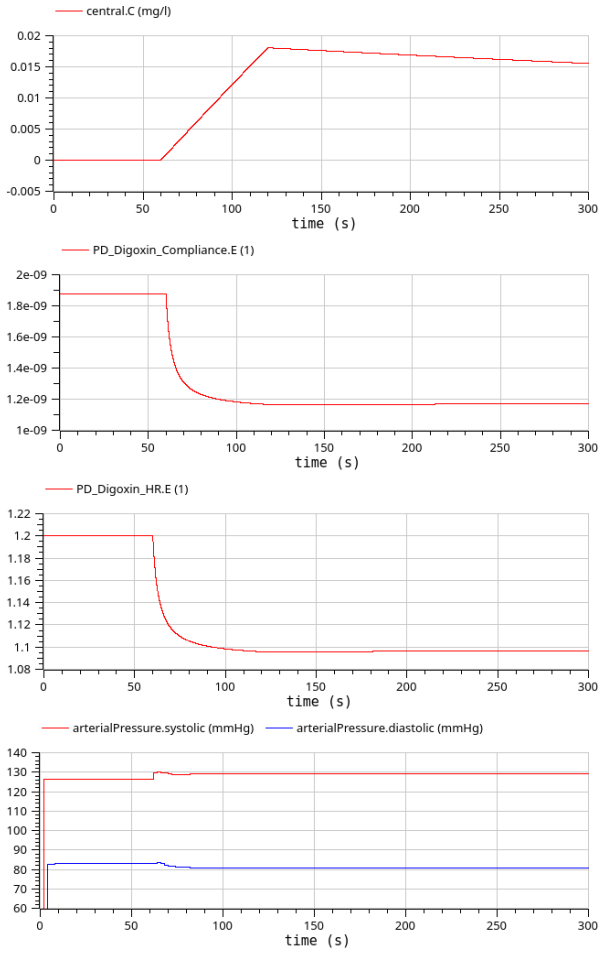


Figure 15. Simulation of PK, PD effect on compliance and heart rate and systolic/diastolic pressure in 5 minutes before, during and after administration of intravenously applied digoxin. Digoxin is administered after 60 seconds. Note that while systolic pressure slightly increases the diastolic pressure slightly decreases when the drug is administered.

DOI 10.3384/ecp21861

Pharmacology modeling must also accommodate extensive genetic and physiological variability. Embedding all genotype- and phenotype-specific variants directly in static diagrams can become impractical as well as to model PK of all drugs using drug-specific models. To address this, our library's core components are designed for post-compilation parameterization: users can adjust pharmacokinetic, pharmacodynamic and genotype-phenotype influences (e.g., via ClearanceModifier or phScaleCL()) at runtime without recompiling it.

Pharmacogenomics is rapidly maturing: e.g. the Dutch Pharmacogenetics Working Group and the Clinical Pharmacogenetics Implementation Consortium now recommend genotype-based dose adjustments for many drugs. Embedding these recommendations into physiologically based models enables truly personalized simulations that account for both genetic polymorphisms and physiological differences. Current Pharmacolibrary allows only basic components for PGx modeling which may be a foundation for systematic approach in this context interconecting phenotypes and it's effect on different tissues during drug lifecycle in organism.

Pharmacolibrary contains also basic PK models of all approved drugs that was generated by a workflow extracting knowledge from the literature—or infers it from structurally similar compounds—using recent advances in AI and large language models (LLMs). These PK models are stored in the "Drugs" package and categorized by Anatomical Therapeutic Chemical (ATC) codes[1]. Each model contains essential drug information and references generic one-, two-, or three-compartment PK model with specific instance parameter. Such generated models needs manual verification and refinement, however, may be useful as starting point for drug modeling. If a human modeler later derives additional parameters or adds another route, that supplementary model is saved alongside the generated one.

Drug effects often exhibit time lags or hysteresis between tissue concentrations and observable responses. Standard Modelica libraries support modeling such delays (e.g., via transport or delay blocks). Additionally, pharmacodynamic outcomes can be modulated by the body's regulatory feedback. By connecting PKPD models to broader physiological frameworks (for example, Modelica's Physiolibrary or Chemicals library), one can simulate homeostatic mechanisms and regulatory dynamics alongside drug action as it is illustrated in one example in Fig.14 and Fig.15.

As models can be exported to FMUs following the FMI standard and integrated into diverse simulation workflows (Python, Julia, MATLAB, or bespoke environments) the interoperability can be made with various other systems biology libraries, frameworks and workflows. One such example is illustrated integrating model in Jupyter notebook with Python code and data from PK-DB in Fig.13.

Because Pharmacolibrary is still in its infancy, a thorough comparison with commercial or open-source alternatives would be premature. Even so, within the broader Modelica ecosystem, the library already fills a crucial niche for modeling biological systems with an emphasis on chemical compounds and drugs.

3.1 Library limitations

To keep models transparent and user-friendly, we adopted pharmacology-centric units and terminology—such as volume flow rate (SI base unit m^3/s, displayed as l/min) via FlowPort connectors, even though under variant temperatures or pressures (e.g., during gas exchange or hypothermia) mass flow rate (kg/s) would be more accurate. A similar point applies to the use of concentration as a quantity: at membranes, it is actually the chemical potentials—not the concentrations—that equilibrate. However, modeling this explicitly would introduce unnecessary complexity, whereas our goal is to keep the model intuitive, and user-friendly.

4 Conclusion

In this work, we have introduced *Pharmacolibrary*, a flexible Modelica-based library for mechanistic pharmacological modeling that spans pharmacokinetics, pharmacodynamics, toxicokinetics/toxicodynamics, physiology based PK/PD and pharmacogenomics. By providing standardized acausal connectors, reusable compartment and effect components, and genotype–phenotype records, our library simplifies model construction, ensures mass and concentration balance. The models are Exportable as FMUs in accordance with the FMI standard, Pharmacolibrary models can be embedded into diverse computational workflows (e.g., Python, Julia, R,...), enabling interactive parameterization, data-driven validation, and high-throughput simulations.

The library is open-source maintained at GITHUB [2] and listed in ZENODO (Kulhánek and Ježek 2025)

Implementations of gentamicin, midazolam, fentanyl, and digoxin—including genotype/phenotype-adjusted clearance—demonstrate that Pharmacolibrary can replicate experimental data and underpin personalised dosing. Moreover, the library contains simplified generated PK models of all drugs classified by anatomical therapeutical chemical (ATC) codes. As of 06/2025 there were 5496 unique records at ATC level 5, corresponding to 4363 distinct chemical substances.

By separating model definition from the solver and building on the Modelica ecosystem (Physiolibrary, Chemicals, Biochem), it promotes reproducibility, interoperability, and extensibility. Ongoing work will add transporter and enzyme-kinetic blocks, compile and improve a catalogue of simplified drug models, incorporate

[1] https://www.who.int/tools/atc-ddd-toolkit/atc-classification

[2] https://github.com/creative-connections/Pharmacolibrary

physiological feedback, and validate clinical case studies to further advance quantitative systems pharmacology.

Conflict of Interest

No conflicts are to be reported.

Acknowledgements

This library was inspired by the PharmaKin Modelica library (MediWare a.s., CZ, co-authored by Jan Šilar) as a part of a joint research grant on computational modeling for optimized pharmacotherapy (Potůček et al. 2016). F.J. has been partially supported by NIH; subaward to R01 HL173346. T.K. was supported by the ONCOSCREEN project, funded by the European Union's Horizon Europe research and innovation programme under grant agreement No. 101097036.

References

Azer, Karim et al. (2021). "History and future perspectives on the discipline of quantitative systems pharmacology modeling and its applications". In: *Frontiers in physiology* 12, p. 637999.

Cleton, A, R A Voskuyl, and M Danhof (1998-10). "Adaptive changes in the pharmacodynamics of midazolam in different experimental models of epilepsy: kindling, cortical stimulation and genetic absence epilepsy: Epilepsy induced reduction in midazolam efficacy". en. In: *Br. J. Pharmacol.* 125.4, pp. 615–620. ISSN: 0007-1188,1476-5381. DOI: 10.1038/sj.bjp.0702088. URL: http://dx.doi.org/10.1038/sj.bjp.0702088.

Fleishaker, J C and R B Smith (1987-12). "Compartmental model analysis in pharmacokinetics". en. In: *J. Clin. Pharmacol.* 27.12, pp. 922–926. ISSN: 0091-2700,1552-4604. DOI: 10.1002/j.1552-4604.1987.tb05591.x. URL: http://dx.doi.org/10.1002/j.1552-4604.1987.tb05591.x.

Fritzson, Peter et al. (2020). "The OpenModelica Integrated Environment for Modeling, Simulation, and Model-Based Development". In: *Modeling, Identification and Control* 41.4, pp. 241–295. DOI: 10.4173/mic.2020.4.1.

Grzegorzewski, Jan et al. (2020-11). "PK-DB: pharmacokinetics database for individualized and stratified computational modeling". In: *Nucleic Acids Research* 49.D1, pp. D1358–D1364. ISSN: 0305-1048. DOI: 10.1093/nar/gkaa990. eprint: https://academic.oup.com/nar/article-pdf/49/D1/D1358/35364573/gkaa990.pdf. URL: https://doi.org/10.1093/nar/gkaa990.

Heizmann, P, M Eckert, and W H Ziegler (1983). "Pharmacokinetics and bioavailability of midazolam in man". en. In: *Br. J. Clin. Pharmacol.* 16 Suppl 1, 43S–49S. ISSN: 0306-5251,1365-2125. DOI: 10.1111/j.1365-2125.1983.tb02270.x. URL: http://dx.doi.org/10.1111/j.1365-2125.1983.tb02270.x.

Hodiamont, Caspar J et al. (2022-08). "Clinical pharmacokinetics of gentamicin in various patient populations and consequences for optimal dosing for Gram-negative infections: An updated review". en. In: *Clin. Pharmacokinet.* 61.8, pp. 1075–1094. ISSN: 0312-5963,1179-1926. DOI: 10.1007/s40262-022-01143-0. URL: http://dx.doi.org/10.1007/s40262-022-01143-0.

Hornestam, Björn et al. (2003). "Intravenously administered digoxin in patients with acute atrial fibrillation: a population pharmacokinetic/pharmacodynamic analysis based on the Digitalis in Acute Atrial Fibrillation trial". In: *European journal of clinical pharmacology* 58.11, pp. 747–755.

Hosseini, Iraj et al. (2018). "gPKPDSim: a SimBiology®-based GUI application for PKPD modeling in drug development". In: *Journal of pharmacokinetics and pharmacodynamics* 45, pp. 259–275.

Jamei, Masoud et al. (2009). "The Simcyp® population-based ADME simulator". In: *Expert opinion on drug metabolism & toxicology* 5.2, pp. 211–223.

Kaneda, Kotaro and Tae-Hyung Han (2009-09). "Comparative population pharmacokinetics of fentanyl using non-linear mixed effect modeling: burns vs. non-burns". en. In: *Burns* 35.6, pp. 790–797. ISSN: 0305-4179,1879-1409. DOI: 10.1016/j.burns.2008.12.006. URL: http://dx.doi.org/10.1016/j.burns.2008.12.006.

Koepsell, Hermann (2013). "The SLC22 family with transporters of organic cations, anions and zwitterions". In: *Molecular Aspects of Medicine* 34.2. The ABCs of membrane transporters in health and disease (SLC series), pp. 413–435. ISSN: 0098-2997. DOI: https://doi.org/10.1016/j.mam.2012.10.010. URL: https://www.sciencedirect.com/science/article/pii/S0098299712001240.

Kulhánek, Tomáš and Filip Ježek (2025-07). *Pharmacolibrary: Modelica library for pharmacology*. Version v25.08-beta2. DOI: 10.5281/zenodo.15322052. URL: https://doi.org/10.5281/zenodo.15322052.

Lippert, Jörg et al. (2019). "Open systems pharmacology community—an open access, open source, open science approach to modeling and simulation in pharmaceutical sciences". In: *CPT: pharmacometrics & systems pharmacology* 8.12, p. 878.

Matejak, Marek et al. (2015). "Free modelica library for chemical and electrochemical processes". In: *11th International Modelica Conference, Versailles, France*. Vol. 118, pp. 359–66.

Mateják, Marek (2023). "Creating cardiovascular and respiratory models using Physiolibrary 3.0". In: *Modelica Conferences*, pp. 463–468.

Mateják, Marek et al. (2014). "Physiolibrary - Modelica Library for Physiology". In: *Proceedings of the 10th International Modelica Conference, March 10-12, 2014, Lund, Sweden*. Linköping University Electronic Press, pp. 499–505. DOI: 10.3384/ecp14096499.

Nilsson, Emma Larsdotter and Peter Fritzson (2005). "A metabolic specialization of a general purpose modelica library for biological and biochemical systems". In: *4th International Modelica Conference*. Citeseer.

Potůček, J et al. (2016). "Use of computer models for optimized pharmacotherapy (in Czech)". In: *MEDSOFT* 28, pp. 120–129. URL: https://www.medsoft.website/sbornik/2016/Medsoft_2016_Potucek.pdf.

Rosenbaum, Sara E (2016). *Basic pharmacokinetics and pharmacodynamics: An integrated textbook and computer simulations*. John Wiley & Sons.

Schön, Kristyna et al. (2022). "Comparison of Mw\Pharm 3.30 and Mw\Pharm++, a Windows version of pharmacokinetic software for PK/PD monitoring of vancomycin. Part 1: A-posteriori modelling". In: *Computer Methods and Programs in Biomedicine* 214, p. 106552.

Van Meurs, Willem (2011). *Modeling and simulation in biomedical engineering: applications in cardiorespiratory physiology*. McGraw-Hill Education.

Combining static and dynamic optimization approaches for path planning, with collision avoidance

Dr. Clément Coïc[1] Marco Masannek[1,2]

[1]Siemens Healthineers AG, Germany, {clement.coic, marco.masannek}@siemens-healthineers.com
[2]Lab for mobile Robotics, Nuremberg Institute of Technology, Germany, {marco.masannek}@th-nuernberg.de

Abstract

In response to staff shortages in hospitals, healthcare providers aim at increasingly automating their systems. Defining automated system paths – with collisions avoidance – is a critical step towards automation. In this paper, three different approaches for path planning are investigated: a static, a dynamic and a hybrid approach. The hybrid approach, that sequentially combines part of the static and dynamic approaches, results in improved accuracy against the static approach and improved performance against the dynamic approach.
Keywords: Path planning, C-space, Dynamic Optimization, Hybrid approach



Figure 1. Project AURORA: Target operation room environment (left) and real robot (right)

1 Introduction

The staff shortage in the healthcare industry is a bottleneck when it comes to patient touchpoints. Healthcare system automation can partially solve this issue. Should automated systems reduce the workload of qualified healthcare professionals, these would be available for interventions - where they can add value, save life, care for the patients or plan procedures, to name but a few. There are different tasks currently performed by qualified by healthcare professionals that could be taken over by automation. First, existing systems could perform in a more automated manner. For the sake of simplicity, this paper focuses on robotic systems evolving freely on the hospital floor - though the entire discussion could be extended to any type of system, e.g. ceiling systems. A typical example could be to provide qualified voice command to, for example, a CIARTIC Move (Siemens Healthineers AG 2025) - such as "Take a 3D scan of the patient right foot." - and the Ciartic Move would autonomously evolve in the room until the foot is at the isocenter of the scan, would ask for permission to scan and proceed. In a second step, new robotic systems could be developed to perform actions that are not part of today's offering. For example, the research project Autonomous Robotic Operation Room Assistance (AURORA) (MITI Research Group 2024) aims to assist in operations by performing tasks required by circulators, which focus on actions in the periphery of the operating table, e.g. equipment preparation (see Figure 1).

In both above examples, one key capability the system shall have is determining the path it shall take to safely evolve in its environment. Additionally, the system shall sense its moving environment and adapt the path based on the potential new or moving obstacles along its path. This paper focuses on solving that path planning problem with fixed obstacles. This can be justified by the fact that the authors already mitigated part of the risk associated with moving obstacles by presenting a novel approach relying on a semantic understanding of the environment, and path adaptation can be done with reinitiating the path planning under the new constraints. The remaining of the paper will present three different approaches for solving the path planning problem. In Section 2, a static approach is discussed. This method relies on discretizing the environment and determining the C-space of the robotic system to define the feasible paths. Section 3 tackles the problem from the perspective of the dynamic motion of the robotic system: what is the optimum path under the constraints of the robotic system kinematic. Section 4 discusses a hybrid approach consisting of sequentially applying part of the static and dynamic approaches. Finally, the paper is concluded and perspective to the work are presented.

2 Static Path Planning

In most mobile robotic applications, the problem of finding a collision-free path is approached by first creating a map of the environment using sensors and Simultaneous Localization and Mapping (SLAM) algorithms (Placed et al. 2022). In a second step, the map is then converted into a discretized occupancy grid map, also commonly referred to as cost map, which assigns specific cost values for each

cell depending on the proximity of obstacles. By utilizing these cost values as a basis for node evaluation during graph search algorithms such as A* or RRT, current robotic systems achieve computation times of often less than 100ms even for longer paths.

However, this approach has two major limitations affecting the quality of generated paths, which are the missing consideration of vehicle dynamics and the often-inaccurate representation of valid configurations for non-circular robots, e.g. for rectangular shapes. To overcome the latter, we introduce a method for fast and memory-efficient computation of the collision-free configuration space (C-space) for any shape robots.

2.1 C-Space for non-circular robots

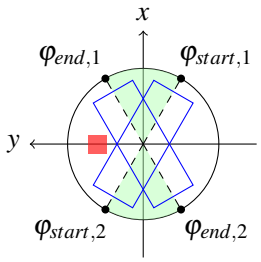
To determine valid poses within a given environment, most approaches start by creating a Euclidian Signed Distance Field (ESDF) which describes the distance to the nearest obstacle for each x-y-cell in the cost map. For circular robots, these values can already be utilized to perform complete collision evaluation by comparing the distance with their own radius. Cells below this threshold are usually marked as colliding, while distances above the threshold mark free and therefore traversable space. Since the orientation (heading) of a circular robot has no effect on its radius and therefore minimal distance threshold, the classification holds true for all headings φ.

However, rectangular or any shape robots need to consider their actual occupied space for each discrete heading φ_h at a given position $p(x,y)$ to reason about whether or not the robot would collide at a specific pose $P(x,y,\varphi)$. The angular resolution r_φ that is required to ensure valid collision checks in the discretized grid can be determined using the cell resolution r_c and the circumscribing radius of the footprint d_{cr} (furthest point of the shape) and often lies in a range from 1° to 10°. Since this creates a necessity for a three-dimensional grid (third dimension being the heading), generated c-spaces can grow very large for big environments with high resolution requirements.

To overcome this challenge, (Lau, Sprunk, and Burgard 2013) introduced the concept of valid angle intervals (see Figure 2), which describe a set $\mathbf{v_I}$ of collision-free intervals of heading for an arbitrary shaped robot. This method allows for efficient computation of collision checks for any pose $P(x,y,\varphi)$ by simply evaluating if the angle φ intersects with any interval of $\mathbf{v_I}$. Cells with a single interval spanning the full angular range can additionally be marked as completely free to save further memory.

2.2 Example C-Space for a hospital floor

Figure 3 shows an exemplary hospital floor map with hallways, patient rooms, treatment rooms and bathrooms. In order to plan paths for a rectangular robot (e.g. a patient bed or the CIARTIC Move) between any two positions, the complete floor needs to be processed. Brute force calculations of the C-space would result in over 180 million cells when given a map of 1280x720 pixels and a required



$$\mathbf{v_I}(\mathbf{x}|\mathbf{y}) = \begin{bmatrix} (60°, 120°) \\ (240°, 300°) \end{bmatrix}$$

Figure 2. Example angle intervals for a rectangular robot (blue) for a close obstacle (red square)

angular resolution of 1.8° (equals 200 angles per rotation). Utilizing the method described in section 2.1, we are able to reduce the size of the c-space map by over 80% to roughly 35 million cells.

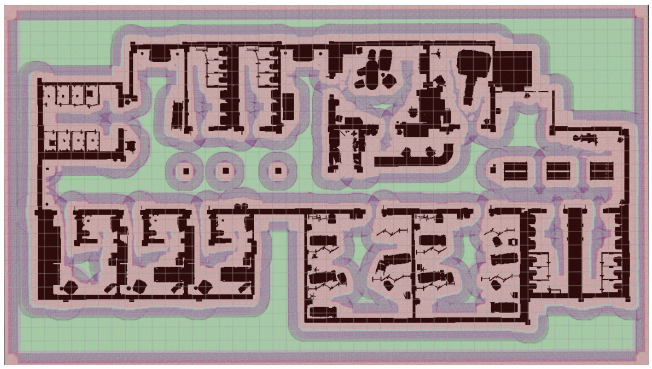


Figure 3. C-Space for a hospital floor and rectangular robot. Black cells mark physical obstacles such as walls, beds or other objects. Green cells are completely free, red cells cause collisions. Purple cells mark cells with valid angle intervals.

2.3 Collision graph from angle intervals

In order to plan a path through the occupancy grid including its angle intervals, a customized graph structure needs to be created that accounts for the presence of multiple angle intervals on the same position $p(x,y)$. We therefore create an adapted neighborhood discovery step during expansion in our graph search (utilizing A*) which discovers all valid intervals of a neighboring cell. Connections are established only when intervals overlap to ensure that a continuous path can be created in the end. Since completely free cells have overlaps with all intervals, connections between them can always be established (see Figure 4).

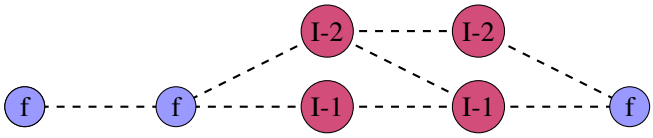


Figure 4. Example edges between free and and orientation interval nodes

DOI
10.3384/ecp21873

2.4 Limitations of grid-based planning

Figure 5 shows a typical path planning problem in hospitals, where a patient bed must dock to a magnetic resonance imaging system in a narrow room.

While our static path planning is able to produce collision-free paths for non-circular robots even through the narrow sections (see Figure 5), the generated trajectories are not smooth and also imply following using an omni-directional movement model (e.g. shopping cart).

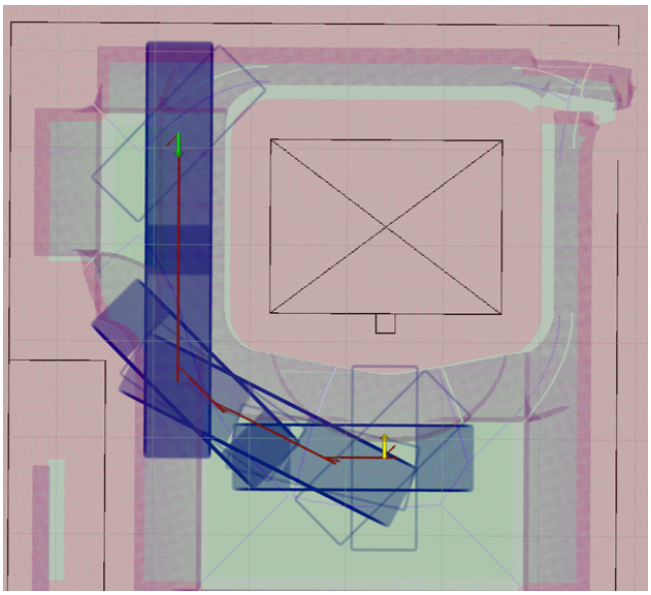


Figure 5. Planned path using our custom collision graph. The arrows mark the start (green) and goal (yellow) pose for the robot; the blue phantoms depict the footprint along the path.

3 Dynamic Optimization

The static approach to path planning proved to be very efficient in finding a solution. However, it does not guarantee that a robotic system, with constrains on its kinematic motion, could actually achieve the found path. This might not be an issue for some robotic systems - holonomic, such as the CIARTIC Move, which drive system allows for any type of motion on the 2D plan (i.e. rotation on the spot, translation in all directions (even lateral) and combination of both motions). However, many other types of drive systems are non-holonomic - i.e. they have constrained kinematics that should be accounted for when planning for their path.

Dynamic optimization (Åkesson 2008) is leveraged here to solved for the optimum path of a behavioral model of the robotic system, under its environment constraints. Four steps have been followed to implement this solution. First, a simplified behavioral model of the robotic system is developed. Then, a simplified model of the environment is introduced together with the collision assessment with the robot. In a third step, the dynamic optimization function is implemented. Finally, the resulting path is run in a forward simulation of a more detailed behavioral model. These four steps are detailed below, after a quick discussion on the need for these two fidelities of behavioral models.

3.1 Behavioral Model Purposes

Models are developed with a purpose in mind. Here, our aim is to easily assess the behavior of different drivetrain set ups for our robotic systems and their impact on the trajectory they can achieve. We collected the two different set of requirements associated with this purpose.

1. Path planning: the behavioral model shall represent the system dynamics, while being as simple as possible, to allow for a fast optimization. As directional derivatives are leveraged by the dynamic optimization solver, it is of high importance that the behavioral model is C2-continuous, i.e. continuous, derivable and the derivative is continuous too.

2. Behavioral path following: the behavioral model shall allow for modeling the different drivetrain in more details to assess how they perform when following a specified path.

In order to meet both requirements, the choice was made to have two separate models - one for each purpose. The behavioral model dedicated to path planning is a point-heading motion model, with potential constraints on the trajectories it can follow. The behavioral model dedicated to studying the path following response is a multi-body model with detailed kinematics of the drivetrain.

3.2 Point-heading Optimization Model

A point-heading motion model, as the names indicates, consists in modeling the system as a point with a heading, and some constraints on its motion. In addition, the robot envelope around the point is also known so that we can easily check for collision of the real system - and not only the point. At this stage, two different cases are made possible:

1. The robotic system is holonomic, i.e. it can evolve freely in all directions and motion types (like the CIARTIC Move).

2. The robotic system is non-holonomic, i.e. it can only continuously change the path curvature with respect to its reference frame.

The former case does not constrain the path with respect to the robot heading. For example, a path forming a 90° angle with the heading corresponds to a sideway translation. The latter means that the rate of change (derivative) of the path heading cannot be higher than the point steering capability. In other words, the path and point headings shall align. These two cases can easily be modeled by adding a constraint on the path to heading angle that should be 0 in case of non-holonomic robots.

Listing 1. Point-heading Optimization Model

```
model PointHeading
 // Cartesian coordinates
 Modelica.Units.SI.Position x "Global x
     position";
 Modelica.Units.SI.Position y "Global y
     position";
 Modelica.Units.SI.Angle phi(start=phistart,
     fixed=true) "Global heading";
 // Change to local coordinates
 //...
 parameter Boolean isHolonomic = true "false
     , if heading should be fixed =0";
 // Defining robot body
 // ...
equation
 // motion ...
end PointHeading;
```

The resulting Modelica model shown in Listing 1 is very simple and includes some additional variables for changing the reference frame from global to local and for defining the body of the robotic system around the point. It remains C2-continuous and suitable for a fast dynamic optimization application.

3.3 Simplified Environment Model

The environment where the robot evolves is a bounded rectangle. Internally, it is composed of the floor set-up - including walls, defining the rooms and corridors - and all types of objects and persons that could be on the way - e.g. an MRI or CT machine, a bed with patient, or even a plant. From the perspective of the robot, these are all seen as obstacles. Therefore, there is no need to differentiate them in this approach, and it was decided rather to allow for modeling a couple of shapes - specifically rectangles and circles - as footprint on the floor. As discussed in section 2, circles are efficient as there is no need to model their orientation - the Euclidian distance is the same for any angle. Rectangles require their orientation to be modeled. For this purpose, specific classes have been developed - Circle, Rectangle -, that contain their dimensions and take the position of the object in motion to check for collision. As many obstacles as desired can be introduced in the environment and a combination of them can represent more complex shapes. For example, a square room with an open-door access can be represented with 4 rectangle obstacles. (Alternatively, should a scenario happen exclusively within a rectangular room, it is possible and more efficient to define the bounds of the room as constraints to the robot coordinates in the optimization problem.)

3.4 Optimization Problem

Now the point-heading model and its environment are defined, the next step is to look at the path the robot shall follow. Note again that the path and the trajectories of the robot model (x, y, phi) are not necessarily the same: specifically, for non-holonomic robots, the heading of the robot and of the path may not be aligned. Therefore, a heading variable for the path itself, phi_p, is introduced. Additionally, we define a velocity along the path, v_p, its projections on the x and y axes, v_x and v_y, and its time integral gives us the length of the path d_p.

The cost function of our optimization model is a combination of the path length and the rate of change in path curvature. The goal is to minimize the distance while avoiding abrupt changes in curvature. Obviously, this cost function can easily be adapted based on the needs and would lead to different optimum paths.

The inputs to optimize are the heading derivatives of the path and of the bed. Note that because the velocity is a set, these headings fully define the trajectories. (Alternatively, the velocity could be considered an input.) However, the headings derivatives are selected here as these are needed for the cost function (no abrupt changes in heading) and because the headings are easily obtained by further integrations.

To fully define the optimization problem, the following constraints are added:

- The robot remains within the bounds of the environment.
- Potential alignment of headings for non-holonomic robots (see end of subsection 3.1).
- Checking that the distance to obstacle remains positive.
- Ensuring that the initial and final target positions are met.

The optimization problem is written in the Optimica language, allowing us to extend the existing Modelica behavioral and environment models, similarly to (Coïc, Budinger, and Delbecq 2022).

Listing 2. Pseudo Point-heading Optimization Model

```
optimization PointHeading
 objective = d_p(finalTime) + cost(finalTime
     ),
 finalTime(
  free=true,
   min=1,
   max=10,
   start=5)

 // Floor or Rooms bounds
 parameter Modelica.Units.SI.Position xMin
     =0;
 parameter Modelica.Units.SI.Position yMin
     =0;
 parameter Modelica.Units.SI.Position xMax
     =10;
 parameter Modelica.Units.SI.Position yMax
     =10;

 extends PointHeading(
  // modifiers to pass floor bounds as min,
      max
 );
```


DOI
10.3384/ecp21873

```
parameter Integer n_rectangles=5;
Rectangle[n_rectangles] rectangles(
 //modifers
 );


// Start and end point positions-headings
// ...

// Path variables
Modelica.Units.SI.Angle phi_p "Path heading
    ";
Modelica.Units.SI.Length d_p(start=0,fixed=
    true) "Path length";
Modelica.Units.SI.Velocity v_p=1 "Velocity
    along path";
parameter Modelica.Units.SI.Angle max_phi_p
    =if isHolonomic then Modelica.Constants
    .pi else 0;

// Inputs for trajectory optimization
input Modelica.Units.SI.AngularVelocity
    der_phi "Point heading derivative";
input Modelica.Units.SI.AngularVelocity
    der_phi_p "Path heading derivative";

equation
 der(phi)    = der_phi;
 der(phi_p)  = der_phi_p;
 v_p         = der(d_p);
 cost        = d_phi^2 + d_p^2;
 // ...

constraint
 phi_p <=  max_phi_p;
 phi_p >= -max_phi_p;
 // ...
 x(finalTime) = x_end;
 y(finalTime) = y_end;
 phi(finalTime) = phi_end;
end PointHeading;
```

Figure 6 is an example of a rectangular holonomic robot (in grey) entering a challenging room. The blue rectangles represent walls with small tolerances. The room dimensions are above the minimum recommendations, yet the door is placed with an angle, which limits the robot in its movements. This simulation proved that the non-holonomic robot was not able to find a path in this challenging environment (only by few centimeters), yet the holonomic could manage.

The path planning was implemented as web application within Modelon Impact - as in (Coïc, Andreasson, et al. 2020) - to easily generate scenarios. The resulting paths are stored as csv artifact that can easily be imported into the workspace. This results very handy as the next step is to validate that our point-heading model simplification is not too simplistic. This is verified by simulating a higher fidelity (multibody) path following robotic system.

3.5 Multibody Drive Model

For the purpose of accurately modeling the kinematics of the robotic system, it is enough to model its drive system,

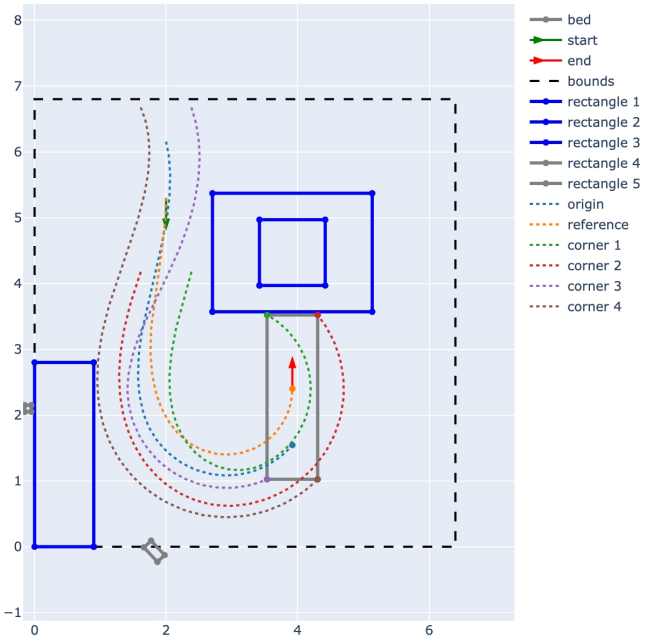


Figure 6. Optimum path planning example

and the rest as point-mass located in the center of gravity - as shown in Figure 7. Each wheel module - see Figure 8 - is a submodel that allows for representing different types of wheels: steered, driven, free, with a caster trail, etc. The tyre model is based on (Pacejka 2012). To each wheel is connected a controller that computes their commands in steering and driving - should these be activated. This makes a very flexible model that allows for investigating a variety of different robotic drive system.

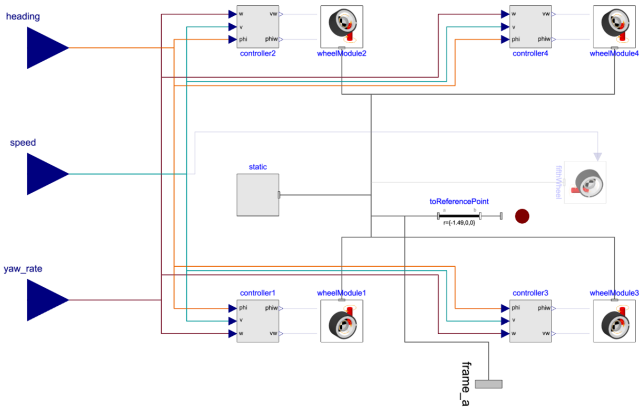


Figure 7. Multibody Model of the Robotic System Drive

As an example of holonomic robot, the four lateral wheels can be active with two caster wheels and two wheels that are both steered and driven. A robot with a fifth wheel in the middle that is locked in steering and driven while one of the other four wheels would be steered only would lead to a non-holonomic configuration - as the robot could only turn in forward motion and around the fifth wheel point of contact.

Resimulating the optimum path from Figure 6 - of the

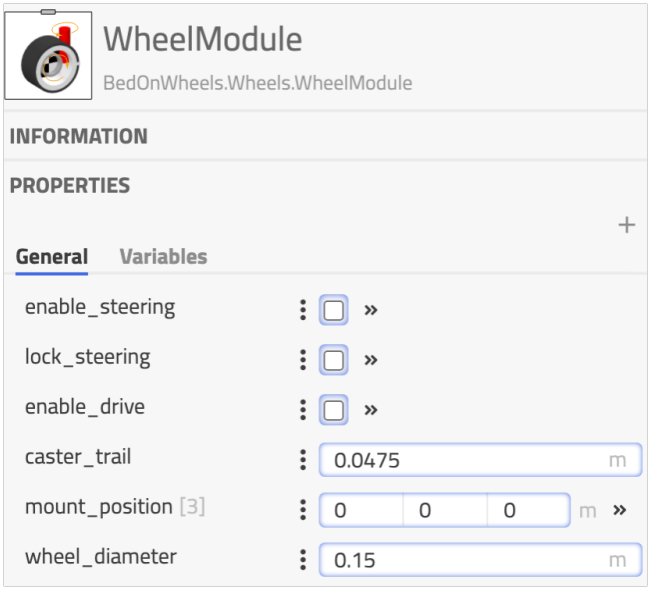


Figure 8. Fully configurable wheel module

reference point, in orange - with a holonomic drive robot, it is observed that the followed path is nearly identical to the optimum one (see Figure 9). And the small divergence could potentially be due to the path following controller. For this case, it is possible to conclude that the point-heading model is a reasonable simplification for the optimization problem.

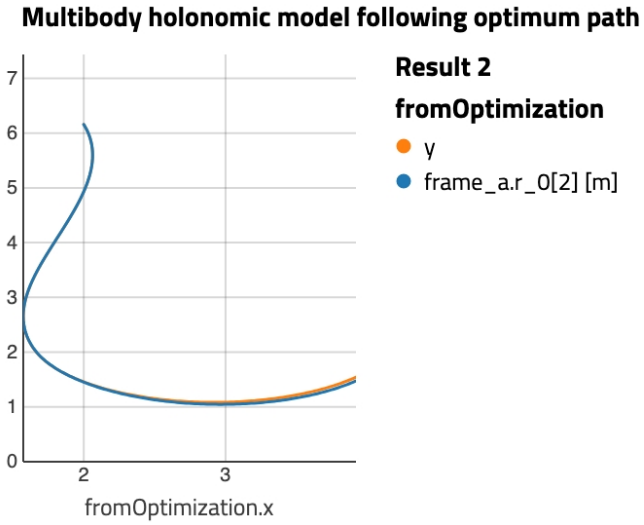


Figure 9. Multibody drive model following optimum path

4 Hybrid Approach

The dynamic optimization solution, combined with multibody path following verification, yields good results. Nevertheless, the static approach proved to be much faster - thanks to the pre-computation of the C-Space. Naturally, the next step in this research is thus to combine both approaches, to get an improved speed with respect to the dynamic optimization solution while conserving its accuracy.

The benefit of defining the environment in a full fledge programming language are multiple. Especially, the environment can be much more complex and programmatically scripted from existing hospital plans. And the C-Space can be easily precomputed with the desired discretization. However, such a discretization would introduce discontinuities, which are not optimization-friendly.

Here, the decision was taken to include the C-Space as part of the cost function (see Figure 10), instead of having it as constraints (as previously in the dynamic optimization implementation). The C-Space is imported into the Optimica model as a table with the three coordinates as inputs - (x, y, phi) - and output an associated cost. The cost is extremely high if the input result in a collision, moderately high if the collision is near and zero if the desired margin is ensured. This cost decrease is made continuous and the table is configured to have a smooth interpolation - making the C-Space optimization friendly.

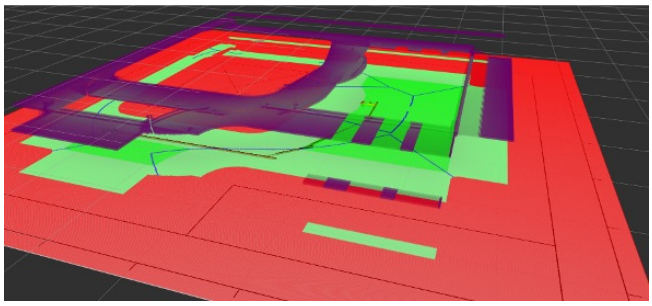


Figure 10. C-Space as a cost "heat map"

This solution also allows for an easy enhancement of the C-Space - e.g. leverage semantic knowledge of the environment to keep a larger distance from people than from fixed objects.

Figure 11 shows two results of the same scenario, with different C-Space costs. The yellow path ensures the desired margin on the persons and finds an optimum that reduces the total length of the path. The green path includes additional costs for the persons and thus leads to a longer path as optimum, with increase safety for the personal. (It is worth noting that a different cost has been applied to the chair - represented in blue -, and thus the green path is nearer to the chair than to the personal.)

As expected, dynamic optimization converges much faster with a good initial trajectory (Delbecq et al. 2021). In this case, it is trivial to provide the one found by running the static approach - which might not satisfy all motion constraints and yet ensure collision-avoidance.

While the initial development of this approach was not trivial, the results are very satisfying in terms both of speed and accuracy. The automation of the environment import, via the C-Space cost table, allows for easy scenario generation and thus increases the representativeness of this solution.

DOI
10.3384/ecp21873

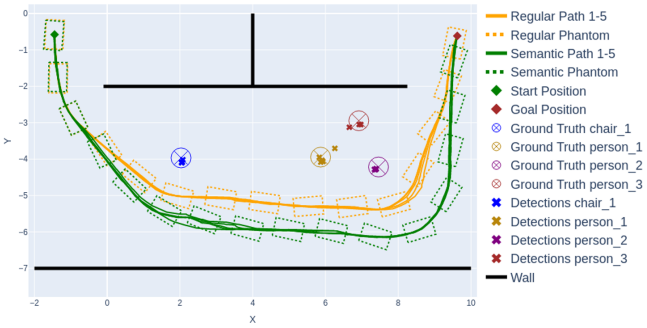


Figure 11. Semantic path planning / social path planning

5 Conclusion and Perspectives

Developing a fast and accurate path planning with collision avoidance is mandatory for automation of robotic systems. This paper presented three different approaches.

1. The static approach proved to be very fast, over a full map of a hospital floor, and yet did not ensure that path found could necessarily be achieved by a non-holonomic robot.

2. The dynamic optimization solution showed achievable trajectories, for simplified scenarios, while requiring much higher convergence times.

3. The hybrid approach showed the best trade-off between accuracy, speed, and flexibility.

As of now, only the static approach has been validated on a real robotic system. Implementing the hybrid solution - relying on dynamic optimization - on the embedded hardware is a next step to this work. This might not be trivial, and a first proof of concept might rely on IoT communication protocols (such as MQTT or OPC UA) to communicate with Modelon Impact cloud software.

Acknowledgements

The first version of the dynamic optimization and multibody models, as well as the associated web application, presented in Section 3 were developed in the scope of a paid project with Modelon AB. The authors would like to thank especially Peter Sundström for his great contributions to this project.

References

Åkesson, Johan (2008). "Optimica—An Extension of Modelica Supporting Dynamic Optimization". In: *6th International Modelica Conference, Bielefeld*.

Coïc, Clément, Johan Andreasson, et al. (2020-10). "Collaborative Development and Simulation of an Aircraft Hydraulic Actuator Model". In: *Proceedings of the Asian Modelica Conference, Tokyo*. DOI: 10.3384/ecp202017467.

Coïc, Clément, Marc Budinger, and Scott Delbecq (2022-10). "Multirotor drone sizing and trajectory optimization within Modelon Impact". In: *Proceedings of the American Modelica Conference, Dallas*. DOI: 10.3384/ECP2118656.

Delbecq, Scott et al. (2021-01). "Trajectory and design optimization of multirotor drones with system simulation". In: *Proceedings of the American Institute of Aeronautics and Astronautics, SciTech*. DOI: 10.2514/6.2021-0211.

Lau, Boris, Christoph Sprunk, and Wolfram Burgard (2013). "Efficient grid-based spatial representations for robot navigation in dynamic environments". In: *Robotics and Autonomous Systems*. DOI: 10.1016/j.robot.2012.08.010.

MITI Research Group, Technical University of Munich (2024). *AURORA: Autonomous Robotic OR Assistance*. https://www.pm.mh.tum.de/en/miti/research/projects/aurora/.

Pacejka, Hans B. (2012). *Tyre and Vehicle Dynamics*. Butterworth-Heinemann. ISBN: 978-0-08-097016-5.

Placed, Julio A. et al. (2022). "A Survey on Active Simultaneous Localization and Mapping: State of the Art and New Frontiers". In: DOI: 10.48550/arXiv.2207.00254.

Siemens Healthineers AG (2025). *CIARTIC Move: Robotic C-Arm*. https://www.siemens-healthineers.com/surgical-c-arms-and-navigation/mobile-c-arms/ciartic-move.

DOI
10.3384/ecp21873

Prototypical Control for the Digital Twin of Aircraft Environmental Control System

Corentin Lepais[1] Dirk Zimmer[1]

[1]German Aerospace Center, Germany, {corentin.lepais,dirk.zimmer}@dlr.de

Abstract

A digital twin of the overall aircraft Environmental Control System is being developed as part of TheMa4HERA, a large European research initiative. It shall support verification and validation activities by virtually demonstrating the behavior of the complete system in various conditions. To this end, a prototypical control scheme needs to be developed so that a dynamic simulation through complete flight missions is enabled. The prototypical control scheme is tuned using a simplified version of the Digital Twin which focuses on robustness and fast computation time, while making it robust enough to be stable when used with the high-fidelity Digital Twin. An already stable and working controller for simulating the detailed Digital Twin, provides a significant gain of time, allowing for immediate preliminary results on both steady-state and transient system behaviors. This paper describes the base model of the Digital Twin and the methodology used to design the prototypical control architecture.

Keywords: Digital Twin, Modelica, Environmental Control System, Prototypical Control

1 Introduction

Inside an aircraft, the Environmental Control System (ECS) controls temperature, pressure, humidity and fresh air supply of the pressurized areas of the fuselage (Sielemann et al. 2007). Conventionally, this system uses bleed air from the engine as a source of fresh air. Therefore, the engines act as compressors for the cabin, and excess energy can be used in an air cycle machine to cool the air before distributing it to the fuselage. Consequently, ECS increases engine fuel consumption and hence, greenhouse gas emissions. To reduce these emissions, a number of research projects are looking at hybrid, hydrogen-powered or sustainable-fuel aircraft. These new aircraft concepts thus require innovative ECS architectures to replace the traditional bleed air-driven systems.

As part of the European Clean Aviation project TheMa4HERA, the German Aerospace Center (DLR) is developing a Modelica-based Digital Twin of the ECS for a hybrid electrical regional aircraft. Modelica is an object-oriented modeling language (P. Fritzson 2014) and has already been successfully used in the creation of digital twins (Delussu et al. 2022) (Peter Fritzson 2021) (Magargle et al. 2017). The aim of this model is to serve as a virtual test bench for evaluating different ECS architectures throughout a complete gate-to-gate mission and under different environmental conditions (standard, hot or cold day, high or low ambient humidity) and to support the verification and validation process by demonstrating the behavior of the complete ECS architecture in a representative virtual environment. To guarantee the validity of the results, high-fidelity models from each component of the Digital Twin are built by the different partners inside the project and validated with dedicated hardware tests. Then, they are integrated as Functional Mock-up Unit (FMU) in the simulation framework built by DLR using the Thermofluid Stream Library (TFS) (Zimmer 2020), to form the complete Digital Twin. Inside this framework, components are defined as replaceable to be able to easily switch between the robust but simple versions and the high-fidelity ones once they are available. The integration of FMU follows a new method based on a control-like approach described in (Gebhart and Lepais 2025). In this method, FMU outputs, such as a mass flow rate, are treated as setpoint signals used to control the actuator models, such as a compressor, from the TFS library. With this approach, one can integrate FMUs inside a Modelica library framework without decreasing the robustness of the overall model, especially at initialisation. Finally, the Digital Twin is simulated for complete gate-to-gate mission under different environmental conditions. This combination of hardware tests and virtual simulations allows to test and compare different ECS architectures and to evaluate their performance at aircraft level.

In order to carry out these simulations in a realistic way, a prototypical control must be designed to guarantee the respect of various mission setpoints, such as temperature and pressure inside the cabin. This prototypical control must be as robust as possible in order to accommodate various simulation conditions and ECS models of varying fidelity levels. The robustness is particularly crucial since the controller design is carried out without high-fidelity models, relying solely on a simplified version of the Digital Twin that prioritizes computational robustness and fast simulation times. Both robust and high-fidelity models share the same interface, enabling the design of a common control architecture but exhibit different dynamic behaviors, such as response time or damping. The goal of this prototypical control is not to optimize performances with the detailed models while tuning it with the simpler

ones, but to ensure it can be directly applied with the high-fidelity models in order to test their integration inside the overall Digital Twin and get preliminary results on both steady-state and transient behaviors.

The Digital Twin framework with robust models, detailed in section 2, includes all necessary components for running a complete mission, such as an atmospheric environment model for outer temperature, pressure and humidity, a mission profile block to define mission-related inputs, such as aircraft altitude, speed and controller setpoints. It also defines the interfaces of the different components and how they exchange information in the Digital Twin. In this regard, the Modelica language proves to be highly valuable because it enables the creation of a common interface for a given class of models which makes the exchange of one model by another version very simple and easy. The prototypical control scheme is designed around this common interface definition to ensure compatibility with both the base and high-fidelity versions of the Digital Twin, as shown in section 3. Finally, section 4 presents the simulation results of the base Digital Twin and provides a comparative analysis of two ECS pack architectures, both simulated using the same prototypical control to demonstrate its robustness.

2 Digital Twin

This section outlines the architecture of the Environmental Control System, providing an overview of the entire system. It describes also the objectives of the control architecture, detailing the associated requirements and the various actuators available inside the system.

2.1 Presentation of the model

The open-source Modelica library DLR Thermofluid Stream (TFS) is used as the main modelling library to develop the Digital Twin. This library enables fast and robust modeling of complex thermofluid architectures (Zimmer 2020; Zimmer, Meißner, and Weber 2022), and therefore aligns with the intended goal for the base version of the Digital Twin. The development is done using Dymola 2022x, since it is focus on Modelica development and supports the import and export of Model Exchange and Co-simulation FMUs 2.0. The overall model describes the Environmental Control System of a future regional hybrid electric aircraft, including two ECS packs, a model of the pressurized areas of the fuselage, one outflow valve (OFV) and the ambient environment. An overview of its architecture is available in this paper (Gebhart and Lepais 2025) and the top-level view in Modelica is shown in Figure 1.

ECS packs are used to supply fresh air to the crew and passengers, to control the temperature of the cabin and cockpit and to cool down the avionic bay. The reference architecture consists of an electric cabin compressor connected to the air cycle machine (ACM) and is shown in Figure 2. It is adapted from (Bender 2018) which describes a 3-wheel bootstrap cycle that uses bleed air from the engines as a source of fresh air for the cabin. Since the goal is to model a bleedless aircraft, the pack has been electrified and the bleed air is replaced by fresh air from the outside compressed using an electric cabin compressor. The other modification concerns the ram air fan which is not connected to the turbine anymore, as it is the case in the 3-wheel bootstrap cycle architecture, but to an electric motor in order to be able to fully control the ram air mass flow inside the ram air channel, especially on ground. After being compressed by the electric compressor, fresh air is cooled down against the ram air flow before entering the conventional ACM itself. At this point, part of the air flow is separated and bypassed through the Temperature Control Valve (TCV). The remaining part is compressed a second time, cooled down against the ram air channel and expanded inside the turbine to a sufficient pressure level. Before being expanded, air passes through a water extractor to remove the excess humidity. This dehumidification process, called high-pressure dehumidification, is very important on ground to prevent an excess of humidity inside the cabin and ice at the outlet of the turbine. Finally, air from the turbine is mixed with the separated air from the TCV to reach the correct temperature before leaving the ECS pack. As in today's commercial aircraft, the model includes two ECS packs for the sake of redundancy, contained in the block *environmentalControlSystem* in Figure 1. Compressors, turbine and heat exchangers models are directly taken from the TFS library. Their sizing parameters (e.g. reference mass flow rate, heat exchange coefficient) have been defined according to the project requirements and by manual testing to ensure sufficient cooling in the all conditions. Therefore, these components do not represent existing ones.

The fuselage is a generic model of air circulation between the different pressurized areas (cabin, cockpit, avionic bay and cargo bay) and through systems strongly related to air circulation (mixer, recirculation fans, outflow valve). Its implementation in Modelica is shown in Figure 3. Thermal loads, such as passengers' heat, lights and electric devices as well as heat exchanges inside each area are also part of the model, but there is no leakage to the outside. Certain thermal loads such as battery, fuel cell, or power electronics (converters, ...) are not taken into account because they have their own cooling system. Fresh air from both ECS packs enters the fuselage on the left, is mixed with recirculated air inside the mixer and afterwards distributed to the cabin and the cockpit. Air from the cabin goes directly to the cargo bay while air from the cockpit passes through the avionic bay before entering in the cargo bay. At this stage, air is warmer and more humid than fresh air. A fraction of this air leaves the fuselage by the right to be expelled through the outflow valve to maintain the correct pressure inside the pressurized areas while the rest is fed back to the mixer by the recirculation fans. Inside the cabin and the cockpit, passengers and crew members are modeled by combination of a sink and a source, representing respectively the inhalation of cabin air and the exhalation of saturated, warmer air. With this

DOI 10.3384/ecp21881

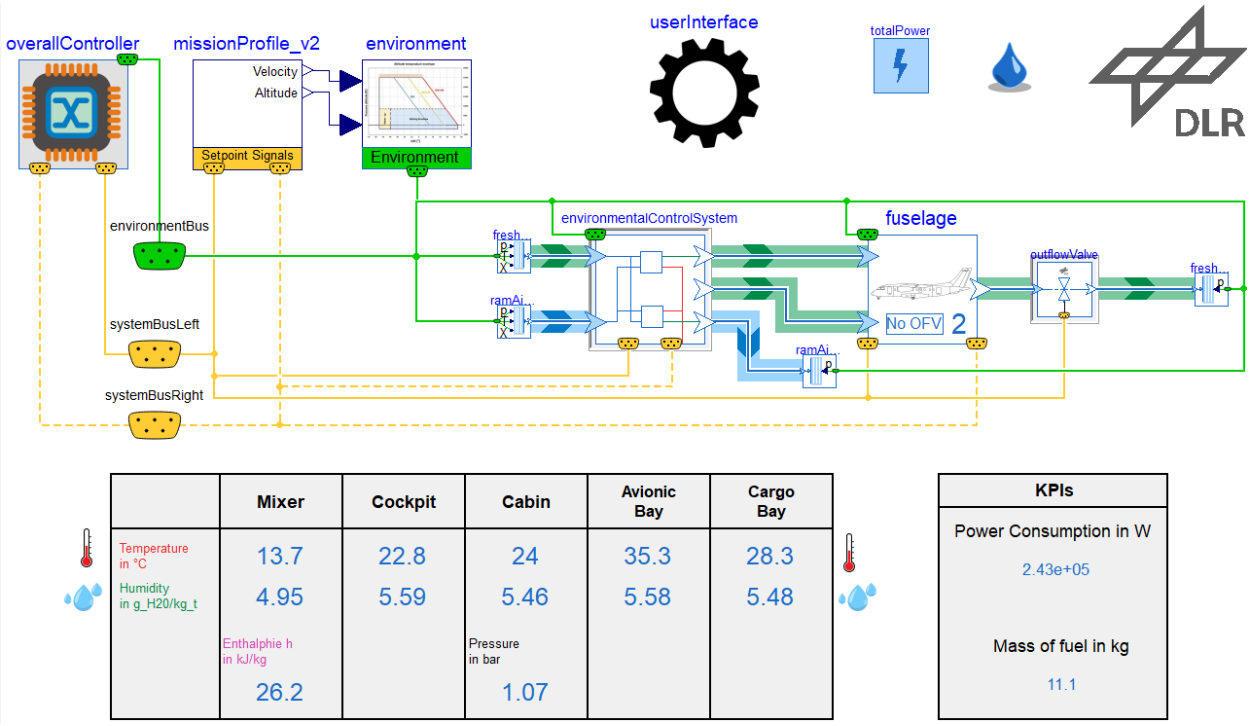


Figure 1. Digital Twin of the Environmental Control System

approach, the additional humidity brought by the passengers when breathing can be taken into account. Additionally, a heat flow source accounts for the body's heat. Each passenger is assumed to breathe at a constant volume flow rate.

The last part of the Digital Twin is the mission and environment model. It computes the temperature, pressure and mass fraction of water outside the aircraft for a given altitude profile and ground conditions, according to the International Standard Atmosphere (ISA). It also models the amount of solar radiations, as a linear function between defined ground and cruise levels. Solar radiations are used to compute the heat transfer through the windows and the fuselage of the aircraft.

This Digital Twin can be simulated for two different mission cases:

- Normal case: gate-to-gate mission with both ECS packs working
- Failure case: gate-to-gate mission with only one ECS pack working.

As described in (Gebhart and Lepais 2025), 3 typical ground conditions can be simulated:

- Standard day (STD): temperature of 15°C (288K) – 100% relative humidity
- Hot day (HD): temperature of 40°C (313K) – 50% relative humidity
- Cold day (CD): temperature of -25°C (248K) – 100% relative humidity

The "Hot day" case is the most challenging case for the design of the ECS pack and the control architecture because the highest amount of cooling power is needed.

This model is an ideal model in many different aspects. All variables can be measured and all sensors are ideal. Hence, there is neither noise nor any transport delay from the sensors or in the actuators command signals.

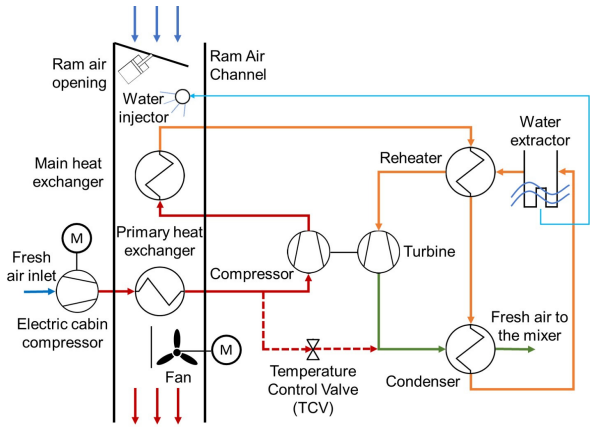


Figure 2. ECS pack architecture

2.2 Control requirements

The control requirements are various. Concerning the environmental conditions inside the cabin, the following requirements are defined:

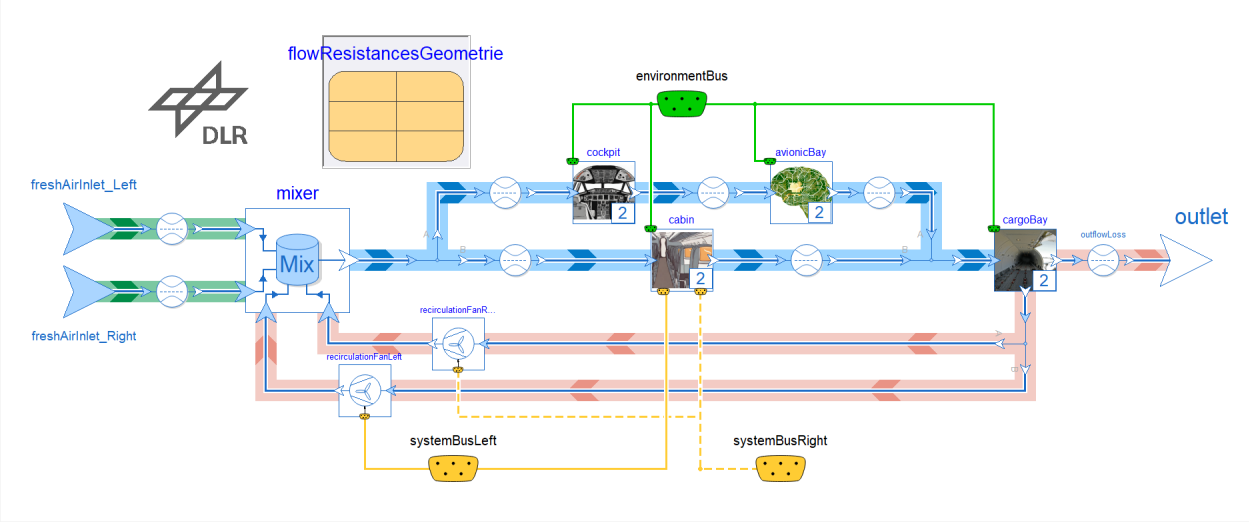


Figure 3. Fuselage architecture

- Temperature inside the cabin: in the certification specification and acceptable means of compliance 25 (CS-25), EASA defines all the constraints concerning cabin heating and ventilation for a cabin temperature equals to 24°C (297K) (European Union Aviation Safety Agency 2022).

- Pressure inside the cabin: a pressurization curve defines the cabin altitude depending on the aircraft altitude. The cabin altitude is converted into cabin pressure using the ISA model. The resulting cabin pressure curve is shown in Section 4.

- Cabin volume flow: the volume flow of air entering the cabin must be kept constant at a certain level to ensure a correct ventilation of the cabin. In today's aircraft, the mixing ratio between fresh air and recirculated is around 50% (Bagshaw and Illig 2019), which gives a volume flow of 0.75m³/s

In the current state of the model, only the cabin environment is controlled that is why there are no requirements concerning the other pressurized areas.

There are two more constraints for the ECS packs:

- Fresh air mass flow: EASA CS-25 defines a minimum amount of fresh air per person to be provided by both ECS packs in normal operation: 0.25 kg/min/person (0.55 lb/min/person)(European Union Aviation Safety Agency 2022). The aircraft transports 80 passengers, 2 crew members and 2 pilots. An additional 20% mass flow of fresh air is also considered to meet additional cooling requirements. It leads to a total fresh air mass flow in normal operation of 0.42kg/s, i.e. 0.21kg/s per ECS pack. In case of failure of one ECS pack, the minimum fresh air flow is decreased to 0.182 kg/min/person (0.40 lb/min/person). In this case, no additional fresh air is considered, leading to a total fresh air mass flow in case of failure of 0.255 kg/s.

- Compressor temperature: this requirement is only used to control the ECS packs. It defines the maximum possible temperature at the outlet of the compressor mechanically connected to the turbine. This temperature will also influence the cabin temperature. Thus, the setpoint value could be the output of a controller that takes as input the cabin temperature error and the flight profile. But this would make the prototypical control scheme more complex, which is not the aim of this first implementation. Therefore, a constant value is considered for the compressor temperature setpoint. The chosen value is 180°C (453K), as done in (Bender 2018), to prevent using too heavy and expensive compressors.

A last constraint is the humidity level inside the cabin. ECS packs are also used to dehumidify the air and avoid too high humidity levels inside the cabin. Usually, the goal is to keep the relative humidity inside the cabin around 20% (Bagshaw and Illig 2019). On cruise it is not a problem since the absolute humidity outside is very low due to the low temperature, but it can be very different on ground for a hot day, for example. The current architecture of the ECS pack contains a water extractor before the turbine, which removes the excess of water inside the air above saturation. Therefore, the inlet temperature of the water extractor is very important since it will directly influence the amount of water extracted by this component. Unfortunately, the humidity level inside the cabin cannot be directly controlled with the current architecture. Thus, the humidity level is verified afterwards to make sure it stays around this value. If it is not the case, the size of the heat exchangers has to be modified.

A summary of all the values for each requirement is given in Table 1.

DOI
10.3384/ecp21881

Table 1. Target setpoints for the control architecture

Requirements	*Normal Case*	*Failure Case*
Cabin temperature	24°C	24°C
Cabin pressure	See Figure 6	See Figure 6
Cabin volume flow	0.75m³/s	0.75m³/s
Fresh air mass flow	0.42kg/s	0.25kg/s
Maximum Compressor outlet temperatures (x2)	180°C	180°C

2.3 Actuators and controlled variables

All target variables (temperatures, mass flow, volume flow, pressure) can be measured so they will be directly used in the control scheme. There are 11 actuators available in the model. Each ECS pack has 4 actuators inside, as shown in Figure 2: cabin air compressor, Temperature Control Valve (TCV), ram air channel (ram air opening and electrical fan). The ram air channel is considered as one big actuator, even if the flow inside is controlled by both the inlet valve and the electrical fan. The fan comes in addition to the valve to increase the ram air flow if needed once the ram air channel is fully open. It is often the case on ground when the aircraft speed is not sufficient enough to generate a ram air flow. Additionally, there are 2 recirculation fans in the fuselage model, as shown in Figure 3, and one outflow valve. In total, there are 11 actuators and each of these actuators will be connected to a Single Input Single Output (SISO) controller.

Input signals have been normalized between 0 and 1. For valve actuators (TCV, outflow valve and ram air channel opening), the opening command is limited by default between 0, valve fully closed, and 1, valve fully opened. The generic TFS valve model integrates leakage when the valve is fully closed, meaning that there is still a small air flow across the valve. For turbo-actuators (compressor, fan), the rotational speed is taken as input. The reference speed ω_{ref} used in the computation of the pressure ratio is set to 1, therefore normalizing the input speed command between 0 and 1.

3 Control Architecture

This section describes the prototypical control architecture and gives some details about the methodology used to tune the different controllers.

3.1 Control Allocation

To design the control loop, all variables are considered independently. For each variable, a limited PI-Controller with anti-windup is designed. Anti-windup is used in order to guarantee a robust controller when dealing with saturation (Bohn and Atherton 1995). These controllers take the error signal between the setpoint and the current value of the controlled variable and compute the input signal for the actuator. Since only SISO controllers are considered, the control loop for each controlled variables defined in Table 1 can be represented by the generic control loop in Figure 4. The control allocation, i.e. which actuator is used to control which variable, is described in Table 2. The use of the TCV to control the cabin temperature and of the ram air opening to control the compressor outlet temperature are derived from (Bender 2018).

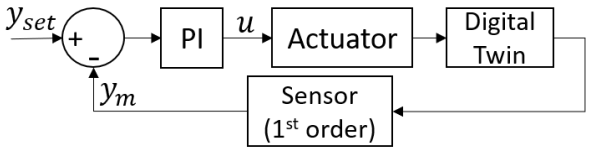


Figure 4. Generic control loop for one variable

For the cabin temperature, there are 2 control loops, one for each TCV. It only happens that both loops are identical since there is only one cabin and therefore one setpoint and one sensor signal. It is the same for the cabin volume flow with the recirculation fans.

3.2 Tuning of the different controllers

3.2.1 Limited PI-Controller with anti-windup for the fresh air mass flow

This controller computes a speed command between 0 and 1 for the electric cabin compressor inside the ECS pack, which is used to regulate the mass flow of fresh air. It has been tuned using a simple test model. In this test model, the fresh air mass flow setpoint follows a step between 0.1kg/s (first value to initialise and start the simulation) and 0.21 kg/s (setpoint of the complete model in normal operation). All other actuators inside the ECS pack (TCV, ram air fan, ram air opening) have constant inputs. To tune the controller, only a P-Controller was considered at the beginning. The gain was changed until the response approached the setpoint. But a steady-state error was still present and the controller output reached the limits. Then, the integrator was added and tuned in order to reach the setpoint value with only a small overshoot and without oscillations afterwards. Finally, the proportional gain was decreased to keep the controller output between the boundaries and avoid saturation, but without decreasing too much the response time. Anti-windup uses the default values of the Modelica limPID model.

3.2.2 Limited PI-Controller with anti-windup for the cabin volume flow

This controller computes a speed command for the recirculation fan between 0 and 1. The tuning of this controller follows the same process as described above. A simple test model including the recirculation fan and the controller, connected to a volume inside a loop, has been used for the tuning. The controller is used to control the speed of the recirculation fan in order to set a given volume flow in the loop. The setpoint for the volume flow is a step between 0.25m³/s and 0.75m³/s. The method de-

Table 2. Target setpoints for the control architecture

Variable	y_{set}	y_m	*Actuator*	u
Cabin temperature	T^{set}_{cabin}	T_{cabin}	TCV (x2)	Valves opening
Cabin pressure	p^{set}_{cabin}	p_{cabin}	Outflow valve	Valve opening
Cabin volume flow	$\dot{V}^{set}_{cabin}$	$\dot{V}_{cabin}$	Recirculation fans (x2)	Fan speed
ECS pack left fresh air mass flow	$\dot{m}^{set}_{left}$	$\dot{m}_{left}$	Cabin compressor left	Compressor speed
ECS pack right fresh air mass flow	$\dot{m}^{set}_{right}$	$\dot{m}_{right}$	Cabin compressor right	Compressor Speed
Maximum compressor outlet temperature left	T^{set}_{comp}	$T_{comp,left}$	Ram air channel left	Valve opening, Fan speed
Maximum compressor outlet temperature right	T^{set}_{comp}	$T_{comp,right}$	Ram air channel right	Valve opening, Fan speed

scribed above is also applied to tune the controller, with the goal to have a response time as small as possible without oscillations or significant overshoot.

3.2.3 Limited PI-Controller with anti-windup for the cabin pressure

Concerning the cabin pressure, the controller to be tuned computes the input signal of the outflow valve (0 valve fully closed - 1 valve fully opened). The test model used for the tuning is very simple and consists of a volume with a constant inlet mass flow. The outflow valve is placed at the outlet of the volume before the sink and is used to regulate the pressure inside the volume. The setpoint for the pressure is not a step but a trapezoid profile reproducing approximately the cabin pressure profile during a mission. The method used to tune the controller stays the same. The controller was tuned to be as fast as possible without creating too much overshoot in order to limit as much as possible drastic pressure changes in a short amount of time.

3.2.4 Limited PI-Controllers with anti-windup for the cabin temperature & compressor temperature

The control loop for the maximum compressor outlet temperature is distinct from the other loops. In this case, the ram air channel is treated as a single actuator, controlled by a SISO PI-Controller. However, two command signals are required: one for the valve opening and another for the fan. The fan operates in tandem with the valve to increase the mass flow when necessary. Since the control signals for both actuators are normalized between 0 and 1, the PI-Controller computes the sum of these two command signals, resulting in a combined signal ranging from 0 to 2. If this sum exceeds 1, the valve opening is set to 1, and the fan speed is adjusted to the remaining value. If the sum is less than or equal to 1, only the valve is opened, and the fan speed remains at 0. This command allocation strategy is made possible by the normalized actuator command signals, enabling the use of a SISO controller to manage two actuators simultaneously.

The method used to tune the two controllers differs from the one mentioned before. For the previous controllers, it was possible to use simple test models to tune them because the cabin pressure, the cabin volume flow and the mass flow of fresh air provided by one ECS pack behaves independently. To control the pressure, the outflow valve controller regulates only the amount of air expelled to the outside and does not need to know the exact mass flow of fresh air supplied by the ECS packs or the exact volume flow inside the cabin. In the test model used for the tuning, it was therefore enough to define a constant mass flow at the inlet of the volume. However, it doesn't work for the cabin and compressor temperatures. The cabin temperature is regulated by changing the pack outlet temperature, but this outlet temperature is strongly linked to the outlet mass flow of the pack, to the recirculation flow and also to the mass flow inside the ram air channel. The choice of a constant temperature setpoint for the compressor outlet temperature was also a way to split these two controlled variables.

Therefore, the complete Digital Twin model has been used for the tuning and, more precisely, a linear version of the model. Linearization has been carried out using the linear analysis tool from Modelica. The TCVs and the ram air openings were set to constant input values and the simulation was run on hot day on ground, since it is the most challenging case, until reaching steady-state (correct pressure and volume flow inside the cabin, correct outlet mass flow from both ECS packs and all other variables constant or at least stabilized). Then, the model was linearized around this steady-state point. Inputs were the TCV command signal and the ram air opening and outputs were the cabin and compressors outlet temperatures. This linear model was then imported in Simulink and both controllers were tuned using Simulink PID Tuner App. During this tuning, robustness was an important aspect so the gain and phase margins have been maximised. The response time of the cabin temperature was chosen to be as fast as possible. To finish, the coefficients of the Simulink PID were translated into the Modelica limPID to finally obtain the

DOI
10.3384/ecp21881

last two controllers.

4 Results

This section starts by showing some simulation results obtained when simulating the base version of the Digital Twin throughout a complete gate-to-gate mission using the prototypical control architecture. In a second time, the ECS pack model is replaced by a new version to mimic the exchange of the simple model by the high-fidelity one. The new model is again simulated on a complete mission, demonstrating the stability of the controllers with a different model and showing the kind of comparison that can be performed with the Digital Twin.

4.1 Simulation of the base version of the Digital Twin

The Digital Twin is simulated by considering a unique gate-to-gate mission and the 3 environmental conditions defined in section 2 (standard, hot and cold day). The mission is defined by an altitude profile and a speed profile, shown in (Gebhart and Lepais 2025). Initially, the aircraft is stationary on ground, at sea level (0m), to simulate the passenger boarding phase. After the taxi phase and take-off, the aircraft climbs at 7620m and progressively reaches its cruise speed (154 m/s). The cruise phase has a duration of 24 minutes. After the descent, the aircraft lands at sea level (0m) and taxis to the boarding gate. Once at the boarding gate, the mission is finished that is why there is no stationary phase at the end.

The results display all the controlled variables outlined in Table 1, with the exception of the compressor outlet temperature for the right ECS pack. Since both packs and their associated controllers are identical, their results are identical as well. Therefore, only the compressor outlet temperature for the left ECS pack is presented in Figure 9. The prototypical control architecture is successful on standard and hot days. Indeed, Figure 5 to Figure 9 show that all controlled variables reach their target setpoint. Concerning the compressor outlet temperature shown in Figure 9, the goal of the control loop is mainly to prevent reaching a too high temperature, so a lower steady-state temperature is not really a problem.

On the other hand, on cold day the temperature inside the cabin is too low, as shown in Figure 5. The problem doesn't come from the controllers but from the ECS pack architecture itself. Indeed, in Figure 9, the compressor temperature on cold day is also very low compared to standard and hot days. However, this temperature is one of the hottest one inside the ECS pack. Therefore, it can be inferred that the ECS pack simply cannot heat the air sufficiently to achieve a correct temperature inside the cabin and another device is needed in this case, for example an electric heater. This behavior is directly linked to the decision of designing a bleedless ECS pack. In conventional bleed packs, the bleed air comes directly from the engine at high temperature and therefore always needs to be cooled, even on cold days.

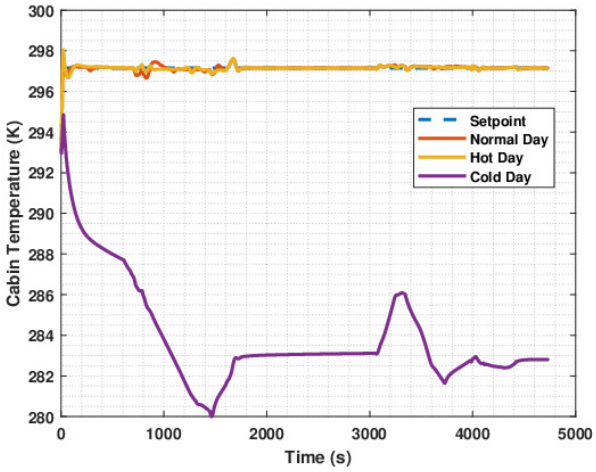


Figure 5. Cabin temperature for a gate-to-gate mission with 3 different environmental conditions

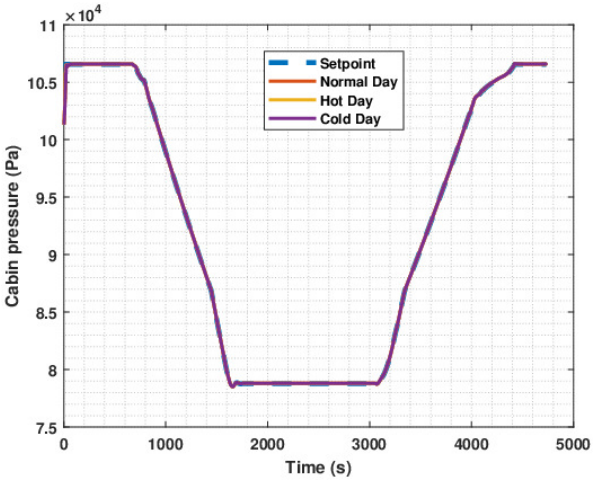


Figure 6. Cabin pressure for a gate-to-gate mission with 3 different environmental conditions

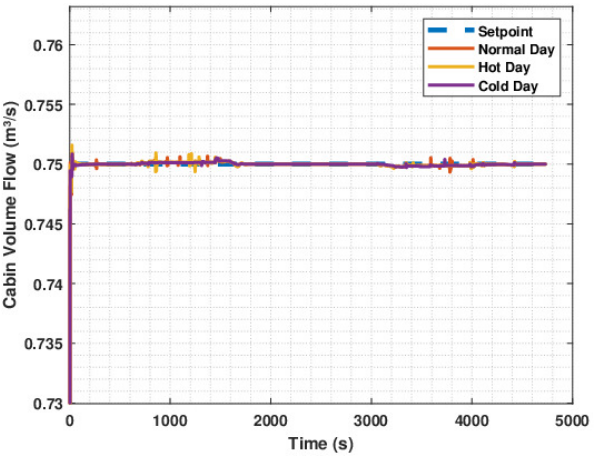


Figure 7. Cabin volume flow for a gate-to-gate mission with 3 different environmental conditions

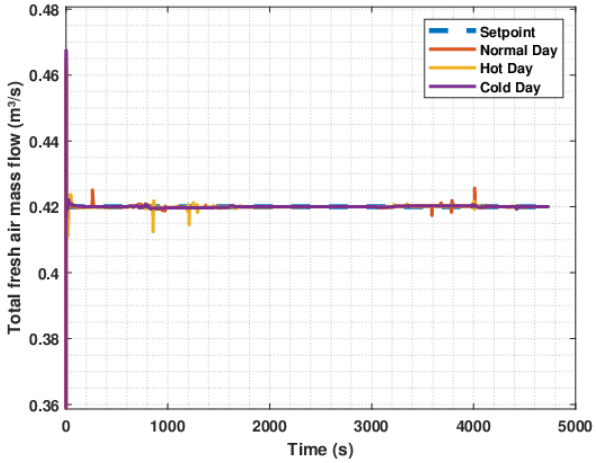


Figure 8. Total mass flow of fresh air provided by the ECS for a gate-to-gate mission with 3 different environmental conditions

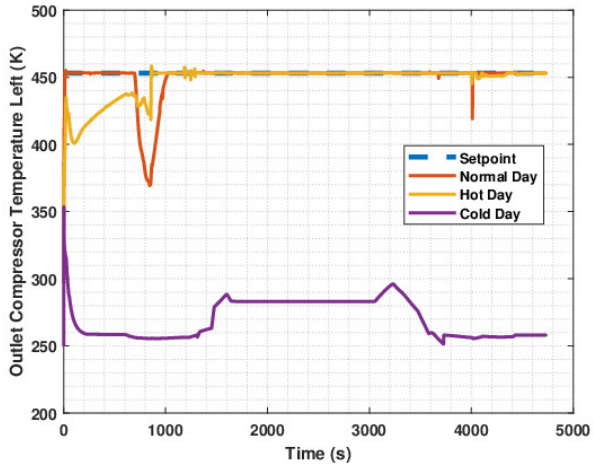


Figure 9. Left compressor outlet temperature for a gate-to-gate mission with 3 different environmental conditions

4.2 Comparison with a second ECS pack architecture

Now that the prototypical controller has been validated, the current ECS pack architecture is replaced by a modified version of a 4-wheel bootstrap cycle, described in (Sielemann et al. 2007) and (Bender 2018), to mimic what would happen when replacing the base ECS pack by a high-fidelity FMU. The conventional 4-wheel bootstrap cycle is based on the 3-wheel bootstrap cycle and adds a second turbine at the outlet of the pack, after the condenser. One of the goals of this architecture is to avoid the problem of icing at the outlet of the first turbine. The ram air fan is still driven by an electric motor. Bypassed air from the TCV is now injected at the outlet of the second turbine.

The results of the simulation with this new architecture are shown in Figure 10, Figure 11 and Figure 12. Since only the ECS packs have been changed, the focus is given on the new cabin temperature and compressor temperature. Results are only shown for standard and hot days since the new pack does not include neither any heating component.

Figure 10 shows the temperature difference inside the cabin between the new and the reference architectures ($\Delta T_{cab} = T_{cab}^{new} - T_{cab}^{ref}$). The difference is small on hot day (maximum 0.2K), but is bigger on standard day (maximum 0.8K) with oscillations during the climbing phase (at time = 1000s). When comparing with Figure 5, there are also oscillations at time = 1000s on standard day. So the controller generates stronger oscillations with the new architecture, but it is still able to ensure the correct temperature inside the cabin. Table 3 also shows a larger overshoot interval for the cabin temperature with the second ECS pack architecture but the difference is relatively small, around 0.5%. For both ECS packs architecture, the cabin temperature is maintained inside a bandwidth of 3% around the setpoint for the whole mission. Since the robustness is the main criteria, the cabin temperature controller still fulfills its goal.

Table 3. Comparison of the performances of the cabin temperature controller for initial and second ECS pack architectures

	Overshoot (%) $[min; max]$	*Mean Square Error Cabin Temperature*
Initial ECS pack	$[-2.06; 2.84]$	0.0777
Second ECS pack	$[-2.43; 3.06]$	0.1218

Figure 11 shows the temperature difference of the compressor outlet between the new and the reference architectures ($\Delta T_{comp} = T_{comp}^{new} - T_{comp}^{ref}$). On standard day, the biggest temperature difference also happens at time = 1000s and is negative. It means that the compressor outlet temperature of the new architecture is smaller than the compressor outlet temperature of the reference architecture. Since the setpoint for the compressor temperature defines a maximum value and not a target value, this smaller temperature is not an issue. However, at time = 4000s, the compressor outlet temperature of the new architecture is higher (around 20K) than the one of the reference architecture. But in Figure 9, the compressor temperature is lower than the temperature setpoint at time = 4000s on standard day. So, this positive difference does not mean that the compressor temperature is 20K above the temperature setpoint. On hot day, the compressor outlet temperature of the new architecture is always lower or equal to the reference architecture.

As for the other controlled variables (cabin pressure, cabin inlet volume flow, fresh air mass flow) they all reach their target setpoint. Performances may vary (more oscillation for example) but the steady-state behavior is the desired one.

Since the prototypical control is working successfully for both ECS architectures, the Digital Twin can now be

DOI
10.3384/ecp21881

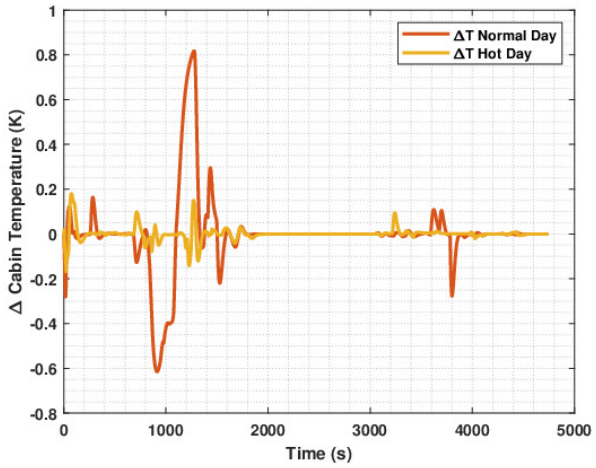


Figure 10. Cabin temperature difference between the new architecture and the reference one

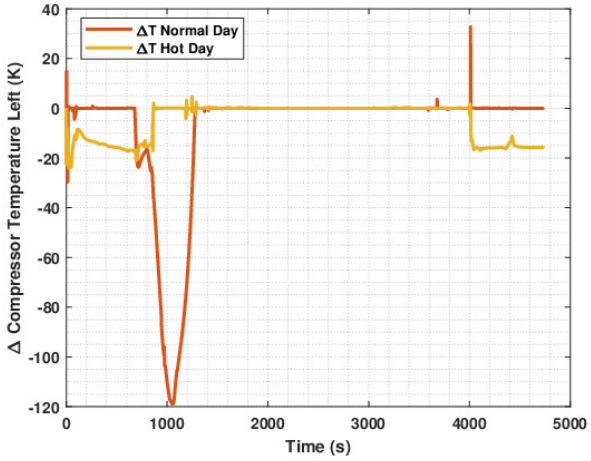


Figure 11. Left compressor outlet temperature difference between the new architecture and the reference one

used to compare them at aircraft level. For example, it can be interesting to compare the total power consumption of both architectures during the flight. Figure 12 shows the difference between the power consumption of the new architecture and the reference one ($\Delta P = P^{new} - P^{ref}$), on standard and hot days. In both cases, the difference is negative. So the new architecture has a lower power consumption for the same prototypical control. By looking more closely at the results, this difference mainly comes from the electric cabin compressors which consume less power, meaning a lower flow resistance to overcome to reach the fresh air mass flow setpoint. Besides, the TCVs are less used in the second architecture. Since, the expansion is done in two steps and air is heated up between the two steps, the temperature at the outlet of the second turbine is higher than the temperature at the outlet of the turbine in the reference architecture. A higher temperature means less hot air needed from the TCV to reach the required pack outlet temperature. A lower mass flow across the TCVs could be the reason of the lower flow resistance and so the lower power consumption from the electric cabin

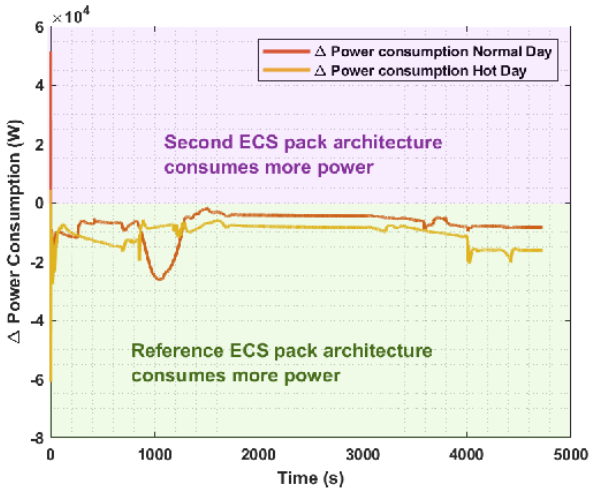


Figure 12. Power consumption difference between the new architecture and the reference one for both standard and hot day

compressors. From a power consumption point of view, the new ECS architecture seems more efficient but power consumption is not the only relevant parameter to design an aircraft. For example, this new ECS will certainly be heavier due to the second turbine. Therefore, the impact of this increased mass on the aircraft should also be taken into account. This short comparison gives an overview of how different architecture can be evaluated at aircraft level using the Digital Twin.

5 CONCLUSIONS

This paper presented the development of a prototypical control for the Digital Twin model of an aircraft environmental control system.

The approach is based on SISO limited PI-Controllers with anti-windup to robustly deal with the saturation of the different actuators, and normalized actuator command signals to ease the transfer of the control authority from one actuator to the other. The normalization is also a good solution to avoid including inside the control architecture the saturation characteristics of specific actuators, which then would require to change the tuning of the controllers each time a characteristic changes. It shows that is possible to obtain a global prototypical control that works for a complete gate-to-gate mission, for different environmental conditions and different ECS pack architectures. Improvements are still needed, especially on cold day, but the problem originates from the ECS pack architecture itself and not from the control architecture.

A comparison with an alternative ECS pack architecture was also conducted. All setpoints were reached without altering the prototypical control, although the transient behavior differed, with more and bigger oscillations observed in the cabin temperature. Nevertheless, the robustness of the controller was successfully tested, although further tests will be needed to confirm the results, especially with FMU models. The new architecture had a

lower power consumption compared to the reference one while applying the same prototypical control, highlighting the type of study that can be carried out with the Digital Twin. This demonstrates the possibility to develop a prototypical control using simple models while ensuring it is robust enough for models with different dynamic behaviors, and therefore saving time on tuning again the controllers when changing the models. The next step would be to test the prototypical control with an actual FMU of a high-fidelity ECS pack model to assess the impact of a more detailed dynamic behavior and of sampling time, particularly if a co-simulation FMU is being used. Furthermore, it would also be interesting to observe the effect of sensor and actuator signal delays, sampling times and noise on the stability of the controller. These factors often reduce the robustness, making it potentially unstable when used on the high-fidelity models.

Once the robustness is confirmed, it could also be interesting to test different controller architectures in order to optimize given parameters such as the energy consumption at aircraft level. It would be interesting to see if it is possible to design a control architecture which would minimize a top-level aircraft indicator, independently of the model used for tuning. For future hybrid electrical aircraft, a lowest possible energy consumption of the Environmental Control System would be very interesting because it could lead to smaller batteries and so significantly reduce the weight of the aircraft. This type of study would also require to validate the Digital Twin results with field data to assess the accuracy of the simulations.

Acknowledgements

This work was carried out as part of the European project **TheMa4HERA**. The project is supported by the Clean Aviation Joint Undertaking and its members. Funded by the European Union under Grant Agreement No. 101102008.





References

Bagshaw, Michael and Petra Illig (2019). "47 - The Aircraft Cabin Environment". In: *Travel Medicine (Fourth Edition)*. Ed. by Jay S. Keystone et al. Fourth Edition. London: Elsevier, pp. 429–436. ISBN: 978-0-323-54696-6. DOI: https://doi.org/10.1016/B978-0-323-54696-6.00047-1. URL: https://www.sciencedirect.com/science/article/pii/B9780323546966000471.

Bender, Daniel (2018-09). "Exergy-Based Analysis of Aircraft Environmental Control Systems and its Integration into Model-Based Design". PhD thesis. Technische Universität Berlin. URL: https://elib.dlr.de/124010/.

Bohn, C. and D.P. Atherton (1995). "An analysis package comparing PID anti-windup strategies". In: *IEEE Control Systems Magazine* 15.2, pp. 34–40. DOI: 10.1109/37.375281.

Delussu, Federico et al. (2022). "Experiments and Comparison of Digital Twinning of Photovoltaic Panels by Machine Learning Models and a Cyber-Physical Model in Modelica". In: *IEEE Transactions on Industrial Informatics* 18.6, pp. 4018–4028. DOI: 10.1109/TII.2021.3108688.

European Union Aviation Safety Agency (2022). *Certification Specifications and Acceptable Means of Compliance for Large Aeroplanes - CS-25 Amendment 27*. Tech. rep. Linköping. URL: https://www.easa.europa.eu/en/document-library/easy-access-rules/online-publications/easy-access-rules-large-aeroplanes-cs-25?page=28#_DxCrossRefBm1730187479.

Fritzson, P. (2014). *Principles of Object-Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach*. Wiley. ISBN: 9781118859124. URL: https://books.google.de/books?id=uRnCDwAAQBAJ.

Fritzson, Peter (2021). "The Openmodelica Environment for Building Digital Twins of Sustainable Cyber-Physical Systems". In: *2021 Winter Simulation Conference (WSC)*, pp. 1–12. DOI: 10.1109/WSC52266.2021.9715443.

Gebhart, Raphael and Corentin Lepais (2025). "Novel Control-like Approach for the Robust Integration of Functional Mock-Up Units into Digital Twins". In: *Engineering Proceedings* 90.1, p. 12.

Magargle, Ryan et al. (2017). "A Simulation-Based Digital Twin for Model-Driven Health Monitoring and Predictive Maintenance of an Automotive Braking System." In: *Modelica*, pp. 132–003.

Sielemann, Michael et al. (2007). "A Flexible Toolkit for the design of environmental control system architectures". In: *Proceedings of the First CEAS European Air and Space Conference*. URL: https://elib.dlr.de/51311/.

Zimmer, Dirk (2020). "Robust object-oriented formulation of directed thermofluid stream networks". In: *Mathematical and Computer Modelling of Dynamical Systems* 26.3, pp. 204–233. DOI: 10.1080/13873954.2020.1757726. URL: https://doi.org/10.1080/13873954.2020.1757726.

Zimmer, Dirk, Michael Meißner, and Niels Weber (2022). "The DLR ThermoFluid Stream Library". In: *Electronics* 11.22. ISSN: 2079-9292. DOI: 10.3390/electronics11223790. URL: https://www.mdpi.com/2079-9292/11/22/3790.

Aspects and Ideas for the FMI-based Modeling of Railway Digital Twins

Andreas Heckmann[1] Alexander Poßeckert[1] Vijaya-Bhaskar Adusumalli[2]

[1]Institute of Vehicle Concepts, German Aerospace Center (DLR), Germany,
{andreas.heckmann,alexander.posseckert}@dlr.de
[2]Hitachi Rail GTS Deutschland, Germany

Abstract

This papers reports on activities in the European project MOTIONAL that aims at the development of a digital twin environment which facilitates the modularity, interoperability and composability of complex digital twin assemblies of railway systems. The approach that refers to the Functional Mock-up Interface is justified by a discussion of the comparable activities in industry and in the automotive field compared to particularities in the railway system. The work was initiated by the selection and analysis of nine use cases. An introductory digital twin example illustrates the current implementation status and related aspects, while an outlook presents the integration into the Federated Rail Data Space as the business case and as a vision of the activity.
Keywords: FMI, railways, digital twin

1 Background and Motivation

Europe's Rail Joint Undertaking (EU Rail) established by (Council of European Union, 2021) is the European public-private partnership for rail research and innovation. As described in (Europe's Rail Joint Undertaking, 2022), it aims at

- digitising and greening the railway sector,
- improving its efficiency, amenity, user-friendliness and maintainability,
- supporting the development of a globally competitive European railway industry and
- achieving the Single European Railway Area.

To this end, EU Rail organizes a series of projects[1]. MOTIONAL, the first in this series, inter alia addresses the topic of digitalisation of the railway sector and reach for concepts, models, technologies and methods associated to data, services and digital infrastructur.

The expected output shall be exploited by all EU Rail projects in accessing and exchanging data and in building and executing digital twins. The considered use cases range from virtual certification, automatic train operation to diagnostics and railway asset management originating from other EU Rail projects such as R2DATO[1] or IAM4RAIL[1].

With this background, the digital twin related work in MOTIONAL is specifically focussed on organizing a framework which facilities the **modularity**, **interoperability** and **composability of complex digital twin assemblies**.

After this section on the project background, a short literature review on digital twins and system simulations in railway engineering will be given. This is followed by a step-by-step derivation of the approach taken, which is then illustrated using an introductory digital twin model of specific signalling devices. A more elaborate digital twin model, which is associated to the system architecture in the right hand side of Fig. 1, is being presented separately in an acompanying paper entitled *Collaborative Digital Twin Development for Railway Braking and Traction Applications* at this conference.

2 Related Work

EU Rail mainly emphasizes the expected benefits such as improved safety, availability and lower operational costs in (Europe's Rail Joint Undertaking, 2022), but only roughly specifies the concept as **„ .. a Digital Twin is a virtual representation able to imitate the behaviour of a physical system during the spans of its lifecycle .."**.

It is to the credit of (Sjarov et al., 2020) that they have narrowed down and contextualised the ambiguous, often misunderstood or misused term *digital twin* with the help of their very readable literature research. They conclude their overview on digital twin definitions by what they call "**the smallest common denominator, comprising a real and a virtual world being connected by streams of data and information**." (Singh et al., 2021) as well gives a good overview on characteristics and advantages of digital twins. (Van Der Valk et al., 2022) supplement their paper survey by a qualitative interview study with 15 industry experts. The interested reader may be referred to the long list of references of the three last-mentioned papers in order to find more specific digital twin literature,

The exploitation of the Functional Mock-up Interface (FMI) (Modelica Asssociation Project FMI, 2024) is a

[1] https://rail-research.europa.eu/eu-rail-projects

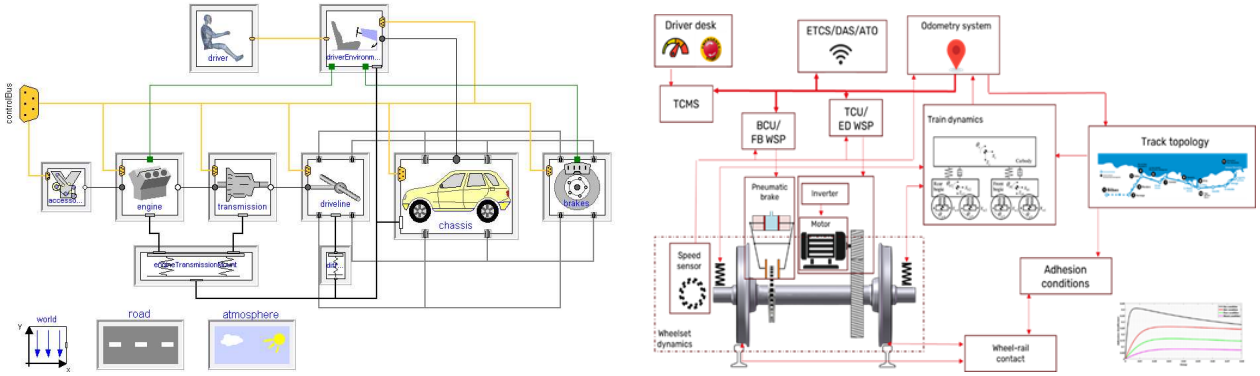


Figure 1. The system architecture of the Vehicle Interfaces Library (Dempsey et al., 2006) on the left compared to an initial architecture proposal in MOTIONAL on the right hand side (courtesy of CAF S.A.).

widespread strategy in order to build up digital twins by assembling individual component models. In fact, this free standard that defines a low-level interface to exchange dynamic models is well suited to organize the simulation of modular multi-domain systems pretty often required in digital twin applications, see e.g. (Wiens et al., 2021).

Although (Golightly et al., 2022) noted a low representation of scientific papers on FMI usage in railways, (Kugu et al., 2023) even used the System Structure and Parametrization standard (Modelica Asssociation Project SSP, 2024), which is layered upon FMI, to present two digital twin use cases from the railway domain. (Kugu et al., 2024) include the set-up and maintenance of digital twins into the Continuous Integration pipeline in order to control their changes and updates process.

The Modelica Railway Dynamics Library (Heckmann et al., 2019) provides the capability to consider vehicle dynamics issues such as traction, comfort and safety in multi-domain engineering tasks on different levels of detail. The library offers a convenient export of portable FMUs that may include railway vehicle particularities such as track joints or wheel-rail contacts. (Heckmann and Streit, 2012) explored the introduction of external infrastructure data given in the open XML-based format (RailML®, 2025) simulating the energy flow in electrical railway networks on which a fleet of railway vehicles is running.

3 Derivation of the Approach

3.1 Comparison to Automotive Activities

For the sake of exemplification of the objectives **modularity**, **interoperability** and **composability**, consider the modeling of a railway vehicle running along the track. This system maybe modularized in subsystems which e.g. represent the track infrastructure, the vehicle dynamics, the traction system, the brakes and so on. In the same manner as the real vehicle system is composed out of integrated subsystem hardware provided by supplying companies, the digital twin of the system is considered to be an assembly of subsystem models as well prepared by the same suppliers. Presumed that the interfaces of each subsystem modul, i.e. its input and output signals, are uniquely specified, the subsystem model of supplier A may be exchanged by the subsystem model of supplier B.

This concept corresponds to a coordinated activity in the automotive field which led to the initial release of the free Modelica Vehicle Interfaces Library (VIL) (Dempsey et al., 2006) in 2006. In fact, Fig. 1 demonstrates the comparability of the former automotive activity with the task considered in this paper by a face to face comparison of exemplary vehicle system architectures. As already indicated by the name, the focus of the VIL was the definition of standard interfaces, which allows for an easy exchange of submodels on brakes, engine, driveline and so on. The component models provided by the VIL only reflect their physical behavior on a very high, simplified level and are assumed to be detailed when actually used in practise. This is an important aspect which is intended to be considered in the context of the modeling of railway digital twins as well.

3.2 Particular Challenges

However, there are number of challenges which appear when the VIL approach is supposed to be transfered into the field of railway digital twins:

- It is a general observation that the modeling of railway vehicles is more complex than for automotive vehicles. It involves more mechanical degrees of freedom, since a train usually consists of more than one car and two-level suspensions are implemented to guarantee both separately, running stability and passenger comfort. In addition, a geometrical track description that is two-times continuously differentiable is required and the very stiff, geometrically and physically non-linear wheel-rail contact is a modeling and simulation challenge on its own, see (Heckmann et al., 2019).

- FMI is a free standard supported by more than 200 different commercial or open modeling tools.

DOI
10.3384/ecp21891

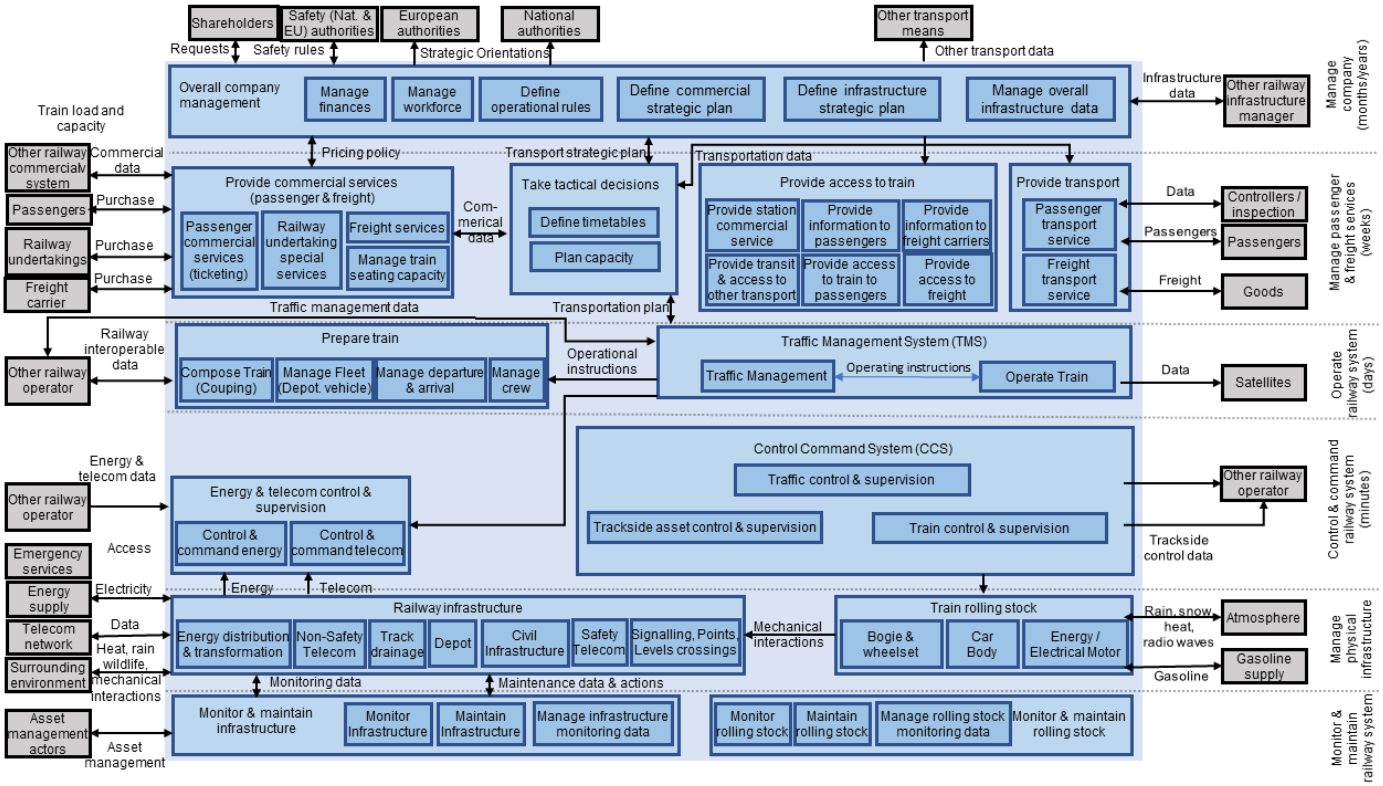


Figure 2. The high level railway system architecture as summarized by (LINX4RAIL, 2022).

Already existing models don't need to be re-implemented for digital twin modeling but simply can be exported as so-called Functional Mock-up Units (FMUs). Since FMI as well considers the export of binaries, the intellectual property on the modeling content also can be protected. The associated stakeholder is more motivated to contribute its submodels and to support the **interoperability** and **composablity** of the digital twin. However as a disadvantage, FMI does not follow object oriented modeling concepts such as inheritance, which e.g. allows for the convenient extension of Modelica connectors or model interfaces of the automotive VIL. So the flexibility to implement different levels of modeling details reflected by associated interfaces is restricted.

- Fig. 2 presents an overview of the general railway system which has been generated in a predecessor project of MOTIONAL (LINX4RAIL, 2022). It is obvious that the railway vehicle architecture exemplified in Fig. 1 only covers a small area of the whole system. Functions such as traffic management, signaling, supervision, energy supply and so on are interrelated to the motion of the vehicle and increase the complexity. Therefore, it is assumed that not one and only but several interrelated modeling architectures are required to depict the whole railway system. This is why, use cases have been selected first in order to develop this architecture landscape step by step.

- There is another conceptual aspect of **modularisation** that appears in the light of Fig. 2: subsystem or component models should be distinguished from each other in such a way that each subsystem model can be provided by a single stakeholder, such as the infrastructure manager, the railway operator or the vehicle and equipment suppliers, etc. This way, the subsystem model reflects the expertise of the associated stakeholder, which in turn determines the content and conditions of the submodel provision.

- The digital twin concept is a very general one. Fig. 3, not at all meant to be exhaustive, illustrates aspects in which digital twins may differ in practise: they may address different phases of the product life cycle, cover specific components only or represent a complex model assembly, consider CAD geometry or take transient behavior into account, just reflect a stationary property or predict dynamic evolutions and so on. This high degree of individuality is a challenge for the endeavour to create a framework for complex digital twins in the railway field.

3.3 Interim Status

The objective of the digital twin activities in MOTIONAL is the development of a digital twin environment which facilities the modularity, interoperability and composability of complex digital twin assemblies originating from the railway system in Fig. 2. The work was initiated in 2023



Figure 3. Aspects in which digital twins may differ in practise according to (Stark et al., 2019)

by collecting and analyzing use cases, shortly summarized by the following list:

- Braking and traction of railway vehicles: Vehicle models such as in Fig. 1 and data from braking experiments are compared to verify and validate the brake system design.
- ERTMS[2] digital twin: On-board units of the train protection system are certified in hardware-in-the-loop laboratories.
- Pre-assessment of automatic train operation scenarios: The ERTMS digital twin environment above is extended for the sake of examination of future automatic train operations.
- Maintenance digital twin based on graph database: Data of past maintenance actions are analyzed in order to optimize maintenance scheduling and execution.
- Failure mode of point machines: The actuators of switches, so-called point machines are monitored for the purpose of optimizing their maintenance.
- Fault detection and diagnosis in the rail infrastructure: Multi-sensor data are fused and analysed in order to explore the AI-usage in infrastructure monitoring.
- Driver cabin HVAC[3] digital twin: HVAC sensor data and environmental conditions are analyzed to schedule maintenance.
- Station Asset Management Digital Twin: BIM[4] data of stations and IoT sensor data are fused.
- Bridge Asset Management Digital Twin: BIM and sensor data of bridges are used to monitor construction.

In parallel, a review of published activities in industry and in the automotive field together with EU Rail internal discussions provided additional ideas, arguments and aspects, which are reported in this section.

The ongoing implementation of the above use cases is stored in a GitLab repository hosted by the EU Rail and accessible by all EU Rail participants. In the long term, making the use case models accessible beyond the EU Rail network is desirable, but this still needs to be agreed by all contributors.

It turned out that the provision of and access to infrastructure data, including track topologies and geometries, signalling equipment, etc., is a prominent interrelation on which many use cases rely. This is therefore addressed in the introductory digital twin example presented in the following section.

As the above discussion and the cautious paper title suggest, the findings explained in this paper are merely the first stage of a longer journey. This point is reflected in the project description by demanding two validated digital twin uses case of Technology Readiness Level 5 (European Commission et al., 2017) for the project outcome until 2026, only. Further stages of this journey are covered in the Outlook section.

4 Introductory Digital Twin Example

4.1 ETCS Eurobalises

In Europe, many different and incompatible train protection systems are currently used by individual European Union (EU) member states. In order to improve interoperability, country-specific protection systems are to be replaced by the European Train Control System (ETCS). An essential part of ETCS are the so-called Eurobalises installed on the track, see Fig. 4 and (Winter, 2009).

An Eurobalise is a standardized electronic beacon system used to communicate vital information to trains. It may be understood as a smart rail road sign that "talks" directly to trains as they pass over it. These small, robust devices are mounted between the railway tracks and contain electronic circuits that transmit data to onboard train systems when a train's antenna passes above them. By default, the ID number and the geographic coordinates of the Balises are transmitted. Furthermore, they can tell a train important things like the maximum safe speed for the upcoming section of track, whether signals ahead are green or red, the distance to the next station, or warnings about track conditions.

[2]ERTMS: European Rail Traffic Management System
[3]HVAC: Heating, Ventilation, Air Conditioning
[4]BIM: Building Information Modeling

DOI 10.3384/ecp21891



Figure 4. ETCS Eurobalise installed on the track see https://en.wikipedia.org/wiki/European_Train_Control_System, CC BY-SA 4.0

This system is crucial for modern train safety and automation, allowing trains to automatically adjust their speed or apply brakes if necessary, while also enabling features like precise positioning and automated train control across different European countries that previously had incompatible railway signaling systems. In the following, this introductory example tackles the monitoring of the Balise installations during daily operation.

4.2 Use Case Description

The use case is intended to compare two different information, one from the real world object, one from the virtual object:

- The real train runs along a given track, receives messages from the Eurobalises which are passed over and logs these messages or even transmits them as a real time data stream.
- The virtual train runs along the same track as the real train and passes over Balises, however uses the track design information in simulation in order to record virtual messages as they are to be expected by design.
- An algorithm compares these two records and issues an alert if a mismatch is observed. This mismatch indicates a malfunction to the railway infrastructure manager, who then is in charge of introducing appropriate measures.

Note, the messages nominated as real world messages above, are still results of a simulation, only. This limitation is due to the current status of the project. It will be brought closer and closer to the later application of the true digital twin as the Technology Readiness Level increases in the course of the project.

4.3 Modeling of the railway infrastructure

When working with digital twins that represent parts of the railway system, the appropriate modeling of the infrastructure is an essential component. However, the modeling requirements differ considerably depending on the specific use case. An incomplete collection of these is listed below:

- For vehicle dynamics simulations, detailed knowledge of both track layout and irregularities is mandatory
- Breaking and traction scenarios require data such as the current track gradient, speed limits and friction conditions between wheel and rail
- Railway dispatchers are dependent on an enhanced description of the railway network. This includes signalling facilities such as Eurobalises already mentioned in Section 4.1

The applications just presented make it clear that many different requirements are imposed on the description of railway infrastructure. Therefore, it is difficult provide one model which covers all scenarios simultaneously. Even if it were possible, the model would be overloaded with extensive data making it unnecessarily complex.

Therefore, it is proposed to provide a basic track model, as already exists in the Railway Dynamics Library (Heckmann et al., 2019). It only provides a rudimentary description of the track layout. The 3D-curve of the track contained in this model consists of B-Splines which are defined by supporting points imported from a track file.

If further demands are put on the description of the infrastructure, a more comprehensive model can be derived from the basic track model through inheritance. An example of this approach is shown in List. 1. The code snippet outlines the *TrackBalises*-model. Note that the corresponding graphical representation is shown in Fig. 5. It extends the features of the basic track model by outputting the ID number of the last crossed Balise based on the current vehicle position.

Listing 1. Track model extended with data on installed Eurobalises along the track

```
within RailwayDynamics.General;
model TrackBalises
  extends Track;
  import Modelica.Blocks.Tables;
  import Modelica.Blocks.Types;
  import Modelica.Blocks.Interfaces;

  Tables.CombiTable1Ds lookupTable(
    fileName="balises.mat",
    smoothness=Types.Smoothness.
       ConstantSegments);
  Interfaces.RealOutput baliseID
    "ID of the last crossed balise";
  Interfaces.RealInput s
    "Vehicle position along track";
equation
  connect(baliseID, lookupTable.y);
  connect(s, lookupTable.u);
end TrackBalises;
```

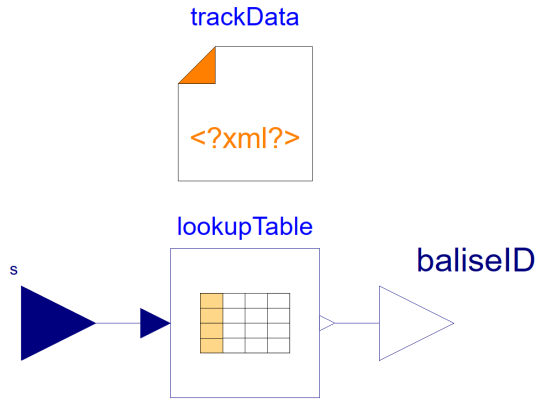


Figure 5. Graphical representation of the *TrackBalises*-model

4.4 Track description

In Section 4.3 a basic track model was introduced that provides a fundamental infrastructure description via supporting points imported from a file. In this paper we focus on a XML-based track description. In practice, however, there are many country-specific formats. It is therefore planned to support a more general approach for future work using ontologies, see (Camarazo et al., 2025).

In the XML-based data format the railway network is described as a graph in which nodes are connected by links. In the following, links are to be considered as track sections. A generic representation of a railway network is presented in Fig. 6.

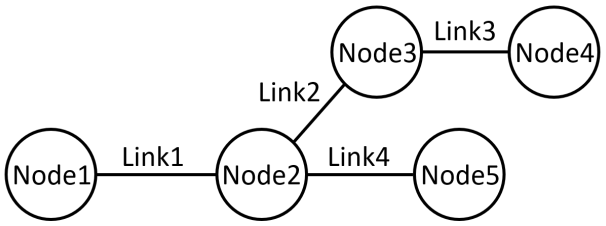


Figure 6. Graph of a generic railway network, which consists of nodes and links

The corresponding textual description of the railway network is shown in List. 2. It contains a 'Link'-element representing a track section, which is identified by the tag 'GlobalId'. Its start and end points are defined by 'FromNodeId' and 'ToNodeId', respectively, which are in turn identifiers of the corresponding 'Node'-elements. Other data included are section length and geographic coordinates of the supporting points which are used for interpolation as already mentioned in Section 4.3.

In addition to the pure track description, the track file contains data on Eurobalises installed on the track. Note that they are referred to in the file with the generalised term 'InformationPoint'. In our case, only data about the ID of the Balise and the distance travelled from the start point of the track section to the Balise are relevant.

The import of supporting points describing the track layout as well as corresponding Balise data is carried out using the ExternData library (Beutlich and Winkler, 2021). The imported data is stored in different arrays for further processing.

```
<?xml version="1.0" encoding="utf-8"?>
<BaseRoot>
  <Infrastructure>
    <Links>
      <Link>
        <Identification>
          <GlobalId>Link1</GlobalId>
          <FromNodeId>Node1</FromNodeId>
          <ToNodeId>Node2</ToNodeId>
        </Identification>
        <Length>TrackLength</Length>
        <Coordinates>
          <Coordinate>Coordinates1</Coordinate>
          <Coordinate>Coordinates2</Coordinate>
          <Coordinate>Coordinates3</Coordinate>
        </Coordinates>
      </Link>
    </Links>
    <Nodes>
      <Node>
        <GlobalId>Node1</GlobalId>
        <Coordinate>Coordinates4</Coordinate>
      </Node>
      <Node>
        <GlobalId>Node2</GlobalId>
        <Coordinate>Coordinates5</Coordinate>
      </Node>
    </Nodes>
  </Infrastructure>
  <InformationPoints>
    <InformationPoint>
      <Identification>
        <GlobalId>Point1</GlobalId>
        <FromNodeId>Node1</FromNodeId>
        <ToNodeId>Node2</ToNodeId>
      </Identification>
      <DistanceFromNode>Dist</DistanceFromNode>
      <Coordinate>Coordinates6</Coordinate>
      <BaliseGroup>
        <Balise>
          <GlobalId>BaliseID</GlobalId>
          <Data>BaliseData</Data>
        </Balise>
      </BaliseGroup>
    </InformationPoint>
  </InformationPoints>
</BaseRoot>
```

Listing 2. Textual description of a generic railway network

4.5 Simulation environment and FMU export

The use case outlined in Section 4.2 shall be supported by the usage of digital twins. The system presented in Fig. 7 is used for this purpose. Note that the simulation environment was created with Simulink (The MathWorks Inc., 2024), however, the individual components are based on the Modelica language and are exported as FMUs for further usage. This was done to emphasise the interoperability of the models used.

The system consists of the following components:

- The signals that come from a *real train* are summarised in one component. While running along a

DOI 10.3384/ecp21891

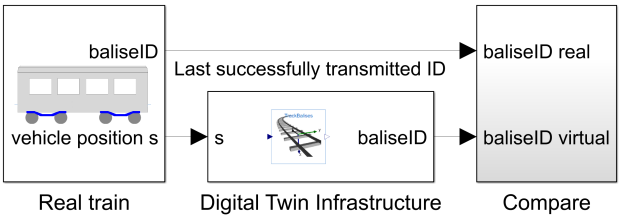


Figure 7. Illustration of the Digital Twin environment for the present use case

given track, the last successfully Balise ID number transmitted to onboard train systems is output. Furthermore, the current train position *s* is issued to the simulation environment. As already mentioned in Section 4.2, the messages nominated as real world messages, are still results of a simulation, only.

- The *TrackBalises*-model introduced in Section 4.3 takes on the role of the *digital twin of the infrastructure*. It takes the real train position signal as input and, based on imported track data as described in Section 4.4, outputs the ID number of the last crossed Balise. After exporting this model as FMU, a string parameter provides the path to a file from which track data is imported. Furthermore, since the XML-file format describes a whole railway network and not just one track, another string parameter has to be provided which represents the identifier of a specific track section. Besides XML-based description, it is currently possible to read track data in *.trm file format as used in Simpack (Dassault Systèmes, 2024).
- The task of the third component is to *compare* the signal from the real train with that of the digital twin. A graphical representation of the preliminary underlying model is presented in Fig. 8. The ID numbers of all Balises are integers. As soon as the difference between the two signals is greater than or equal to 1, it is assumed that the ID numbers do not match.

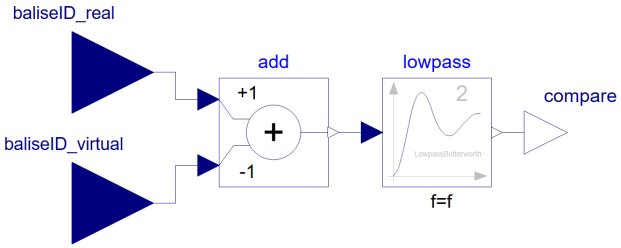


Figure 8. Graphical illustration of the *Compare*-model

The Balise ID numbers as signal outputs of both real train and digital twin come from different sources, but should be identical. As soon as a deviation occurs, this might indicate that a Balise did not transmit any data.

Note that the signals being processed in the *Compare*-model require low pass filtering. This is necessary in order to filter out high-frequency signals that might occur if the train onboard systems and FMUs shown in Fig. 7 use different sampling rates. Furthermore, it is worth to mention that the *Compare*-model presented in Fig. 8 should be considered as initial proposal. Other possible error sources such as inaccurate positioning data or time delays are not taken into account yet.

4.6 Results

An exemplary output generated in the context of the Eurobalise use case is presented in Fig. 9. The current position of the real train is shown in the top plot. The subplots in the centre show the ID number of the last crossed Balise issued by the train onboard system and the digital twin, respectively. Both signals are kept constant until the next Balise is crossed. The bottom plot shows the output of the *Compare*-model. It is clearly visible that there is a mismatch of the recorded Balise ID numbers at track position 5500 m. This indicates that a Balise did not transmit data to the train correctly and therefore should be revised.

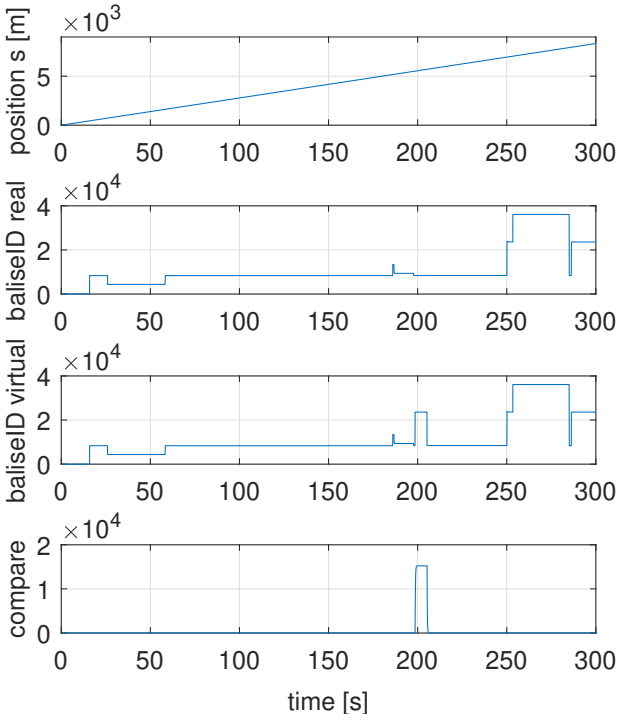


Figure 9. Exemplary history data that shows which Balise was last crossed

To make the comparison presented in Fig. 9 possible, the vehicle position along a specific track section of the digital twin and the actual train must coincide. For this purpose, the first successful transmission between Eurobalise and vehicle can be used for initialization.

The current work is dedicated to the elaboration of the use cases described in Sec. 3.3. Assuming that work can be continued in the next EU Rail funding period, a further improvement in the Technology Readiness Level and the number of validated use cases will certainly be sought.

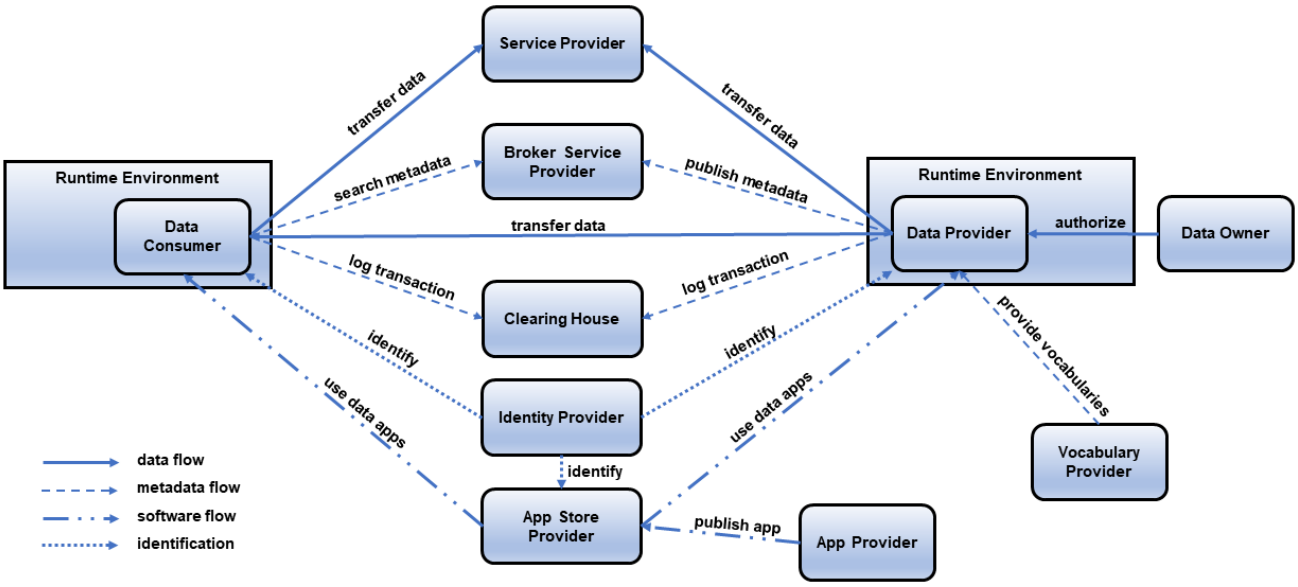


Figure 10. Reference Architecture of the IDS, cp. (Bader et al., 2020)

5 Outlook

As mentioned in Sec. 1, digital data, services and infrastructure are high level objectives of EU Rail. This is why, a parallel activity in MOTIONAL deals with the creation of a Federated Data Space for Rail. Data Spaces based on the Gaia-X[5] principles such as interoperability of data and services, sovereignty over data, security and trust use a concept of data integration without a central data repository. Data remains at its original source and is only exchanged peer-to-peer, if the transfer is explicitly requested and authorised. A data space, formed by the totality of its participants, is a distributed, federated structure with many data providers, consumers and services. The following enumerates, literally quoted from (Kraemer et al., 2023), the expected benefits related to Gaia-X business models:

- Efficiency: data is only exchanged when and to the extent necessary.
- Security: Data is protected from unwanted access.
- Control: Participants can determine exactly what access others have to their data.
- Transparency: Participants are given the necessary knowledge to decide for or against an exchange.
- Quality: Data and services are insured to be of the specified quality.
- Scalability: Required resources can be accessed as needed, quickly and comprehensively.
- Regulatory compliance: The Gaia-X ecosystem is in line with the values and laws of the European Union.

The concept has its main origin in the work of the International Data Space initiative (IDS)[6] (Bader et al., 2020). Fig. 10 presents the reference architecture of the IDS, which also considers the exchange of software tools and algorithms, addressed as applications or apps in Fig. 10.

In our context, the role of a digital twin developer corresponds to the role of an *App Provider* in terms of the IDS. Future digital twin activities in EU Rail will deal with the realization of this concept, which may be interpreted as a future business case for digital twins.

Acknowledgements

The authors gratefully acknowledge the financial support from Europe's Rail Flagship Project 1, MOTIONAL, grant agreement ID 101101973, HORIZON-ER-JU-2022-01.

Disclaimer

Although funded by the EU, views and opinions expressed are however those of the authors only and do not necessarily reflect those of the EU or EU Rail. Neither the EU nor the granting authority can be held responsible for them.

References

S. Bader, J. Pullmann, C. Mader, S. Tramp, C. Quix, A. Müller, H. Akyürek, M. Böckmann, B. Imbusch, J. Lipp, et al. The international data spaces information model–an ontology for sovereign exchange of digital content. In *International Semantic Web Conference*, pages 176–192. Springer, 2020.

Thomas Beutlich and Dietmar Winkler. Efficient Parameterization of Modelica Models. In Martin Sjölund, Lena Buffoni, Adrian Pop, and Lennart Ochel, editors, *Proceedings of the 14th International Modelica Conference*, number 181 in Linköping Electronic Conference Proceedings, pages 141–146. Modelica Association and Linköping University Elec-

[5] https://docs.gaia-x.eu/

[6] https://internationaldataspaces.org

tronic Press, September 2021. ISBN 978-91-7929-027-6. doi:10.3384/ecp21181141.

David Camarazo, Ana Roxin, and Mohammed Lalou. Railway systems' ontologies: A literature review and an alignment proposal. In Pari Delir Haghighi, Michal Greguš, Gabriele Kotsis, and Ismail Khalil, editors, *Information Integration and Web Intelligence*, volume 15342 of *Lecture Notes in Computer Science*, pages 311–319. Springer Nature Switzerland, Cham, 2025. ISBN 978-3-031-78089-9. doi:10.1007/978-3-031-78090-5_27.

Council of European Union. Council regulation (EU) 2021/2085 of 19 november 2021 establishing the joint undertakings under horizon europe, 2021. https://eur-lex.europa.eu/eli/reg/2021/2085/oj/eng.

Dassault Systèmes. Simpack documentation, 2024.

M. Dempsey, M. Gäfvert, P. Harman, C. Kral, M. Otter, and P. Treffinger. Coordinated automotive libraries for vehicle system modelling. In *Proceedings*, pages 33–41, 2006. URL https://elib.dlr.de/47220/.

European Commission, Directorate-General for Research, Innovation, C Strazza, N Olivieri, A De Rose, T Stevens, L Peeters, D Tawil-Jamault, and M Buna. *Technology readiness level – Guidance principles for renewable energy technologies – Final report*. Publications Office, 2017. doi:10.2777/863818.

Europe's Rail Joint Undertaking. Multi-Annual Work Programme, 2022. URL https://rail-research.europa.eu/wp-content/uploads/2022/03/EURAIL_MAWP_final.pdf.

D. Golightly, K. Pierce, R. Palacin, and C. Gamble. A feasibility assessment of multi-modelling approaches for rail decarbonisation systems simulation. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, 236(6):715–732, 2022. doi:10.1177/09544097211039395. URL https://doi.org/10.1177/09544097211039395.

A. Heckmann and S. Streit. The Modeling of Energy Flows in Railway Networks using XML-Infrastructure Data. In *9th International Modelica Conference*, pages 125–132, September 2012. URL https://elib.dlr.de/77216/.

A. Heckmann, G. Ehret, M.and Grether, A. Keck, D. Lüdicke, and C. Schwarz. Overview of the DLR Railway Dynamics Library. In *13th International Modelica Conference*, volume 157 of *Linköping Electronic Conference Proceedings*, pages 323–332. Linköping University Electronic Press, Linköpings universitet, 2019. URL https://elib.dlr.de/126755/.

P. Kraemer, C. Niebel, and A. Reiberg. Gaia-X and Business Models: Types and Examples, 2023. URL https://gaia-x-hub.de/wp-content/uploads/2023/11/GX-White-Paper-Business-Models.pdf.

O. Kugu, S. Zhou, R. Nowak, G. Müller, S. Reiterer, A. Meierhofer, S. Lachinger, L. Wurth, and M. Grafinger. An FMI- and SSP-based Model Integration Methodology for a Digital Twin Platform of a Holistic Railway Infrastructure System. In *Modelica Conferences*, pages 717–726, 2023.

O. Kugu, S. Zhou, S. Reiterer, M. Schwaiger, L. Wurth, and M. Grafinger. Pipeline-Based Automated Integration and Delivery Testing of Simulation Assets with FMI/SSP in a Railway Digital Twin. In *Modelica Conferences*, pages 189–198, 2024.

LINX4RAIL. Report on Supporting Work for whole Railway System high level Architecture, 2022. URL https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e502c59f1e&appId=PPGMS. Project funded under the Shift2Rail Joint Undertaking Grant Agreement Number 881826.

Modelica Asssociation Project FMI. Functional Mock-up Interface Specification Version 3.0.2, 2024. URL https://fmi-standard.org.

Modelica Asssociation Project SSP. System Structure and Parameterization Version 2ad515a2, 2024. URL https://ssp-standard.org.

RailML®. http://www.railml.org/, 2025. [Online; accessed 5 March 2025].

M. Singh, E. Fuenmayor, E. P Hinchy, Y. Qiao, N. Murray, and D. Devine. Digital twin: Origin to future. *Applied System Innovation*, 4(2):36, 2021.

M. Sjarov, T. Lechler, J. Fuchs, M. Brossog, F. Selmaier, A.and Faltus, T. Donhauser, and J. Franke. The digital twin concept in industry - a review and systematization. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 1789–1796, 2020. doi:10.1109/ETFA46521.2020.9212089.

R. Stark, C. Fresemann, and K. Lindow. Development and operation of digital twins for technical systems and services. *CIRP Annals*, 68(1):129–132, 2019.

The MathWorks Inc. Simulink version: 24.2 (r2024b), 2024.

Hendrik Van Der Valk, Hendrik Haße, Frederik Möller, and Boris Otto. Archetypes of digital twins. *Business & Information Systems Engineering*, pages 1–17, 2022.

M. Wiens, T. Meyer, and P. Thomas. The potential of FMI for the development of digital twins for large modular multi-domain systems. In *Modelica Conferences*, pages 235–240, 2021.

Peter Winter, editor. *Compendium on ERTMS, European Rail Traffic Management System*. DVV Media Group Eurailpress, Hamburg, 1. ed. edition, 2009. ISBN 978-3-7771-0396-9.

DOI
10.3384/ecp21891

Collaborative Digital Twin Development for Railway Braking and Traction Applications

Gerhard Hippmann[1] Blas Blanco Mula[2]

[1]Institute of Vehicle Concepts, German Aerospace Center (DLR), Germany, gerhard.hippmann@dlr.de
[2]CAF & CAF I+D, Spain, blas.blanco@caf.net

Abstract

The paper presents the collaborative development of a digital twin for railway braking and traction applications as part of the ERJU MOTIONAL project. The project's goal is to enhance the efficiency, user-friendliness, and maintainability of European railways through the application of digital twin concepts across nine distinct use cases. Contributing components are shared in a common library as Functional Mock-up Units (FMUs), enabling the creation of digital twins for various applications.

Specifically, this study focuses on the braking and traction use case for virtual validation purposes. The paper details the vehicle dynamics model, control units, and their interfaces, presenting simulation results under diverse adhesion conditions. Additionally, it highlights the potential benefits of digital twins in optimizing railway operations and monitoring system integrity, ultimately enhancing the reliability and efficiency of railway transportation.

Keywords: FMI, vehicle dynamics, railways, digital twin

1 Introduction

Europe's Rail Joint Undertaking (ERJU) (Europe's Rail Joint Undertaking 2022) aims to improve the European railway sector in a public-private partnership. The work presented in this paper is part of the ERJU MOTIONAL project which addresses digitisation of railways for enhanced efficiency, amenity, user-friendliness and maintainability. For this the concept of digital twins (DTs) is applied in nine different use cases. The basic idea is that the project partners contribute components as Functional Mockup Units (FMUs) (Modelica Asssociation Project FMI 2024) to a library which can be combined to DTs for various applications. General aspects of the DT development in EU rail such as use case portfolio, maturity or technology readiness level of the development towards true digital twins with real life objects, availability and dissemination of the models beyond the project and cybersecurity are presented in (Heckmann, Posseckert, and Adusumalli 2025).

This paper describes the current status of the braking and traction (B&T) use case for virtual validation purposes. The proper operation of railway vehicles during traction and braking under different environmental and topographical conditions is the result of the synchronised work of multiple subsystems with a high level of interconnection. Figure 1 shows a scheme of the different components involved in the B&T operation. Developing DTs for B&T is challenging since the actual architecture and number of components change among projects. On top of that, it is common that several providers are involved in the development of the components. These features cause a challenge for the correct integration and commissioning of the B&T system, involving, to a greater or lesser extent depending on the complexity of the project, track testing to finalise its adjustment and ensure correct operation.

In this context, the B&T use case is intended to provide methodologies for the seamless integration of models from different suppliers into a common DT. This would redound to a reduction of errors that need to be corrected during the integration and commissioning phase, with the resulting impact on the profitability of the project as a whole and compliance with delivery deadlines.

The remaining sections of this paper are structured as follows: In section 2 the multi-body model of the vehicle is presented. Section 3 describes the control units and their interfaces. The simulation and results verification of the combined system is presented in section 4. And section 5 relates the work to the digital twin concept. Finally, section 6 concludes the paper and gives an outlook to future plans.

2 Vehicle Dynamics Model

This section describes the models that comprehend most of the mechanical components involved in the B&T DT. As described in Figure 1, the vehicle dynamics model would concern the wheelset and train dynamics, the wheel-rail contact forces, the adhesion level and the track topology.

For the B&T use case, a two car multiple unit for passenger transport is regarded. The vehicle configuration resembles that of a light-rail vehicle-type, which is in the mid-way between tram and metro vehicle-types and whose maximum speed is commonly below 100 km/h. The car bodies are carried by three bogies, i.e. two motor bogies at the ends and a trailing Jacobs bogie at the centre of the vehicle. As usual for this kind of railway vehicles, two stage suspension systems are used to achieve optimal running dynamics and passenger comfort: Rather hard primary suspensions between wheelsets and

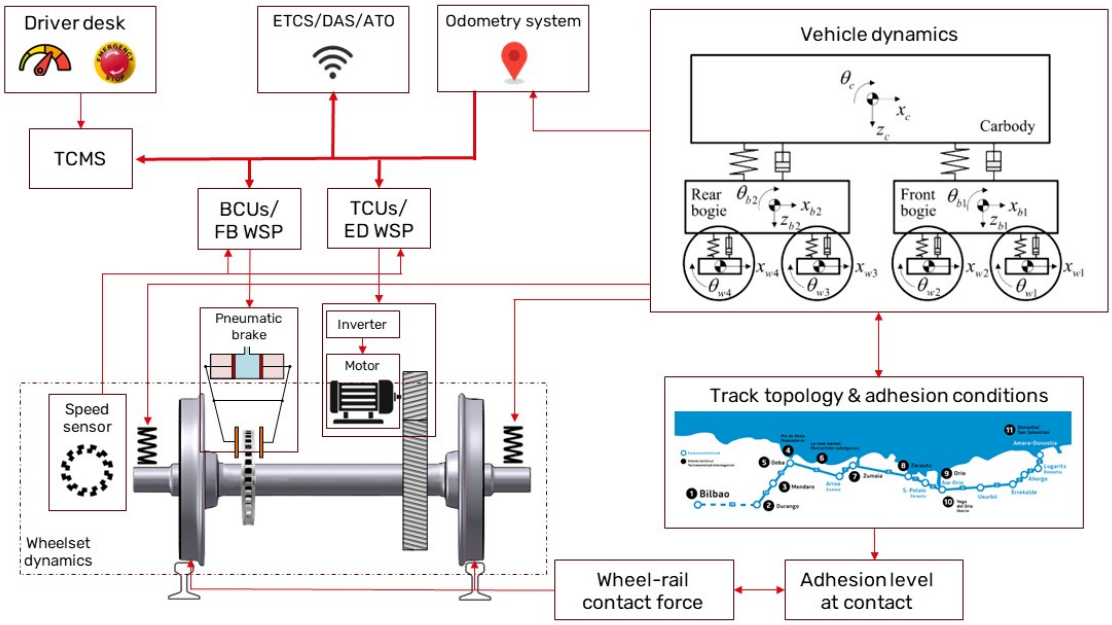


Figure 1. Scheme of the braking and traction system

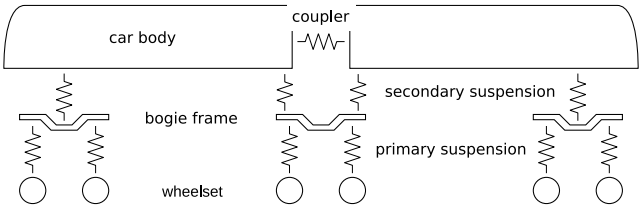


Figure 2. Multi-body structure of the multiple unit vehicle dynamics model.

bogie frames and softer secondary suspensions between bogie frames and car bodies. Completed by a coupler, this structure results in a multi-body system composed of eleven rigid bodies and eleven spring-damper force elements (Figure 2).

Modelica/Dymola using the DLR Railway Dynamics Library (RDL) (Heckmann, M. Ehret, et al. 2019) were chosen for modeling the vehicle dynamics model. In this way it was possible to take advantage of powerful FMU export and sufficient railway dynamics functionality and efficiency. Moreover this setup allowed us to implement the following modifications for the B&T use case.

For analysing B&T phenomena in the scope of this work, lateral, roll and yaw motion of the vehicle are not relevant. As a consequence, these degrees of freedom have been eliminated in the joints of all bodies of the system, resulting in half the number of mechanical dynamic states and accordingly improved computation efficiency. In principle, it would even be possible to consider only half of the vehicle, so that the computation effort of wheel-rail contacts is also halved. However, this has not been implemented so far.

Moreover, an extension of the RDL has been implemented to enable consideration of variable friction coefficients between wheel and rail during simulation. For this a scalar friction scaling factor f_μ has been introduced, which allows to modify the effective friction coefficient of the tangential force law. A common application is to define a lookup table function $f_\mu(s)$ dependent on the longitudinal track position s of the wheel.

The RDL implements different methods for the wheel-rail tangential force determination (Heckmann, Keck, et al. 2014). Since in B&T scenarios high creepage conditions can occur, Polach's method (Polach 2000) is utilised here. It allows to consider variable (e.g. dry/wet) wheel-rail conditions by modification of several parameters A, B, k_A, k_S and μ_0 (Polach 2005). However, so far we just apply the friction scaling factor to the maximum friction coefficient $\mu_0 := f_\mu \cdot \mu_0$ because this approximation is easy to handle and also applicable for other tangential force laws.

Figure 3 shows the block diagram of the vehicle dynamics model. Most of the components are instances of original or modified RDL classes. A linear spring-damper force element with preload has been implemented for sus-

DOI 10.3384/ecp218101

pensions and coupler. Velocity-dependent aerodynamic forces have been added to the RDL car body class. In addition a new brake and propulsion class has been created. It represents a frictional brake and an electric motor acting between bogie frames and wheelsets.

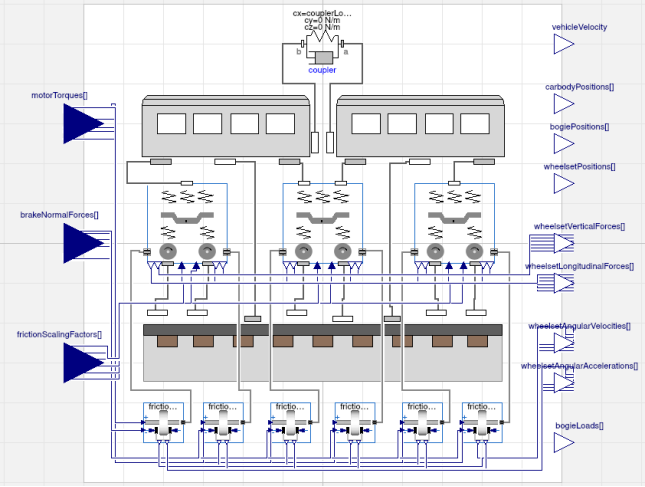


Figure 3. Block diagram of the multiple unit vehicle dynamics model.

At project start it was planned to implement the track as a separate model resp. FMU. However, this concept has been abandoned to avoid confusing and redundant modelling, because defining the track by model exchange would require eight identical track FMUs for all the track joints of car bodies and wheelsets. This is not necessary within Modelica, where the same track is defined as `outer` component of the track joints so that it can be evaluated like a global function. Nevertheless, the architecture of the RDL track implementation has been refactored to enable FMU export for other use cases (Heckmann, Posseckert, and Adusumalli 2025).

All initial conditions, tuneable parameters, inputs and outputs of the model are defined on top level for convenient FMU export. The initial conditions represent the track position and speed of the vehicle at simulation start. There are five scenario parameters, i.e. track data file, car occupancies and friction coefficients of wheel-rail and brake pad-disc, as well as 26 design parameters (mass properties, stiffness and damping values and wind resistance coefficients). Three vectorial input signals are implemented for motor torques, brake pad normal forces and friction scaling factors of the six wheelsets. And finally there are nine output signals, i.e. vehicle speed, track positions of the car bodies, bogies and wheelsets, angular velocities and accelerations of the wheelsets, secondary suspension loads of the bogies and vertical and longitudinal wheel-rail forces of the wheelsets. In the whole these variables form an interface specified for B&T control applications.

A simple test scenario has been used for debugging and testing the model. The vehicle starts at a speed of 10 m/s. From 5 to 10 s of simulation time, a trapezoidal motor

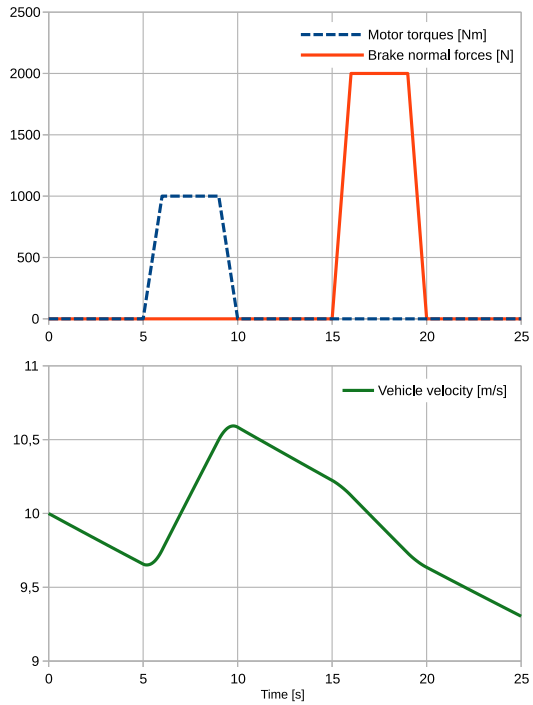


Figure 4. Motor torques, brake pad normal forces and resulting vehicle speed of the test scenario.

torque of 1000 Nm is inputted to the wheelsets. Analogously, a brake normal force of 2000 N is applied from 15 to 20 s. In Figure 4 the inputs and the resulting longitudinal velocity of the vehicle are plotted. During phases without driving torques, the speed decreases slowly due to air resistance forces. When the motor torques are active, the vehicle accelerates, and during braking deceleration increases.

3 Control Model

The control module mainly concerns the description of the logic involved in the B&T operation. Three main functionalities are due to the control module:

- Setting the effort command and operation mode by interpreting the manipulator signals.
- During braking, sharing out the efforts in the most suitable way among the electrodynamic (ED) and friction brake (FB) actuators. This functionality is known as 'blending'.
- To prevent that effort demand exceeds the available adhesion, which involves that maximum admissible longitudinal contact force is about $\mu_0 N$, where N is the normal contact force. This functionality is known as WSP, which stands for wheel-slide/slip protection depending on whether the vehicle is braking or gaining traction, respectively.

As regard to Figure 1, the components concerned by the control module are Traction and Brake Control Units

(TCUs and BCUs, respectively) and Train Control Management System (TCMS). The ETCS (European Train Control System), DAS (Driver Advisory System), ATO (Automatic Train Operation) and the odometry system are not considered in this DT, since they are not core elements of the B&T operation, however, their implementation based on an FMU approach is feasible. The driver desk is highly simplified.

The control architecture is shown in Figure 5. It consists of two TCUs placed at the motor bogies, and three BCUs, one per bogie. The TCMS recieves the driver command and sets the data bus of the vehicle from which the TCUs and BCUs receive the command, read the state of other components and publish their own data. Data communications are compacted in buses signals, which allows to define as many variables as necessary with different data types and sizes without changing the model structure.

In the following subsections, the minimal units describing TCMS, TCUs and BCUs are detailed and their output buses defined. These units are coded in Simulink and subsequently exported as standalone FMUs. The models highly simplify both functionality and included number of variables. However, the overall concept and approach remains valid and further development can effortlessly include more sophisticated or even the actual onboard code.

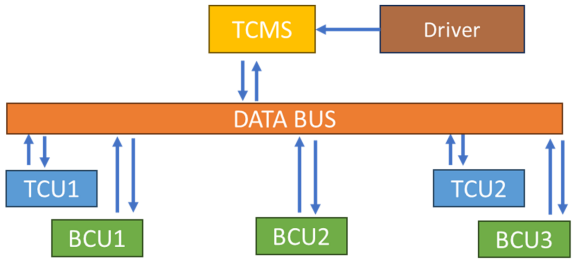


Figure 5. Control architecture of the vehicle

3.1 TCU

TCUs (Traction Control Units) are responsible for controlling the ED actuators (motor) for both ED braking or traction. Moreover, they implement the ED WSP logic. Each TCU publishes a data bus containing four variables included in Table 1.

The TCUs receive the traction or braking demand from the TCMS and calculate how much ED effort they can perform taking into account two main limitations: the effort vs. speed curve of the motor and the available adhesion due to the contact conditions and the normal loads on contact. The resulting effort is the so-called EffANoWSP, which is the effort before WSP control.

Subsequently, they measure the angular velocity of the wheels and, by comparing it with the vehicle speed, estimate whether wheel sliding/slipping is occurring due to degraded adhesion conditions and, in that case, they correct effort. The effort to be applied after WSP correction is denoted as EffA.

Variable name	Unit	Description
EffANoWsp	[N]	Bogie effort before WSP
EffA	[N]	Bogie effort after WSP
b_WSP	[Boolean]	WSP control is acting
b_ED_Ok	[Boolean]	ED is operative

Table 1. TCU data bus variables

3.2 BCU

BCUs (Brake Control Units) are responsible for controlling the FB actuators. Moreover, they implement the pneumatic (or hydraulic, depending on the FB pressure system) WSP logic.

BCUs may act only during braking. The blending strategy seeks to prioritize the ED effort to maximize energy recovery and BCUs may have a small contribution to the total braking effort when adhesion conditions are good in normal braking modes. However, under degraded adhesion conditions, emergency brake mode, or in the presence of failing TCUs, the BCUs will play an important role in the braking operation.

BCUs in trailing bogies are expected to act anytime adhesion conditions are poor due to the TCMS activating low-adhesion mode. However, the blending strategy normally minimizes the actuation of BCUs in motor bogies.

The BCUs read the braking effort demanded by the TCMS and the effort performed by the TCUs and, accordingly, they calculate the braking effort deficit. By measuring the normal load on each bogie they estimate the available adhesion and share the effort among BCUs. In the studied vehicle, only the BCU in the trailing bogie will normally act unless some of the TCUs are failing.

Finally, BCUs also read the wheelset angular velocities and by comparing them with the vehicle speed they decide whether WSP correction is to be applied for avoiding the wheel blockage, which would result in wheel flats. The variables published by the BCU in the BCU data bus are included in Table 2.

Variable name	Unit	Description
WEIGHT	[kg]	Weight on the bogie
FB_REQ	[N]	Requested FB effort
FB_APP_WSP	[N]	FB effort after WSP
FB_AVAIL	[N]	Available FB effort
b_WSP	[Boolean]	WSP is acting
b_FB_Ok	[Boolean]	FB is operative

Table 2. BCU data bus variables

DOI
10.3384/ecp218101

Variable name	Unit	Size	Description
TRC_BRK_DEM	[Integer]	(1×1)	Traction/Braking demand
NB_ED_AVAILABLE	[Integer]	(1×1)	Number of TCUs available
NB_FR_AVAILABLE	[Integer]	(1×1)	Number of BCUs available
TRAIN_WEIGHT	[kg]	(1×1)	Weight of the train
TRAIN_WEIGHT_ROT	[kg]	(1×1)	Weight of the train including rotational mass
SPEED	[m/s]	(1×1)	Vehicle speed
v_EffA	[N]	(1× TCUs)	Applied effort per TCU
v_EDOk	[Boolean]	(1× TCUs)	TCUs availability
v_EffANoWSP	[N]	(1× TCUs)	Applied effort per TCU before WSP
v_ED_WSP	[Boolean]	(1× TCUs)	WSP limiting ED effort per TCU
v_FBR	[N]	(1× BCUs)	Requested FB effort per BCU
v_FBA	[N]	(1× BCUs)	Applied FB effort per BCU
v_BOGIE_WEIGHT	[kg]	(1× BCUs)	Vertical load per BCU
v_FBAVAIL	[N]	(1× BCUs)	FB available effort per BCU
v_FBOk	[Boolean]	(1× BCUs)	BCUs availability

Table 3. Vehicle data bus variables

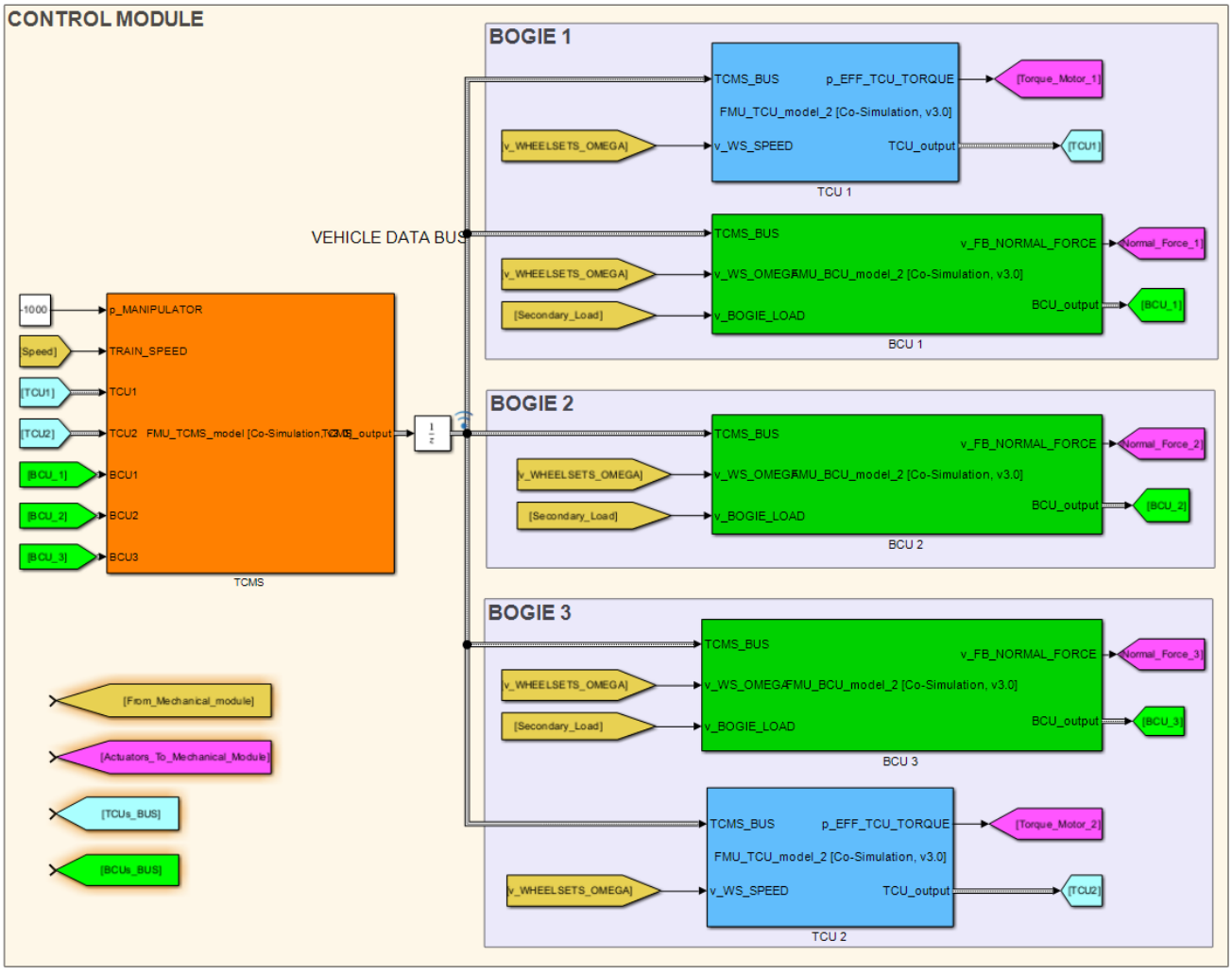


Figure 6. Control module based on FMUs

3.3 TCMS

TCMS (Train Control Management System) is the backbone of the train regarding its logic and electronics. To what concern B&T operation, it is responsible for managing the signals of different components into the vehicle data bus and performing functionalities such as speed es-

timation, setting effort commands, or defining operation mode, among others.

The TCMS receives the output from each TCU and BCU (as indicated in Tables 1 and 2) and updates the vehicle data bus, which is a unified data source from which any component reads all the necessary information to perform its operation. The vehicle data bus contains the variables listed in Table 3.

3.4 FMU-Based Control Model

The control model entirely based on FMUs is shown in Figure 6, which replicates the architecture of that shown in Figure 5. There are only three types of FMU: TCMS, TCU, and BCU types. The difference among TCUs and BCUs resides in the input parameters that identify their location in the vehicle. All TCUs and BCUs include a parameter to define the bogie that they belong to (p_BOGIE_INDEX). Moreover, the BCUs include another parameter to indicate that they belong to a motor bogie and the related TCU when it is defined as a non-zero value (p_TCU_INDEX). By using the latter, the BCUs can track the state of the TCUs and take the correct input data to perform blending from the vehicle data bus.

Figure 6 also includes a legend to indicate the type of signals. Yellow labeled tags are signals coming from the vehicle dynamics model: the speed, the wheelsets' angular velocities, and the load on the secondary suspension. By using the parameter p_BOGIE_INDEX, the TCUs and BCUs can select the correct values of the wheelsets' angular velocities and the load on the secondary suspension. The vehicle data bus is read by all the TCUs and BCUs from where they obtain the necessary information to set effort values (labeled in clear pink), which are subsequently sent to the vehicle dynamics model. Each TCU and BCU output a data bus that is sent to the TCMS that merges all the data to update the vehicle data bus accordingly. As mentioned before, the signals denoted as TCU_X, BCU_X and TCMS_BUS, are programmed as bus signals, that enable a flexible definition of variables. This makes possible to redefine interfaces avoiding changes in the control module.

4 Simulation and Verification

Vehicle dynamics and control models described in Sections 2 and 3, respectively, are coupled together. Figure 7 shows how the FMU of the vehicle dynamics model is coupled with the control module shown in Figure 6. The correct simulation of the model is checked by assessing its behaviour for different adhesion conditions, dry ($\mu_0 = 0.30$), wet ($\mu_0 = 0.10$), and contaminated surface ($\mu_0 = 0.05$) (Gallardo-Hernandez and Lewis 2008). Adhesion values, μ_0, are defined in the vehicle dynamics model as detailed in Section 2.

Figures 8 and 9 show the kinematic and effort results of the simulations, respectively. The simulation is a service braking operation from 80 km/h, which is about its maximum design speed to 0 km/h. In dry conditions, the braking distance is about 220 m. For wet conditions, the braking distance increases up to 260 m, which is 18% higher than for dry conditions. In this case, the vehicle is not able to sustain the maximum deceleration given by the dry

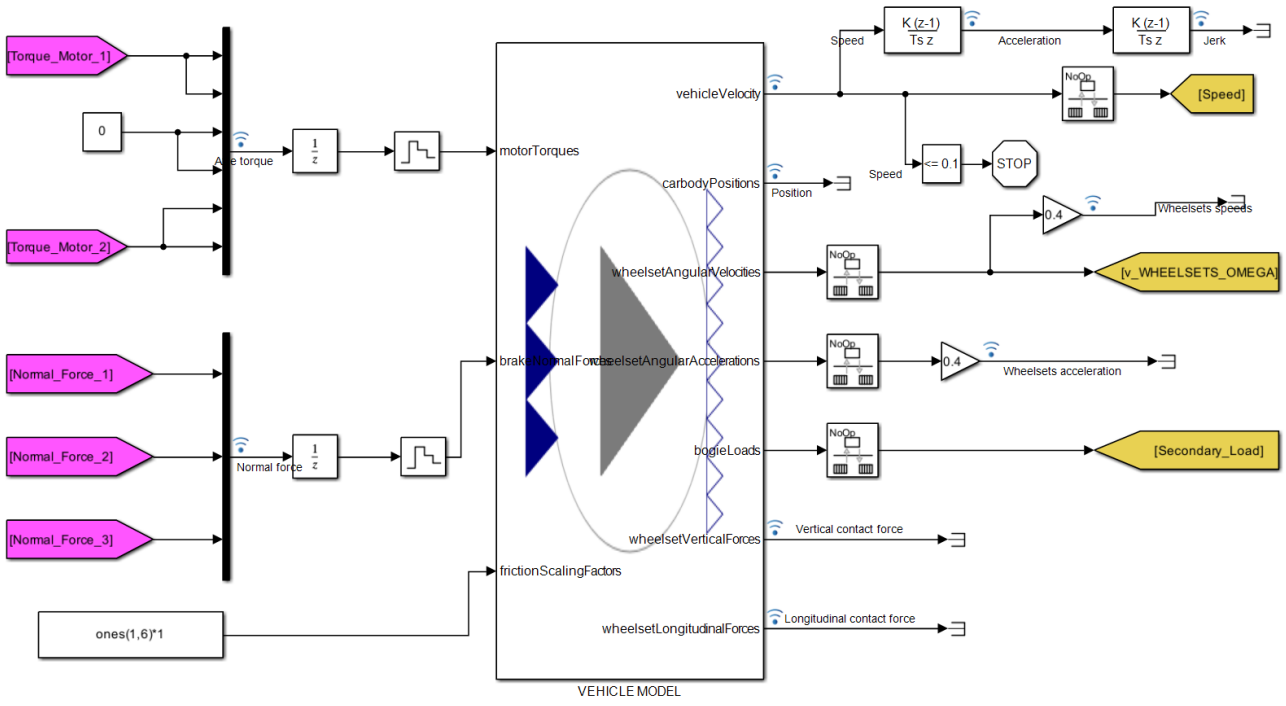


Figure 7. Vehicle module implemented along with the control module interfaces

DOI 10.3384/ecp218101

scenario, and it is reduced as a consequence of the WSP control. For very poor adhesion conditions, the braking distance increases up to 600 m. This value is significantly high and may not be acceptable in a real vehicle. Therefore, the definition of more aggressive braking modes as well as the modelling of rail cleaning phenomenon may be necessary to achieve acceptable braking distances in this scenario.

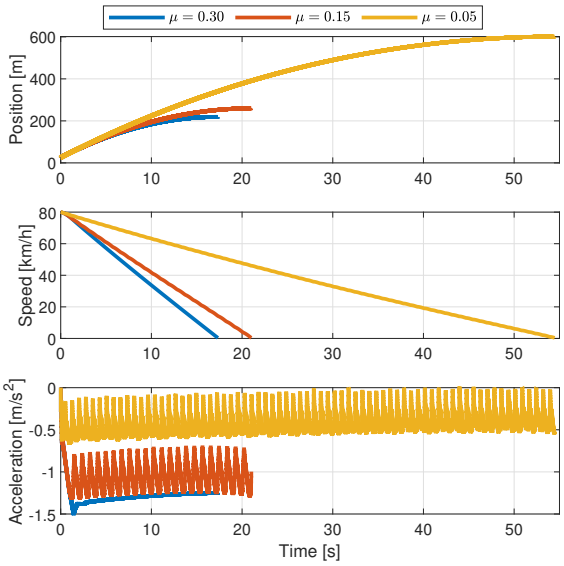


Figure 8. Vehicle kinematic results for different adhesion coefficients

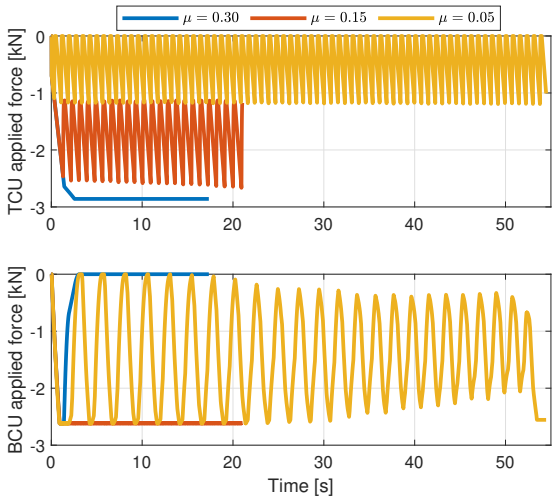


Figure 9. TCU and BCU applied efforts for the first and second bogie, respectively, at different adhesion conditions.

Regarding effort results shown in Figure 9, in the dry case, TCUs are able to provide all the braking deceleration except at the braking start. In the first seconds, the jerk-limitation prevents TCUs from increasing too quickly the effort and the BCU at the trailing bogie compensates for the braking effort deficit. Once the TCUs reach the requested effort, the BCU releases friction effort. For wet conditions, the TCUs start to limit the applied effort via WSP, while the BCU in the trailing bogie applies a constant effort to compensate for the braking deficit in the TCUs. For very poor adhesion conditions nor TCUs or BCUs are able to sustain the effort and both ED and pneumatic WSP start to limit the application of braking efforts. The ED effort is able to release and apply effort much faster than the pneumatic one.

The application of cyclic efforts due to the WSP is directly impacting on the wheelset speeds. Figure 10 shows the vehicle speed along with the linear speed of the wheelset, calculated as the angular velocity times the wheel radius, for the very poor adhesion conditions. As a consequence of the cyclic application of effort due to the WSP control under low adhesion conditions, the wheelset speed oscillates. When the effort is maximum the wheelset speed is minimum, while for close to zero efforts the wheelset speed approaches that of the vehicle. This shows how the WSP control is preventing the wheel from getting blocked, that would result in severe wheel damage.

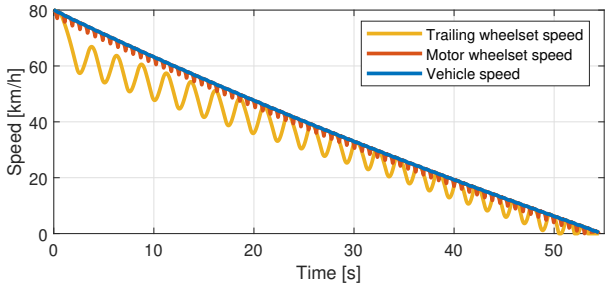


Figure 10. Wheelset speeds and vehicle speed comparison for the braking simulation with $\mu = 0.05$.

5 Digital Twin Context

Further to classical simulation applications as presented in section 4, the DT concept allows for comparison of the real world system with its digital model. Such evaluation can be used to enable improved operation as well as to monitor the current system integrity.

5.1 Improved Operation

In railway signaling, modern communications-based train protection systems like the European Train Control System (ETCS) (Wikipedia 2025) enable to run railway vehicles in *moving block* operation, where the minimum allowed distance between trains can be determined in real time. In this context, a B&T DT allows to estimate the current friction coefficient between wheels and rails (Schwarz 2021), which has significant influence on the possible braking distance (Marc Ehret 2024), and thus allows to determine optimal moving block lengths required between trains. In this way the utilisation of railway infrastructure can be increased, resulting in improved line capacity or safety.

5.2 Monitoring

Another possibility for taking advantage of a B&T DT is rolling stock monitoring. A contemporary approach is to compare sensor measurements of the real world vehicle with expected signals of the DT model using artificial intelligence methods. In this way it is possible to optimise maintenance intervals and to avoid damage and failure of railway vehicles. Hence, economic efficiency and reliability of railway transport can be improved.

6 Conclusions and Outlook

In this paper, collaborative DT development for railway braking and traction applications is presented. The developed DT clearly distinguishes two main domains, on the one hand, the vehicle dynamics involved in the braking and traction operation, and on the other hand, the elements performing the operation control. It is demonstrated that the basic concept of partners developing system modules with different tools as FMUs for DT applications can be applied successfully. Of course, this requires elaborate, iterative specification of interfaces and tuneable parameters.

Regarding the control domain, a modular structure of the control part recreating its real architecture is developed using FMUs basic units of the TCUs, BCUs, and TCMS. Communications among these components are represented via data buses that enable a versatile and compact definition of transferred data. These features are necessary to perform virtual validation and certification via substituting one or several basic units of the control module with the real unit under test.

The vehicle dynamics model successfully couples with the control module and provides it with any necessary physical signals to perform the control operation. Furthermore, this model has been proven to simulate low-adhesion conditions correctly. As a result, the B&T DT shows stable behaviour in degraded conditions such as low-adhesion scenarios. The control module activates WSP logic for both the ED and FB efforts, thus preventing wheel blockage during braking. This proves the feasibility of performing advanced virtual verification and validation of systems under extreme scenarios whose conditions are difficult to replicate via on-track tests.

In future work, the presented B&T DT is to be validated using real measurements that will give an insight of the accuracy level of the presented approach. Moreover, some other modules can be added as FMUs to increase the DT representativity, e.g., modules representing the track and ECTS/DAS/ATO.

Acknowledgements

This work has been supported by Europe's Rail Flagship Project 1, Motional (grant agreement ID 101101973, HORIZON-ER-JU-2022-01).

Disclaimer

Although funded by the European Union (EU), views and opinions expressed are however those of the authors only and do not necessarily reflect those of the EU or ERJU. Neither the EU nor the granting authority can be held responsible for them.

References

Ehret, Marc (2024). "Deterministic and stochastic braking distance prediction of disc-braked rail vehicles". PhD thesis. Technische Universität Berlin. URL: https://elib.dlr.de/203863/.

Europe's Rail Joint Undertaking (2022). *Multi-Annual Work Programme*. DOI: 10.3030/881826. URL: https://rail-research.europa.eu/wp-content/uploads/2022/03/EURAIL_MAWP_final.pdf.

Gallardo-Hernandez, EA and R Lewis (2008). "Twin disc assessment of wheel/rail adhesion". In: *Wear* 265.9-10, pp. 1309–1316.

Heckmann, A., M. Ehret, et al. (2019). "Overview of the DLR Railway Dynamics Library". In: *13th International Modelica Conference*. Vol. 157. Linköping Electronic Conference Proceedings. Linköping University Electronic Press, Linköpings universitet, pp. 323–332. URL: https://elib.dlr.de/126755/.

Heckmann, A., A. Keck, et al. (2014). "The Foundation of the DLR RailwayDynamics Library: the Wheel-Rail-Contact". In: *10th International Modelica Conference*. Linköping Electronic Conference Proceedings. Linköping University Electronic Press, Linköpings universitet, pp. 465–475. URL: https://elib.dlr.de/88522/.

Heckmann, A., A. Posseckert, and V.-B. Adusumalli (2025). "Aspects and Ideas for the FMI-based Modeling of Railway Digital Twins". In: *16th International Modelica FMI Conference*. Linköping Electronic Conference Proceedings. Lucerne, Switzerland: Linköping University Electronic Press, Linköpings universitet.

Modelica Asssociation Project FMI (2024). *Functional Mock-up Interface Specification Version 3.0.2*. URL: https://fmi-standard.org.

Polach, O. (2000). "A fast wheel-rail forces calculation computer code". In: *Vehicle System Dynamics* 33, pp. 728–739.

Polach, O. (2005). "Creep forces in simulations of traction vehicles running on adhesion limit". In: *Wear* 258.7, pp. 992–1000.

Schwarz, Christoph (2021). "Model-Based Estimation of Wheel/Rail- and Brake Disc/Pad-Friction for Railway Vehicles". PhD thesis. RWTH Aachen University. URL: https://elib.dlr.de/146309/.

Wikipedia (2025). *European Train Control System*. URL: https://en.wikipedia.org/wiki/European_Train_Control_System.

Diagnosing Newton's Solver Convergence Failures in the Initialization of Modelica Models

Francesco Casella[1] Bernhard Bachmann[2] Karim Abdelhak[2] Philip Hannebohm[2] Teus van der Stelt[3]

[1]Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy,
{francesco.casella}@polimi.it
[2]Faculty of Engineering and Mathematics, University of Applied Sciences Bielefeld, Germany,
{bernhard.bachmann,karim.abdelhak,philip.hannebohm}@hsbi.de
[3]Asimptote BV, The Netherlands, teus.vanderstelt@asimptote.com

Abstract

The convergence failure of iterative Newton solvers during the initialization of Modelica models is a serious show-stopper, particularly for inexperienced users. This paper presents the implementation in the OpenModelica tool of methods presented by two of the authors in a previous paper, to help diagnosing and resolving these convergence failure by providing ranked lists of potentially critical start attributes that might need to be fixed in order to successfully achieve convergence. The method also provides library developers with useful information about critical nonlinear equations, that could be replaced by equivalent, less nonlinear ones, or approximated by homotopy for more robust initialization.

Keywords: Initialization failure, Newton's method, Debugging

1 Introduction

The Modelica language (Mattsson, Elmqvist, and Otter 1998; Modelica Association 2023) has firmly established itself as a standard for system-level simulation of cyber-physical systems. The combination of declarative, equation-based modelling style and of modular, object-oriented composition of structured system models allows for flexible, convenient, and self-documented modelling of complex systems. Engineers can easily assemble system models using a drag-and-drop methaphor from libraries of reusable components, whereas domain experts can conveniently develop new component models by describing their behaviour through a-causal differential-algebraic equations. The solution of the overall system equation is entirely off-loaded from the modeller to the simulation tools, which use sophisticated symbolic and numerical methods to generate efficient and numerically robust code to solve the system equations and produce simulation results.

The price to pay for this flexibility and ease of use in building system models is that resulting system models often require the solution of implicit equations, both during initialization and during simulation. In some cases, implicit equations can be automatically solved in closed form using symbolic algorithms; however, in many cases, particularly for systems of non-trivial size, numerical methods must be employed.

The numerical solution of *linear* implicit equations poses no particular challenges, as long as the model is well-posed, so that the system is non-singular; automatic scaling techniques, see, e.g., (Casella and Braun 2017) can be used to improve the condition number of the A matrix of systems that are nearly singular because of bad scaling. Conversely, the numerical solution of *nonlinear* systems of equations requires the use of iterative methods, usually some kind of quasi-Newton algorithm, which can be problematic.

Iterative quasi-Newton methods require a suitable initial guess of the solution to get started; provided that the system Jacobian is non-singular in the sought-after solution, they provide fast convergence to the solution if the initial guess is close enough to it. Good quasi-Newton algorithms implement additional strategies such as damped steps, enforcing min-max limits on variables, etc., to enlarge the set of initial guesses that lead to successful convergence and to cope with poorly chosen initial guess values. Unfortunately, it is a sad fact of life that these algorithm sometimes fail to converge.

If convergence failures happen during *simulation*, where the values obtained at the previous time step (or some suitable extrapolation thereof) are used as initial guess, an effective strategy to cope with the failure is simply to retry with a shorter time step; as long as the variables of the problem are changing continuously, it is possible to get an initial guess which is arbitrarily close to the solution by picking a short enough time step, thus ensuring convergence.

The most critical scenario from a practical standpoint is the convergence failure during *initialization*, where `start` attributes are used as initial guess, because there is no easy backup strategy available in that case. If the initialization problem fails to converge to a meaningful solution, the model at hand is pretty useless.

The problem of convergence failure during initializa-

tion is compounded by the fact that the feedback provided to the modeller by Modelica tools is very low-level, compared to the nice, high-level, modular modeling style of the Modelica language. What one typically gets is an ugly log message from the solver, showing the selected initial guess values and some error message such as "The nonlinear solver failed to converge during initialization, please provide better start attributes". The question is: how is the poor end user supposed to figure them out?

A naïve strategy in this case is to inspect the list of start values that led to the failure, comparing them to one's best estimate of what their value at the solution should be; values which are obviously very far from the estimated solution can be improved one by one. However, this is a very frustrating and time-consuming strategy, particularly in the case of problems with dozens or hundreds of unknowns. For reasons that will become clear in Section 2, in some cases very large errors in the initial guess are not a problem for convergence, whereas comparatively small errors could be critical in other cases, so one may end up wasting a large amount of time fixing `start` attributes which are not critical at all. If the problem at hand has 100 unknowns, is it really necessary to fix all the 100 start attributes, or maybe fixing only a few ones will suffice to attain convergence? Answering to this question is the main goal of the methods discussed in this paper.

More specifically, based on results previously published by Casella and Bachmann (2021), this paper discusses how a Modelica tool can rank the initial guess values of failing nonlinear systems of equations, highlighting the ones which are more likely to be responsible for the convergence failure, thus helping the end user to succeed in solving the initialization problem and getting the simulation to run. It also shows how critical nonlinear equations can be identified, so that they can be, e.g., approximated by means of homotopy by the model developer. The proposed methods were implemented in the OpenModelica tool (Fritzson et al. 2020) and successfully tested against the test cases discussed in that paper.

The paper is structured as follows: Section 2 reviews the methods proposed by Casella and Bachmann (2021), introducing the notation and the main results; Section 3 discusses the implementation in the OpenModelica tool, while Section 4 shows some of the results obtained on some of the test cases discussed in (Casella and Bachmann 2021). Section 5 concludes the paper with final remarks and suggestions for future work.

2 Method

This section introduces the notation for the problem at hand and recalls the main results of (Casella and Bachmann 2021) which are relevant for the discussion.

2.1 Notation

Consider the problem

$$f(x) = 0, \tag{1}$$

where $x \in \mathbb{R}^m$ and $f : \mathbb{R}^m \to \mathbb{R}^m$ is a vector function which is continuously differentiable in an open neighbourhood $\mathfrak{D}$ of the solution $\bar{x}$, $f(\bar{x}) = 0$. In the context of this paper, problem (1) corresponds to a strong component of the initialization problem of a Modelica model, where x is a subset of the set of unknowns of such problem, namely state variables, their derivatives, algebraic variables, and unknown parameters with `fixed = false` attribute. In case homotopy-based initialization is used (Sielemann et al. 2011), the considered problem is the one at $\lambda = 0$, for which the intial guess is taken from the `start` attributes.

Denote the Jacobian matrix of function $f(x)$ with respect to x as $f_x(x)$. Assume the vector of the unknowns x is suitably ordered, so that it can be split into two subvectors $w \in \mathbb{R}^q$ and $z \in \mathbb{R}^{m-q}$

$$x = \begin{bmatrix} w \\ z \end{bmatrix}, \tag{2}$$

w being the smallest possible sub-set of x such that

$$f_x(x) = J(w), \tag{3}$$

i.e., the Jacobian matrix of $f(x)$ depends only on w and not on z. Thus, w are defined as the *nonlinear variables* of the problem, while z are defined as the *linear variables* of the problem.

Assume the equations in (1) are ordered so that $f(x)$ can be split into two vector functions $n(x)$ and $l(x)$, $n : \mathbb{R}^m \to \mathbb{R}^p$, $l : \mathbb{R}^m \to \mathbb{R}^{m-p}$

$$f(x) = \begin{bmatrix} n(x) \\ l(x) \end{bmatrix}, \tag{4}$$

where $n(x)$ contains the non-linear equation residuals and $l(x)$ contains the linear equation residuals. System (1) is thus split into *nonlinear equations* $n(x) = 0$ and *linear equations* $l(x) = 0$. The system (1) can then be rewritten as

$$f\left(\begin{bmatrix} w \\ z \end{bmatrix}\right) = g(w) + f_z z. \tag{5}$$

where $g(w)$ contains all the nonlinear terms of the system equations, while $f_z z$ denotes the linear part of the system equations.

The solution $\bar{x}$ can be computed iteratively by Newton-Raphson's method, which requires to solve the following linear equation at each iteration j

$$f_x(x_{j-1})(x_j - x_{j-1}) = -f(x_{j-1}), \qquad j = 1, 2, \cdots \tag{6}$$

starting from a given initial guess x_0, which is obtained from the `start` attributes of the unknowns x.

One very important remark is due when tearing is used to solve the implicit nonlinear system. In this case, the set of unknowns x is the set of *tearing* (or iteration) variables, whereas the set of equations (1) refers to the set of *residual* equations. In this case, determining which are the nonlinear variables and equations of the torn system requires to

DOI
10.3384/ecp218109

consider the dependencies between each tearing variable and each residual equation introduced by the torn equations, which may themselves be linear or nonlinear. For example, the system

$$\sin(a) + b = 1 \tag{7}$$
$$a + b + \cos(c) = 2 \tag{8}$$
$$a + b + c = 0 \tag{9}$$

can be solved by selecting a and c as tearing variables:

$$b := 1 - \sin(a) \tag{10}$$
$$a + b + \cos(c) - 2 = 0 \tag{11}$$
$$a + b + c = 0; \tag{12}$$

where the unknown vector of the torn system is $x = [a \quad c]'$. Although the last of the two residual equation looks linear at first glance, it actually depends non-linearly on a through the nonlinear torn equation. Therefore, both unknowns and both residual equations of the torn system are actually non-linear in this case.

2.2 Theoretical Background

This section recalls the main theoretical results of Casella and Bachmann (2021), putting them in the specific context of a Modelica tool solving a strong component of the initialization problem.

Theorem 1. ***(Superlinear convergence close to the solution)****. If the Jacobian $f_x(\bar{x})$ is non-singular in the solution $\bar{x}$ and Lipschitz-continuous in a neighbourhood of $\bar{x}$, for all x_0 sufficiently close to x, the sequence $\{x_j\}$ of the solutions of* (6) *converges not less than quadratically to $\bar{x}$.*

Theorem 2. ***(Convergence in the linear case)****. If Equation* (1) *is linear and f_x is non-singular, then Netwon's algorithm converges in one step, irrespective of the chosen initial guess x_0.*

Theorem 3. ***(Convergence in the mixed linear-nonlinear case)****. If Newton's algorithm is initialized with a first guess*

$$x_0 = \begin{bmatrix} w_0 \\ z_0 \end{bmatrix}, \tag{13}$$

the values of the approximated solution x_j at each step $j > 0$ only depend on the guess values of the nonlinear variables w_0, regardless of the choice of guess values of the linear variables z_0.

Theorem 4. ***(Linear residuals after the first iteration)****. The residuals of the linear equations in system* (1) *after the first iteration of Newton's algorithm are zero, i.e. $l(x_1) = 0$, regardless of the initial guess values x_0.*

Theorems 1 and 2 are well-known, whereas Theorems 3 and 4 were first stated by Casella and Bachmann (2021) and have two very important implications for the problem at hand:

1. only *nonlinear* equations $n(x) = 0$ and only initial guesses of *nonlinear* variables w_0 matter for convergence;

2. there is no reason to bother about `start` attributes of *linear* variables

For the subsequent discussion, it is necessary to answer the following question: how can we determine if the initial guess x_0 is close enough to the solution $\bar{x}$, so we can expect that convergence will be achieved? One naïve answer could be to consider this criterion:

$$\left\| f(x_1) \right\| \ll \left\| f(x_0) \right\|, \tag{14}$$

i.e., if the residual after the first iteration is much smaller than the residual computed with the initial guess, then it's likely that the residual after the next iteration will be even smaller and convergence will be achieved soon.

In fact, due to the consequences of Theorems 3 and 4, this is not a sound criterion: Theorem 3 suggests to trivially set $z_0 = 0$, which may lead to large values of the linear residuals $l(x_0)$, while Theorem 4 states that after one iteration, $l(x_1) = 0$ no matter what. Hence, the norm of the overall residual vector may dramatically decrease after one iteration, just because large linear residuals becoming zero in one shot; this is by no means an indication that the nonlinear variables are close to convergence.

This observation leads to the need of defining a better criterion for being close to convergence.

Definition 1. *With reference to Newton's iteration* (6)*, define the nonlinear residual at iteration point x_{k-1} as*

$$r(x_{k-1}) = f(x_{k-1}) + f_z(z_k - z_{k-1}). \tag{15}$$

This definition basically removes all the linear components that will vanish after one iteration from the residual at step $k-1$. Another possible interpretation, with reference to the decomposition shown in (5), is that it computes the residual of the nonlinear part of the system equations only, which is the only one that is relevant for possible convergence issues.

We can then replace the right hand side of the naïve criterion (14) with the nonlinear residual, thus leading to this heuristic criterion for being close to convergence:

$$\left\| f(x_1) \right\| \ll \left\| r(x_0) \right\|. \tag{16}$$

This criterion is heuristic in the sense that it is possible to build on purpose counter-examples in which this condition holds, but yet Newton's method does not converge to a solution. However, in most practical cases, if the residual $f(x_1)$ after the first iteration is much smaller than the nonlinear residual computed with the initial guess $r(x_0)$, then the residual $f(x_2)$ after the second iteration will be even smaller, and the method will converge to a solution in a few iterations.

The next step is to come up with the formulation of some indicators that can be used to express relationship (16) in an equivalent way. This requires to introduce the following definitions.

Definition 2. *Consider Newton's iteration* (6)*. Assume that the function $f(x)$ is three times continuously differentiable in an open neighbourhood $\mathfrak{D}$ containing the initial guess x_0 and the result of the first iteration x_1. Denote the i-th component of function $f(x)$ as $f^i(x)$, its Jacobian matrix with respect to x as $f^i_x(x)$, and its Hessian matrix as $f^i_{xx}(x)$. One can write the following Taylor expansion*

$$f^i(x_1) = f^i(x_0) + f^i_x(x_0)(x_1 - x_0) + \\ + \frac{1}{2}(x_1 - x_0)' f^i_{xx}(x_0)(x_1 - x_0) + h^i(x_1, x_0) \quad (17)$$

which implicitly defines the higher-order residual functions $h^i(\cdot,\cdot)$.

Definition 3. *Define the coefficients $\alpha_i > 0$, $i = 1, \cdots, m$, such that*

$$|h^i(x_1, x_0)| = \alpha_i \left\| r(x_0) \right\|_\infty, \quad (18)$$

i.e.,

$$\alpha_i = \frac{\left| f^i(x_1) - \frac{1}{2}(w_1 - w_0)' f^i_{ww}(w_0)(w_1 - w_0) \right|}{\left\| r(x_0) \right\|_\infty} \quad (19)$$

and let

$$\alpha = max(\alpha_i). \quad (20)$$

Remark 1. *It may happen that when computing $f(x_1)$ some of the residuals cannot be computed because x_1 is out of their range of definition, e.g. because of logarithms or square roots getting a negative argument. In this case, it is necessary to perform a* damped *Newton iteration:*

$$f_x(x_{j-1})(x^*_j - x_{j-1}) = -\lambda f(x_{j-1}), \qquad j = 1, 2, \cdots \quad (21)$$

By taking a small enough positive value of λ, it is possible to get x_1 arbitrarily close to x_0, thus avoiding validity range issues. The Taylor expansion can be used to define $h^i(x^*_1, x_0)$ leading to the following modified definition of α_i

$$|h^i(x^*_1, x_0)| = \alpha_i \lambda^3 \left\| r(x_0) \right\|_\infty, \quad (22)$$

motivated by the fact that $h^i(x^*_1, x_0)$ is a third-order term in the Taylor expansion, so it is expected to shrink as λ^3, thus making the definition asymptotically invariant as $\lambda \to 0$. Hence, α_i can be computed as

$$\alpha_i = \frac{\left| f^i(x^*_1) - (1-\lambda) f(x_0) - \frac{1}{2}(w^*_1 - w_0)' f^i_{ww}(w_0)(w^*_1 - w_0) \right|}{\lambda^3 \left\| r(w_0) \right\|_\infty} \quad (23)$$

Definition 4. *Define the* curvature factor Γ_{ijk} *of the i-th nonlinear equation with respect to variables w_j, w_k after the first iteration as*

$$\Gamma_{ijk} = \left| \frac{1}{2} \frac{\partial^2 g^i(w_0)}{\partial w_j \partial w_k} \frac{(w_{1,k} - w_{0,k})(w_{1,j} - w_{0,j})}{\left\| r(x_0) \right\|_\infty} \right| \quad (24)$$
$$i = 1, \ldots, p,$$
$$j = 1, \ldots, q,$$
$$k = 1, \ldots, q.$$

Equipped with these definitions, we can now state the following Theorem:

Theorem 5. *Given a constant $\beta > 0$,*

$$\sum_{jk} \Gamma_{ijk} \leq \beta \qquad \forall i = 1, \cdots, p. \quad (25)$$

implies that

$$\left\| f(x_1) \right\|_\infty \leq (\alpha + \beta) \left\| r(x_0) \right\|_\infty \quad (26)$$

The main consequence of Theorem 5 is that if $\alpha_i \ll 1$ and $\Gamma_{ijk} \ll 1 \; \forall i, j, k$, then $\alpha \ll 1$ and it will be possible to pick a value of $\beta \ll 1$ that satisifes (25); hence, $\alpha + \beta \ll 1$ and therefore, thanks to (26), the heuristic convergence condition (16) will be satisfied. In short, if all the α_j and Γ_{ijk} coefficients are small, Newton's method is very likely to converge in a few steps. This condition is sufficient but by no means necessary to achieve convergence, in particular if more robust quasi-Newton methods are used to actually solve (1). However, one can put forth the following heuristic argument:

> *If the (quasi-)Newton solver fails to converge, and if some of the α_i and Γ_{ijk} coefficients are much larger than one, they are likely to be pointing out the initial guesses and nonlinear equations that are responsible for the convergence failure.*

Specifically, large α_i values will indicate that the i-th equation is involved, whereas large values of Γ_{ijk} will indicate that the i-th equation and the initial guesses of w_j and w_k are involved.

Note, from definition (24), that the Γ_{ijk} indicators exploit second-order information provided by the system Hessian, by weighting the magnitude of the change of the solution estimate $(w_{1,j} - w_{0,j})$ after the first Newton step with the curvature of the residual function.

This is a crucial point: the fact that the initial guess value of some variables are very far from the solution, so that $(x_{1,k} - x_{0,k})$ is very large, is *per se* not a relevant indicator of potential convergence issues; in fact, as Theorems 3 and 4 clearly imply, it is not relevant at all for linear variables z. It becomes more and more relevant as the residual depends on the variable in a strongly nonlinear way, as measured by the value of the second derivative.

DOI 10.3384/ecp218109

Regarding the α_i indicators, it is clear from their definition in (17)-(18) that they instead account for the combined effect of terms of order higher than two on the convergence to the solution. Unfortunately, these indicators only point to the involved nonlinear equations, but they cannot discriminate which of the initial guess(es) of nonlinear variables $w_{0,k}$ showing up in those equations are to blame for the convergence issues.

Additional indications about initial guess values potentially being responsible for convergence failure can be obtained by exploiting the following Theorem:

Theorem 6. *The sensitivity of the solution x_1 after the first NR iteration, with respect to changes in the initial guess x_0, can be computed as:*

$$\frac{\partial x_1}{\partial x_0} = \Sigma, \tag{27}$$

where

$$H_i = (w_1 - w_0)' f^i_{ww}(w_0) \tag{28}$$

$$H = \begin{bmatrix} H_1 \\ H_2 \\ \cdots \\ H_p \end{bmatrix} \tag{29}$$

$$\Sigma = -\left[f_x(w_0)\right]^{-1} \begin{bmatrix} H_{p\times q} & 0_{p\times(m-q)} \\ 0_{(m-p)\times q} & 0_{(m-p)\times(m-q)} \end{bmatrix}. \tag{30}$$

When computing the sensitivity Σ at the solution $x_0 = \bar{x}$, since $w_1 = w_0 = \bar{w}$, it follows that $H = 0$, so that $\Sigma = 0$. This means that if an initial guess equal to the solution plus an infinitesimally small perturbation $x_0 = \bar{x} + \delta x$ is chosen, the solution x_1 after the first NR iteration is not affected at all. This is consistent with the fact that $f(x)$ can be approximated as a linear function in a small neighbourhood of the solution $\bar{x}$, so that Theorem 2 guarantees that the first NR iteration converges to the solution $\bar{x}$ in just one iteration, irrespective of the initial guess.

If the initial guess x_0 is close enough to the solution $\bar{x}$ that the function $f(x)$ is still approximately linear in a neighbourhood containing x_0 and $\bar{x}$, then the same behaviour is preserved, i.e., the result after the first iteration will be insensitive to small changes of the initial guess, so $\Sigma \approx 0$.

As x_0 is chosen farther away from the solution $\bar{x}$, nonlinear effects kick in, accounted for by matrix Σ, which can be then considered an indicator of how far the initial guess is from the sweet spot of NR convergence. In particular, $\left|\sigma_{jj}\right| \ll 1$ means that the effect on $w_{1,j}$ of applying a certain perturbation to $w_{0,j}$ will be much less than the perturbation itself, meaning that the nonlinear effects are moderate, whereas if $\left|\sigma_{jj}\right| \gg 1$ the effect of such perturbation will be amplified after the first iteration, which is not consistent with the behaviour close to convergence. Hence, it is possible to put forth an argument similar to the one concerning α_i and Γ_{ijk} indicators:

> *If the Newton solver fails to converge, and if $\left|\sigma_{jj}\right| > 1$ for some j, then the initial guesses $w_{0,j}$ are likely to be responsible for the convergence failure.*

Similarly to the Γ_{ijk} indicators, the σ_{jj} indicators rely on second-order information provided by the system Hessian to assess the effect of the nonlinearity of the system of equations on the convergence of Newton's method. Contrary to Γ_{ijk}, σ_{jj} indicators only provide information about potentially problematic initial guesses $w_{0,j}$, while not providing any information about the involved nonlinear equations.

An interesting property of σ_{jj} coefficients, when computed from sets of equations containing quantities with physical units, is that they are dimensionless, so they are invariant with respect to the choice of units of the problem, and in general to the scaling of the system of equations. This makes them readily computable from the Modelica equations in SI units, even if the equations and variables are badly scaled from a numerical point of view.

The α_i and Γ_{ijk} indicators are also invariant with respect to the choice of the units of the unknowns x; however, they are *not* invariant with respect to the choice of the units of the *residual equations* $f^i(x)$. In fact, the definitions of both indicators rely on the infinity norm of the residual $\left\|r(x_0)\right\|_\infty = \max_i \left|r^i(x_0)\right|$, which is conceptually ill-defined if the residuals $r^i(x_0)$ have non-homogenous units: what is the maximum between 70 kg/s and 10^8 W in a system of equation with a steam flow rate equation and a power balance equation?

Hence, a proper implementation of the computation of α_i and Γ_{ijk} requires to employ *normalized* function residuals, which can be computed as explained in (Casella and Braun 2017). Once the residuals are nomalized, the α_i and Γ_{ijk} indicators will be invariant to the choice of units of the problem and to its scaling in general.

2.3 Diagnosing Newton Solver Failures

In case the quasi-Newton method of choice fails to solve one of the nonlinear strong components of the initialization problem (in the case $\lambda = 0$ if homotopy is used), it is possible to compute the α_i, Γ_{ijk} and σ_{jj} indicators as explained in the previous subsection. They can then be ranked in descending order and displayed.

The initial guesses $w_{0,j}$ and $w_{0,k}$ corresponding to the largest values of Γ_{ijk} and $\left|\sigma_{jj}\right|$, as well as the unknowns showing up in the equations with the larger values of α_i, are the most likely to be responsible for the solver failures, so their start attribute should be checked and possibly fixed first. If it is recognized that they have a different start value than what was expected, e.g. 300 K for a flame temperature instead of 1800 K, this can be easily pin-pointed and corrected. Otherwise, they could be tentatively increased or decreased based on the sign of $(w_{1,j} - w_{0,k})$.

This method will be particularly helpful in the case of large systems with many unknowns and only a few critical

start attributes, that will be spotted immediately at the top of the list of ranked indicators.

This method can also help library developers to pinpoint equations that are heavily involved in the failure because of their strong nonlinearity, as indicated by large α_i and Γ_{ijk} values. Such equations could be replaced with other equivalent ones that are less nonlinear, e.g. $a = b * c$ instead of $a/b = c$, or replaced by less nonlinear approximated equations by means of homotopy.

3 Implementation

The method illustrated in the previous Section was implemented in the OpenModelica runtime, where it can be activated by setting the simulation flag `-lv=LOG_NLS_NEWTON_DIAGNOSTICS`. Jacobians are obtained using the same functions that are employed to get them for the quasi-Newton solvers used for simulation, which can either use code obtained with symbolic differentiation or resort to numerical finite differences when that is not possible. Hessians are computed numerically from Jacobians by means of finite differences.

Although it would be conceptually possible to use symbolic differentiation also for the Hessian computation, developing the code to do so was not really worth the effort: the second derivatives are used to generate heuristic indicators, so a high precision is not really required, so that the trivial task of computing them by numerical differentiation was chosen instead. Computational times are also not an issue, since all the required computations are only carried out once and normally take much less than a second also for systems with hundreds of unknowns, so simplicity of implementation was favoured over efficiency. This also lead to the choice of always using the LAPACK dense solver DGESV to solve the linear equations of the Newton step, also in case of systems with high sparsity ratio.

As mentioned in Section 2.1, the classification of linear and nonlinear equations and variables when tearing is used must also consider the nonlinearities introduced by the torn equation. This information could have been inferred symbolically by extending the structural analysis algorithm of the OpenModelica backend; however, this would have taken a significant development effort. It was therefore chosen to follow a simpler numerical approach instead.

Linear equations $l(x) = 0$ are therefore identified by checking if their residual after the first Newton step $f^i(x_1) \approx 0$, see Theorem 4. In case a damped Newton step needs to be computed to avoid out-of-range residual evaluations, the following condition is checked:

$$f^i(x_1^*) + (\lambda - 1) f^i(x_0) \approx 0 \tag{31}$$

Linear variables z are instead identified by checking if the corresponding column of the Hessians is close to zero.

4 Test Results

At the time of this writing, the implementation of the method in release 1.25.0 of OpenModelica is nearly complete, except for two features:

- getting numerical Jacobians when symbolically differentiated Jacobians are not available;
- scaling the residuals when computing the nonlinear residual norm $\left\| r(x_0) \right\|_\infty$

This unfortunately still prevents from testing the implemented method on the initialization problems of physical models using SI units. In particular, the steady-state initialization of thermo-hydraulic problems is notoriously prone to convergence failures, so it could benefit immensely from this type of diagnostics, but unfortunately it involves badly scaled variables such as pressures in Pa, as well as medium models such as the IF97 water/steam model, that cannot be symbolically differentiated.

The implementation was therefore tested on the three example cases shown in (Casella and Bachmann 2021), namely a simple thermo-hydraulic system, a nonlinear DC circuit, and an AC circuit. In all the three cases, the exact analytic solution of the system is known, so it is possible to set the start attribute of the unknowns at various distances from the solution, testing the ability of the proposed method to highlight the ones that need to be fixed in order to ensure the convergence of the Newton method. These test problems use well-scaled units with all the unknowns having order of magnitude of unity, so there is no need to scale the elements of $\left\| r(x_0) \right\|_\infty$ to obtain good results.

The computed values of the α_i, Γ_{ijk}, and $\left|\sigma_{jj}\right|$ indicators were very close to the ones that are reported in (Casella and Bachmann 2021), which were computed by ad-hoc, manually written Matlab code, thus validating the C-code implementation in the OpenModelica runtime. In order to obtain the same results, tearing had to be de-activated, so that the raw nonlinear system written in Modelica was passed directly to the diagnostic routine.

The results of one test case are shown here. The test system is the model of an electrical DC circuit, where the series connection of N resistors and one diode is connected to an ideal voltage source, which provides a certain fixed power P. The system is described by the following set of implicit equations:

$$i - i_s \left(e^{v_d / v_t} - 1 \right) = 0 \tag{32}$$

$$vi - P = 0 \tag{33}$$

$$v - \sum_{j=1}^{N} v_j - v_d = 0 \tag{34}$$

$$v_j - Ri = 0, \qquad j = 1 \cdots N \tag{35}$$

where i_s, v_t, P, R are known parameters, $x = \begin{bmatrix} i & v_d & v & v_1 & v_2 & \cdots & v_N \end{bmatrix}'$, $w = \begin{bmatrix} i & v_d & v \end{bmatrix}'$, $z = \begin{bmatrix} v_1 & v_2 & \cdots & v_N \end{bmatrix}'$.

DOI 10.3384/ecp218109

Note that the system has $N+1$ linear equations and only two nonlinear equations: the first is very strongly nonlinear, due to the exponential term in the diode equation, while the second, containing the voltage-current product, is only mildly nonlinear. The system has three nonlinear variables, which are the only ones for which the initial guess value matters, according to Theorem 3.

Taking $i_s = 6.9144 \cdot 10^{-13}$, $v_t = 25 \cdot 10^{-3}$, $P = 10.7$, $R = 1$, $N = 10$, the system has an exact solution $i = 1$, $v_d = 0.7$, $v = 10.7$, $v_j = 1$. If the start attributes of i, v_d, and v are reduced by 10% with respect to the known solution, the quasi-Newton solvers of the OpenModelica runtime fail to solve the system.

When activating the Newton Diagnostics runtime flag, OpenModelica generates the diagnostic output shown in Fig. 1.

The first table shows results by variable, ranked in descending order of potential criticality; for each variable, the largest value of Γ_{ijk} and $\left|\sigma_{jj}\right|$ that involve it are reported. The result unequivocally points out some very serious problem with the start attribute of v_d; indeed, due to the exponential term in the diode equation and to the small value of the v_t parameter, an error of 10% in the initial guess for v_d is enough to cause convergence failure. Conversely, the same 10% error in the start values of i and v, which are involved in the mildly nonlinear equation $vi - P = 0$, is not problematic at all. In fact, increasing the start attribute of v_d from 0.63 to 0.66 is enough to ensure the convergence of the quasi-Newton algorithm used by OpenModelica to solve the nonlinear system, without any need to touch the start attributes of i and v.

The second table shows results by equation, ranked in descending order of potential criticality. The "Eq no." index refers to the ordering of the residuals in the runtime solver, in case one wanted to inspect it in detail, whereas the "Eq idx" index refers to the index of the equations in the equation-based debugger (Sjölund et al. 2014), which displays the solved equations after structural analysis and symbolic manipulation, see Fig. 2. The diode equation is clearly indicated as the one creating trouble, while, the power equation is not creating any significant problem with the selected start attributes for its variables i and v.

For more details about the other test cases, the interested reader is referred to (Casella and Bachmann 2021).

5 Conclusions

The convergence failure of Newton's method during the solution of nonlinear systems of equations for the initialization of Modelica model is a possible critical outcome of the attempt of simulating Modelica models with implicit nonlinear equations, particularly when initializing models in steady state. Such an occurrence is a serious issue for Modelica tool users and can be a show-stopper for inexperienced users that have no idea how to fix the problem.

This paper presents the implementation in the OpenModelica tool of methods originally devised by Casella and Bachmann (2021) to diagnose such failures and help resolving them in a user-friendly manner. The presented implementation outputs lists of potentially problematic start attributes and nonlinear equations, ranked in descending criticality order.

In the authors' opinion, the presented methods provide much needed support in the debugging of failing Modelica models, particularly if coupled with user-friendly equation-based debuggers such as the one presented in (Sjölund et al. 2014).

The implementation was successfully tested in some illustrative test cases taken from Casella and Bachmann (2021). Unfortunately, at the time of this writing it was not yet possible to compute the normalized function residuals, as explained at the end of Section 2.2, because that requires a substantial refactoring of the OpenModelica runtime code. This prevented testing the method on more complex and realistic Modelica models using SI unit variables, whose residuals are usually quite badly scaled. Such a refactoring is currently on-going and planned to be finalized for the next 1.26.0 release of the software.

Future work will also involve a tighter integration of these run-time methods with the equation-based debugger of OpenModelica, which currently only provides static information elaborated at compile time.

References

Casella, Francesco and Bernhard Bachmann (2021). "On the choice of initial guesses for the Newton-Raphson algorithm". In: *Applied Mathematics and Computation* 398.125991, pp. 1–18. DOI: 10.1016/j.amc.2021.125991.

Fritzson, Peter et al. (2020). "The OpenModelica integrated environment for modeling, simulation, and model-based development". In: *Modelling, Identification, and Control* 4, pp. 241–285. DOI: 10.4173/mic.2020.4.1.

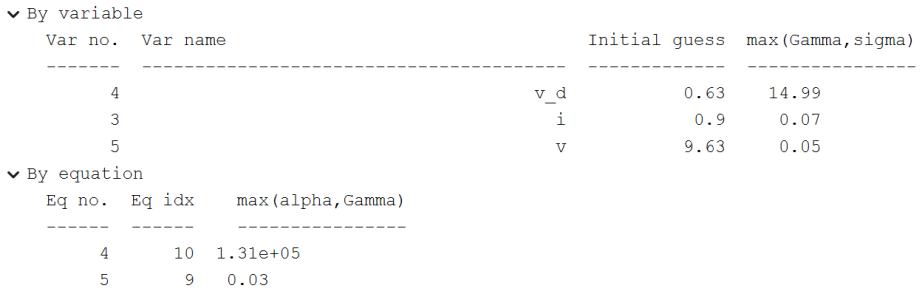
Casella, Francesco and Willi Braun (2017-12). "On the importance of scaling in equation-based modelling". In: *8th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools, EOOLT 2017*. Wessling, Germany, pp. 3–7. DOI: 10.1145/3158191.3158192.

Sielemann, Michael et al. (2011-03). "Robust Initialization of Differential-Algebraic Equations Using Homotopy". In: *Proceedings 8th International Modelica Conference*. Ed. by C. Clauss. Modelica Association. Dresden, Germany, pp. 75–85. ISBN: 978-91-7393-096-3. DOI: 10.3384/ecp1106375. URL: http://www.ep.liu.se/ecp/063/010/ecp11063010.pdf.

Sjölund, Martin et al. (2014-03). "Integrated Debugging of Equation-Based Models". In: *Proceedings 10th International Modelica Conference*. The Modelica Association. Lund, Sweden, pp. 195–204. ISBN: 978-91-7519-380-9. DOI: 10.3384/ECP14096195.

Mattsson, S. E., H. Elmqvist, and M. Otter (1998). "Physical system modeling with Modelica". In: *Control Engineering Practice* 6.4, pp. 501–510.

Modelica Association (2023-02). *Modelica – A Unified Object-Oriented Language for Systems Modeling. Language Specification Version 3.6*. Tech. rep. Linköping: Modelica Association. URL: https://specification.modelica.org/maint/3.6/MLS.html.



By variable

Var no.	Var name	Initial guess	max(Gamma,sigma)
4	v_d	0.63	14.99
3	i	0.9	0.07
5	v	9.63	0.05

By equation

Eq no.	Eq idx	max(alpha,Gamma)
4	10	1.31e+05
5	9	0.03

Figure 1. OpenModelica diagnostic output on the DC circuit example: initial guess values and equations most critical for convergence are shown, ranked in descending order of criticality (rightmost column).

28	initial	non-linear, unknowns: 13, iteration variables: 13
1	initial	(residual) v_[9] - R * i = 0
2	initial	(residual) v_[8] - R * i = 0
3	initial	(residual) v_[7] - R * i = 0
4	initial	(residual) v_[6] - R * i = 0
5	initial	(residual) v_[5] - R * i = 0
6	initial	(residual) v_[4] - R * i = 0
7	initial	(residual) v_[2] - R * i = 0
8	initial	(residual) v_[1] - R * i = 0
9	initial	(residual) v * i - P = 0
10	initial	(residual) i - i_s * (-1.0 + exp(v_d / v_t)) = 0
11	initial	(residual) v_[3] - R * i = 0
12	initial	(residual) (-v_[1]) - v_[2] - v_[3] - v_[4] ...v_[7] - v_[8] - v_[9] - v_d - v_[10] + v = 0
13	initial	(residual) v_[10] - R * i = 0

Figure 2. OpenModelica declarative debugger output on the DC circuit, each equation index (Eq idx) is shown in the first column.

DOI
10.3384/ecp218109

Modelica2Pyomo: a tool to translate Modelica models into Pyomo optimization models

Matteo L. De Pascali[1] Lorenz T. Biegler[2] Emanuele Martelli[3] Francesco Casella[1]

[1]Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milan, Italy, {matteoluigi.depascali,francesco.casella}@polimi.it
[2]Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, United States of America, lb01@andrew.cmu.edu
[3]Dipartimento di Energia, Politecnico di Milano, Milan, Italy, emanuele.martelli@polimi.it

Abstract

Tasks involving Modelica models often do not simply investigate the dynamic behavior of a system, but rather also want to characterize possible optimal control strategies according to suitable criteria. Unfortunately, since Modelica does not support out-of-the-box optimization features, users are often forced to use other tools to code again the system model for optimization studies. For this reason, the authors present Modelica2Pyomo, an open-source tool to translate Modelica models into Pyomo optimization programs, leveraging on their flat Base Modelica representation. This work illustrates the main features of Modelica2Pyomo, including automatic variables and constraints normalization, expressions manipulation and initialization via Modelica simulation results. To demonstrate the capabilities of this framework, two examples are showcased, including an industrial-grade open-loop optimal control problem of a solid-oxide fuel cell.

Keywords: Modelica, Model Conversion, Base Modelica, Pyomo, Dynamic Optimization

1 Introduction

Over the years, the Modelica language has established itself as one of the main tools for the formulation of system-level models for dynamic simulation purposes. Even though the size and the complexity of the systems that can be simulated with this language has increased dramatically since Modelica's inception, its object-oriented and equation-based nature, together with its drag&drop flowsheeting feature, significantly tames the challenges faced in the formulation of such models. Moreover, the possibility of highly customizing pre-existing and newly coded models, makes Modelica a suitable choice for academy and industrial R&D users that study bleeding edge technologies (see for example De Pascali and Casella (2024)).

Such studies often do not consist solely of dynamic simulations but they rather aim at characterizing both the system behavior and apply optimization routines to formulate, for example, optimized control strategies according to specific criteria. Since Modelica does not provide out-of-the-box optimization features, this second task requires a different framework. Often, users resort to different software and need to create from scratch a copy of the Modelica model in another programming language. This process requires modeling the same system twice and is prone to *manual translation* errors.

For this reason, the Modelica community spent quite some effort to build a bridge between the simulation and the optimization worlds. The main contributions were initially brought by the open-source software JModelica.org (Åkesson et al. 2010) and OpenModelica (Fritzson et al. 2020), both through the implementation of the Optimica framework (Åkesson 2008). Optimica extends Modelica with language constructs that enable the formulation of optimization problems based on Modelica models. Optimica-enabled tools communicate with optimization solvers such as IPOPT directly or through the CasADi framework (see Wächter and Biegler (2006) and Andersson et al. (2019), respectively).

This approach has several advantages over the *manual translation* approach. Most importantly, the system model is written only once and in an equation-based object-oriented framework: this avoids the possibility of introducing errors and makes it easier to model systems that would be too complex to code in a flat equation-based framework. Moreover, the resulting optimization problem benefits from the pre-processing executed by the Modelica compiler on the original Modelica model before its translation, e.g. alias elimination, index reduction, parameter evaluation, etc., to further reduce the size and the complexity of the problem that is fed to the underlying optimization solver. Finally, Modelica GUIs can support the formulation of complex models with the drag&drop flowsheeting feature mentioned above, easing the user task of connecting different units and components to build the final model to optimally control.

Unfortunately, the open-source JModelica.org development was discontinued in 2019 in favor of the commercial cloud-based software Modelon Impact, which includes the Optimica Compiler Toolkit; the development and maintenance of optimization features in OpenModelica mostly stopped around 2016, so there is currently no available industrial-strength open-source solution to solve dynamic

optimization problems on Modelica models.

To fill this gap, the authors present Modelica2Pyomo, an open-source tool developed in Python that allows to automatically translate a Modelica model into an equivalent Pyomo model, that can be used to formulate and solve optimization tasks with state-of-the-art solvers.

The input of the tool is a Modelica model, possibly using advanced object-oriented features such as inheritance, a-causal connections, hierarchically structured models with parameter propagation, expandable connectors, stream connectors, inner/outer objects, etc. The model is first flattened by the open-source OpenModelica tool into an equivalent Base Modelica (Kurzbach et al. 2023) representation, that basically contains a set of variable declarations and a corresponding set of (implicit) differential-algebraic equations (DAEs). As a matter of fact, the case of the Modelica2Pyomo tool is a nice application of the main aim of Base Modelica, which is *"to provide a much lower barrier of entry to the Modelica world, since writing a Base Modelica compiler or interpreter is a much easier task than writing a full-fledged Modelica compiler"* (OpenModelica Consortium 2025).

Pyomo (see Bynum et al. (2021)) is an established open-source software package for formulating and solving large-scale optimization problems in Python, with a clear to read semantics. Pyomo's modeling constructs can be used to express a wide range of optimization problems, including nonlinear and mixed-integer programs. In particular, the package Pyomo.DAE by B. Nicholson et al. (2018) allows to formulate the differential-algebraic constraint equations of optimal control problems in a declarative way, which is very close to flat Base Modelica code. The optimal control problem is then automatically transformed by Pyomo into a form suitable for optimization solvers by means of direct collocation or finite difference discretization methods, including the automatic generation of sparse Jacobians and Hessians.

The output of the tool is a configurable Python script containing the generated Pyomo model with the same variables and equations as the original Modelica model. The formulation of the actual optimization model, i.e., the objective function, additional degrees of freedom, and constraints of the optimization problem can be subsequently added manually to perform a wide range of tasks and analyses.

The main advantage of the Pyomo framework compared to other solutions (e.g., CasADi and Optimica Compiler Toolbox) lies in the wide range of supported optimization solvers, both open-source and commercial, which allows to pick the best-suited one for the specific case at hand. The approach proposed in this work thus offers more flexibility compared to the available alternatives. In some cases, commercial optimization solvers may be required to tackle particularly challenging problems; on the other hand, the remaining part of the required toolchain is entirely open-source.

A limitation of the current implementation of Modelica2Pyomo is that it does not yet benefit from the symbolic manipulations such as alias elimination and index reduction, since the Base Modelica code can currently only be generated by the OpenModelica front-end, whereas these operations are carried out by the back-end. Nonetheless, the complex examples presented in this work can be successfully run without such manipulations. Recents results on the impact of symbolic manipulation of the model equations on the efficiency and convergence robustness of optimization solvers could also be considered, see e.g., (Naik et al. 2025) and (Parker et al. 2022). Such manipulations could be performed on the Pyomo side, but some of them could also be carried out on the Modelica tool side, if they are already implemented for the optimization of simulation code.

This paper is organized as follows: Section 2 describes the Modelica2Pyomo tool in terms of both usage and implementation. Section 3 describes two test cases, showcasing the flexibility of the tool and demonstrating the level of complexity of the Modelica models that can be successfully handled. Section 4 draws the conclusion of this work and outline possible future developments.

The code of Modelica2Pyomo, together with the code of the examples showed in this paper, is available in (De Pascali and Casella 2025).

2 The Modelica2Pyomo tool

The Modelica2Pyomo tool is a Python script that translates a Modelica model into an equivalent Pyomo model. In its current form, the tool accepts as input the flattened Base Modelica representation of a square Modelica simulation model with as many equations as unknown variables, with all the input variables bound to some expressions that only depend on time. The corresponding generated Pyomo model is a degenerate dynamic optimization model with a trivial objective function, no free inputs, and no inequality constraints, whose solution corresponds to the simulation of the original Modelica model. The comparison between the simulation results of the Modelica model and the solution of this degenerate dynamic optimization problem allows to verify that the DAEs have been correctly transcribed by the whole toolchain.

One can then formulate actual dynamic optimization problems by manually editing the generated Pyomo code:

- replacing the objective function with the desired one;
- removing the equations that determine the time-dependent system inputs, thus freeing the inputs that should be determined as a result of the optimization problem;
- optionally adding inequality constraints to the system variables.

2.1 Tool inputs

To generate the equivalent Pyomo model, Modelica2Pyomo requires as input the flattened Base Modelica code of the Modelica model and optionally the result file

DOI 10.3384/ecp218117

of a Modelica simulation of the model to provide an initial guess to the solver. Employing Base Modelica code as input allows to process a flat representation of the Modelica model which only contains lists of variables, functions and record declarations, equations and initial equations, without bothering about object-oriented features such as modularity, connectors, inheritance, etc. This greatly simplifies the writing of the Modelica2Pyomo tool.

Additionally, in order to minimize the tool development work, the generated BaseModelica code needs to fulfill some additional requirements, which are obtained by flattening the model with OpenModelica and setting some special flags in the compiler:

- `-d=evaluateAllParameters` replaces all parameter instances in the equations with their literal numerical value; this allows to skip the parameter section of the model entirely, as well as the handling of dependencies among parameters enforced by non-trivial parameter-binding expressions; it also prevents conditional equations and expressions depending only on parameters to show up in the equation that will be passed to Pyomo;
- `--frontendInline` enables inlining of functions (i.e., the body of a function is inserted in all places where the function is called, swapping the formal inputs with the actual inputs), to make the resulting model fully equation-based.
- `--baseModelicaOptions=moveBindings` moves all the binding equations of Real variables into the `equation` section, so there is no need to parse binding equations separately from regular equations;
- `--baseModelicaOptions=scalarize` rolls out array variable declarations and array equations (including `for` loops) into their scalar elements, so that each variable declaration corresponds to a scalar variable and each equation is a scalar equation in the Base Modelica code; this also minimizes the tool development effort, because there is no need to handle arrays, slicing, iterators, etc., which are all handled upstream by the OpenModelica tool.

2.2 Tool capabilities and limitations

Modelica2Pyomo can handle Modelica models of arbitrary complexity from an object-oriented point of view: features such as models defined by instantiating components, parameter passing, a-causal connectors, composite connectors, stream connectors, expandable connectors, inner/outer components, inheritance, replaceable classes and components, N-dimensional arrays of variables and components, are all supported. The availablity of a Base Modelica interface with a state-of-the-art compiler such as OpenModelica makes it possible to off-load the processing of all these features onto the Modelica tool, so that the implementation of the Modelica2Pyomo tool can be kept very lean, while remaining always be up-do-date with the latest additions to the Modelica language, as long as an updated version of the Modelica tool is used.

This approach can make Modelica2Pyomo a very attractive tool for the Pyomo community. Pyomo indeed includes several features for the handling of modular, structured models, but they are by no means as powerful as those made available by the Modelica language. In the authors' opinion, it would be a waste of time to re-invent yet another full-fledged object-oriented modelling language for Pyomo – a much more efficient approach is to re-use Modelica as a modelling front-end (including drag-and-drop graphical user interfaces such as OMEdit) and then use OpenModelica for flattening and Pyomo as an optimization back-end. Modelica2Pyomo is meant to be a first step in this direction.

The main limitation on the models that Modelica2Pyomo can handle stems from the capabilities of the Pyomo.DAE tool it is based upon. Pyomo.DAE can only take as input purely equation-based index-1 DAEs, described by scalar variables and equations. Hence, the upstream Modelica tool must be able to reduce the original model to that form. This entails some restrictions in the set of supported models.

First and foremost, the original Modelica model must be purely continuous-time; hence, it should not contain discrete or clocked variables. Additionally, Pyomo.DAE expects the equations to be continuously twice-differentiable; this also rules out conditional expression and equations, which are potentially not differentiable, and which in any case have no counterpart in Pyomo.DAE.

Conditional expression could be supported by Modelica2Pyomo by replacing expressions such as `if a > b then c else d` with analytic expressions such as

$$\frac{1}{2}\left[d+c+\tanh\left(\frac{a-b}{\varepsilon}\right)(d-c)\right], \tag{1}$$

where ε is a suitably small constant; with the same logic, conditional equations such as

```
if a > b then
  x = y;
else
  z = w;
end if;
```

could be approximated with the following smoothed approximation to a complementary condition:

$$z+x-w-y+\tanh\left(\frac{a-b}{\varepsilon}\right)(z+y-w-x)=0. \tag{2}$$

This feature is not yet implemented at the moment, but it could be easily added in the future.

Last, but not least, Pyomo can only handle index-1 DAE models; at the time of this writing this implies that the original Modelica model also has to be index-1, because OpenModelica currently outputs the flat Base Modelica code immediately after the flattening phase, prior to structural analysis and index reduction. It is possible to check that this condition holds by

compiling the Modelica model for simulation with the flags `--preOptModules-=removeSimpleEquations -d=bltdump`, which reports whether index reduction is necessary to obtain an index-1 system. This restriction may be lifted in the future if OpenModelica becomes capable of outputting index-reduced Base Modelica code.

As mentioned at the beginning of Section 2, the originally supplied Modelica model must have prescribed input variables, which will then be freed and determined as the result of the optimization problem. It is possible, and in fact recommended, to also provide a reference simulation result file, obtained with those prescribed input values, which will be employed to provide an initial guess for the optimal solution, considerably improving the chances of convergence of the optimization problem. If no result file is supplied, the variables will be initialized to zero.

As a final remark, the Modelica2Pyomo tool was mainly designed to solve *dynamic* optimization problems, where the constraint equations are DAEs. However, as a special case, it can also solve *static* optimization problems, where all the constraint equations are purely algebraic. This can be helpful in a number of application fields; for example, in the case of thermal power generation systems, it can be first used to compute optimal (i.e., maximum efficiency) steady-state off-design operating points, and later to compute optimal (i.e., fast) transients to move between them at maximum speed while respecting operational constraints such as maximum temperature gradients.

2.3 Notable Modelica2Pyomo features

A crucial feature offered by Modelica2Pyomo is the automatic normalization of variables, constraints and objective function.

One aspect of declarative equation-based modeling is that the use of dimensionally consistent SI units for the physical variables is preferable; however, in many application areas this can lead to systems of equations which are badly scaled from a numerical point of view. As explained by Casella and Braun (2017), even though this is usually not declared explicitly in the literature and in the documentation about optimization solvers, the implicit assumption taken by their developers is that the unknowns of the problem have an order of magnitude close to unity or at least not too far from that, e.g. in the range $10^{-3} - 10^{3}$. The optimization of badly scaled models of significant complexity are inevitably bound to fail, a fact further confirmed by the authors' experience with convergence failures before scaling was included in Modelica2Pyomo.

For these reasons, the normalization procedures described in Casella and Braun (2017) are implemented in Modelica2Pyomo. For the variables, the normalization value is chosen as the maximum value between the assigned nominal value and the initialization value assigned to the variable in the first time instant.

Regarding the constraint residuals and the objective function, again as suggested by Casella and Braun (2017), they are scaled by the factor

$$f_{norm} = \max_j \frac{df}{dx_j} x_{j,norm} \tag{3}$$

where x_j are the unknown variables and $x_{j,norm}$ are their normalization values. The lines of code responsible for the variable normalization are placed after each variable declaration while the code for the constraints and objective function normalization is included in the Pyomo script right before the solver declaration at the end of the file.

Modelica2Pyomo offers also an expression manipulation feature that consists in the replacement of logarithm expressions with a more suited equivalent formulation. Since logarithms are not defined for arguments equal or less than zero, the solvers will encounter issues if their argument become negative during Newton-Raphson iterations. An alternative and equivalent formulation can be introduced to overcome this issue:

$$\begin{cases} \ln(x) \rightarrow u \\ \exp(u) - x = 0 \end{cases} \tag{4}$$

The logarithm is replaced by an auxiliary variable u and the second equation, which does not suffer from the same domain limitations of the logarithmic function, is enforced to replace the logarithm expression.

Concerning the initial guess values for the optimization problem, Modelica2Pyomo offers three options:

1. no initial guesses are employed (all initial guesses are zero);
2. each variable is initialized to a constant value obtained from the results file of the Modelica simulation at a specified time instant;
3. each variable is initialized with the time evolution of the corresponding Modelica variable obtained from the results file of a suitable reference simulation.

Finally, if a dynamic optimization problem is formulated, two different options are provided to specify its initial conditions:

1. employ the initial equations of the Modelica model;
2. fix the initial values of the variables appearing in the Modelica derivative operator according to the values reported in the results file of the simulation.

The first option is in principle more flexible, since it covers those cases where the initial condition is not fixed a priori but is also part of the optimization problem; on the other hand, difficult-to-solve initialization problem could lead to convergence problems of the optimization solver.

The second option only works in the case of initial conditions that are fixed for the optimization problem, with the advantage that it can leverage on advanced features of the Modelica tool (e.g., homotopy) to successfully solve hard initialization problems. This is essential in cases such as the steady-state initialization of thermal power generation systems.

DOI 10.3384/ecp218117

2.4 Modelica2Pyomo script overview

Modelica2Pyomo is organized in several functions, which are all executed through the *m2p()* function. The arguments that should be passed to this function, together with the paths of the Modelica model and the optional results file, are the following:

- a string with the name of the Python file that will contain the Pyomo model;
- a string with the name of the solver to be employed;
- a string specifying whether the generated problem will be *Static* or a *Dynamic* optimization program;
- in case of a dynamic optimization problem, a string specifying whether the initial equation of the Modelica model should be translated or the initial values of the states should be retrieved from the results file;
- in case of a dynamic optimization problem, the start and finish time instants of the simulation;
- in case of a dynamic optimization problem, a dictionary containing information about the discretization method (e.g. finite differences or collocation) and the respective settings (e.g., number of finite elements, discretization scheme, etc.);
- a flag to enforce the *min*/*max* attributes in the imported reference simulation file;
- a flag to enable expressions manipulation, e.g., logarithms transformations;
- a string containing the Pyomo code to specify the objective function;
- optional strings containing additional constraints and utility Python commands to be inserted before and after the solver definition lines of the Pyomo model.

Once it is run, the tool performs the set of operations that is briefly described below. Each step roughly corresponds to a Python function, which adapts to the user requirements according to the input described above.

1. read the Base Modelica file and store each line in a list;
2. store in a Python dictionary with the DyMat Python package by Jörg (2011) the numerical results of a Modelica simulation of the system;
3. remove from the Base Modelica code comments and other characters which are not related to the model description;
4. identify, if present, the lines related to *TimeTables* and *CombiTimeTables* declarations, store the tables and instantiate a string containing their Pyomo code;
5. analyze the variable section of the Base Modelica model and store name, min, max and nominal values of each variable;
6. employ the information of the previous step and the DyMat results dictionary to build a string containing the necessary code to instantiate the variables of the model, setting min/max bounds, initial values and normalization value employing the Pyomo syntax;
7. analyze the equation section of the Base Modelica model and instantiate a string containing the equations of the problem using the proper Pyomo syntax. In addition, look for the following cases:
   - if a variable is set equal to a number, use the Pyomo *fix* variable attribute to model the equation instead of the original formulation;
   - if a logarithm expression is present and the optional flag is enabled, store the logarithm argument and replace the logarithm as shown in Equation (4);
   - if a time derivative operator is present, store the associated variable to create a list of the states;
8. analyze the initial equation section and instantiate a string with additional equations to define the Modelica parameter characterized by the *fixed = true* attribute, which, otherwise, would be missing their defining equation;
9. use the list of state variables created in the previous step to instantiate the respective time derivatives with the Pyomo *DerivativeVar* object;
10. analyze the initial equation section to instantiate a string featuring the initial equations of the Pyomo optimization problem;
11. write on a .py file the Pyomo code contained in the string created in the previous steps together with the objective function, constraints and objective function normalization code, solver settings and other custom lines included as arguments for the tool.

To execute many of the steps listed above, Modelica2Pyomo employs regular expressions through the Python native re and regex (Matthew 2025) packages to search for predefined patterns in the rows of the Base Modelica file and stores information about variables and constraints according to the specification of the Base Modelica language. Moreover, the quoted identifiers used for Base Modelica variables are adapted to remove the quotes, the Modelica dot notation, square brackets and commas, which are all replaced with underscores in the Pyomo model in order to comply with Python syntax rules.

As an example, Appendix A reports a simple, first-order model formulated in Modelica, its Base Modelica flat representation, and the corresponding Pyomo script generated by the Modelica2Pyomo tool.

3 Test cases

In this Section, two examples of Modelica model translation are showcased; both models are converted into open-loop dynamic optimal control problems, which are then successfully solved by the Pyomo framework. The first example is a simple, conceptual model that can be useful to understand how the tool works. The second example, instead, is a full-fledged industrial-grade system model, aimed at demonstrating the capability of the tool to handle real-life complexity models.

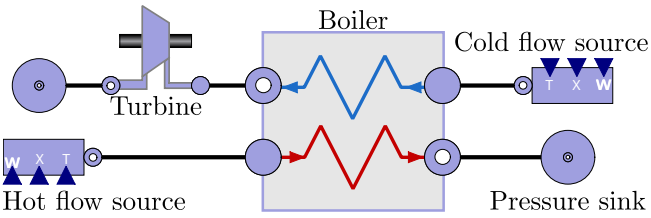


Figure 1. Modelica diagram of the s-CO_2 boiler and turbine model.

The corresponding BaseModelica flat models, which are the actual inputs of the tool, are available in the Examples directory of the Modelica2Pyomo package (De Pascali and Casella 2025).

3.1 Conceptual s-CO_2 boiler-turbine system

The first test features a conceptual Modelica model of a supercritical CO_2 boiler-turbine system. Figure 1 shows the diagram of the model. A hot flue gas stream heats up a cold stream of supercritical CO_2 which is expanded in a turbine with a prescribed outlet pressure to produce electrical power. The two streams are generated by ideal mass flow rate sources and their flow rate and temperatures can be changed arbitrarily. The medium model for supercritical CO_2 is the implicit Peng-Robinson equation of state, written in the equation section of the model.

In the first test case, the flow rates are fixed to constant values and are embedded in the optimization problem with the Pyomo *fix* attribute. The selected simulation time is 500 s and direct collocation with 100 finite elements, each with 3 collocation points, is the employed discretization method. Since the Modelica model is a square system having the same number of variables and equations, the translated optimization problem does not feature degrees of freedom, thus its optimal solution coincides with the simulated Modelica trajectory.

The coincidence of the two solutions demonstrates that the Modelica model was successfully translated into an equivalent Pyomo model, having the same solution.

To formulate a meaningful optimization problem, first two degrees of freedom are introduced, by *unfixing* the two mass flow rate variables (w_{hot} and w_{cold}), then a suitable cost function is chosen to guide the plant along transient conditions, as follows. The system is required to track a time-varying power output P_{turb} setpoint while keeping the turbine inlet temperature TIT constant. To this end, the following objective function is employed, with overbarred variables being the required setpoints, k indicating the simulation time step and coefficients, A_i being suitable weights for the different components of the expression:

$$\begin{aligned}\min \sum_k \; & (A_1 \cdot (P_{\text{turb}}(k) - \overline{P}_{\text{turb}}(k))^2 \\ & + A_2 \cdot (\text{TIT}(k) - \overline{\text{TIT}}(k))^2 \\ & + A_3 \cdot (w_{\text{hot}}(k) - w_{\text{hot}}(k-1))^2 \\ & + A_4 \cdot (w_{\text{cold}}(k) - w_{\text{cold}}(k-1))^2)\end{aligned} \tag{5}$$

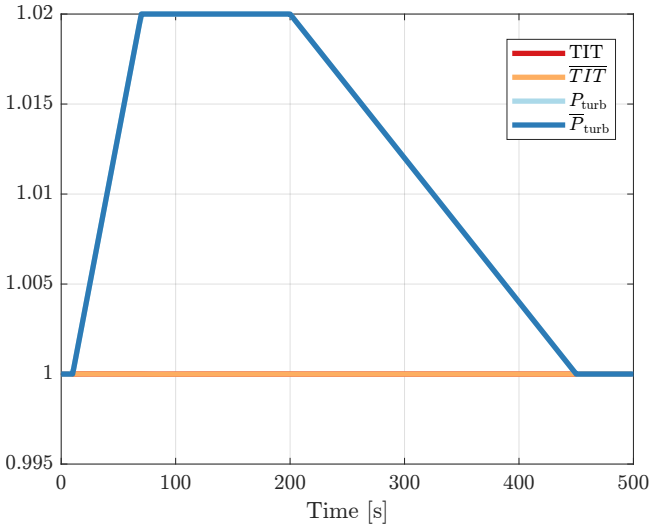


Figure 2. P_{turb} and TIT normalized profiles.

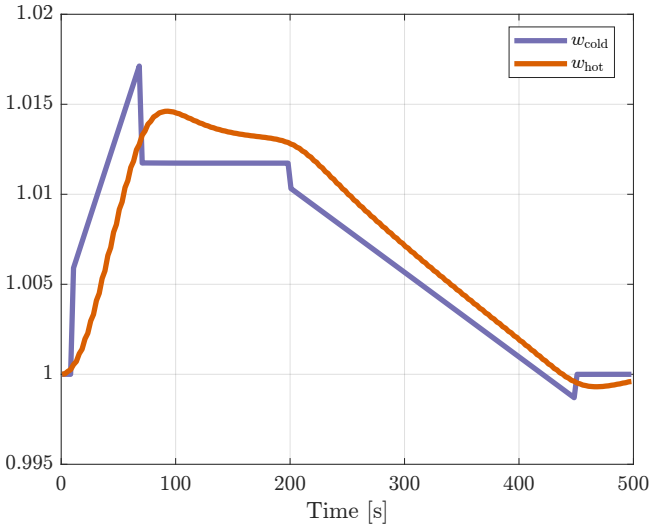


Figure 3. w_{hot} and w_{cold} normalized profiles.

The third and fourth terms of the objective function in (5) limit the rate of change of the optimization variables, moderating the control action.

Figure 2 shows that both P_{turb} and TIT follow accurately their setpoints, while Figure 3 shows the optimization variables w_{hot} and w_{cold}, that, as prescribed by the objective function, do not show abrupt variations. This first example shows that this class of simulation problems can be effectively translated into a nonlinear optimization problem, that features roughly 10000 variables and constraints. The most time consuming operation is the objective function normalization routine, while the Modelica to Pyomo compilation, the variables and constraints instantiation, the constraints normalization and the solver overhead is negligible. Table 1 summarizes these considerations and also shows a comparison between the open-source solver IPOPT and the commercial closed-source solver CONOPT 4.

DOI 10.3384/ecp218117

Table 1. Performance results for the s-CO_2 boiler-turbine test

	Time [s]
Modelica2Pyomo model translation	2.3
Pyomo variables and Constraints instantiation	0.2
Constraints normalization	0.01
Objective function normalization	12
Solver: IPOPT	0.3
Solver: CONOPT 4	0.8

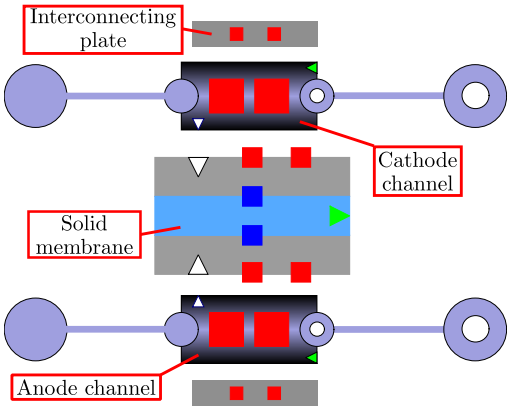


Figure 4. Modelica diagram of the solid-oxide fuel cell model.

3.2 Solid-oxide fuel cell

While the Section above showcased a toy example with relatively few equations, the goal of this Section is to present a relevant industrial case study. To this end, the 1-D solid-oxide fuel cell (SOFC) model presented by De Pascali, Donazzi, et al. (2023) and employed by De Pascali and Casella (2024) to model a complex thermal power generation system, is fed to Modelica2Pyomo to formulate an open-loop optimal control problem. The model has about 9000 variables and equations. Note that the full source code of the model is publicly available on GitHub. The model of the SOFC (see the Modelica diagram in Figure 4) is extremely nonlinear, due to the presence of strongly coupled nonlinear thermal, chemical and electrochemical phenomena. These describe the conversion at high temperatures of natural gas into hydrogen and its oxidation with oxygen ions to produce water, with the final aim of generating electrical power without resorting to fuel combustion.

The extensive use of object-oriented modeling techniques, that are necessary to tame the complexity of the formulation of such a challenging model, would make an hypothetical *manual translation* into an optimization framework such as Pyomo.DAE quite difficult, time consuming, and error-prone. This is exactly the reason that motivated the work presented in this paper: Modelica2Pyomo is capable of *automatically* converting such a complex Modelica model featuring 10000 equations and 160 states into a working Pyomo model.

As in the previous case, to formulate a dynamic optimization problem, direct collocation is used to discretize in time the simulation: for a 120 s simulation with 35 time steps, each with 3 collocation points, a NLP with roughly 850000 equality constraints is formulated. The goal of this test is to evaluate a trajectory for the SOFC extracted current I to optimally follow an electrical power output P_{SOFC} setpoint, while constraining the maximum allowed rate of change of the temperatures T of the fragile ceramic solid membrane of the SOFC. The following constraints is thus added manually to the Pyomo model for all the temperature variables of the SOFC solid membrane:

$$T(k) - T(k-1) < \frac{dT_{max}}{dt} \cdot \Delta t \tag{6}$$

$$T(k) - T(k-1) > -\frac{dT_{max}}{dt} \cdot \Delta t \tag{7}$$

where dT_{max}/dt is the maximum temperature rate of change prescribed by fuel cell manufacturers and Δt is the length of the simulation time step. In this test, dT_{max}/dt is varied to show how the solver is forced to follow less accurately the power setpoint $\overline{P}_{\text{SOFC}}$ when its variation is too steep. The following objective function is chosen:

$$\begin{aligned} \min \sum_k \; & (B_1 \cdot (P_{\text{SOFC}}(k) - \overline{P}_{\text{SOFC}}(k))^2 \\ & + B_2 \cdot (I(k) - I(k-1))^2 \\ & + B_3 \cdot (T(k) - T(k-1))^2) \end{aligned} \tag{8}$$

Also in this case, the coefficients B_i are employed to weight the contributions of the different terms of the objective function and the last two components are employed to moderate the control action.

Figure 5 shows the SOFC power output considering two different values for dT_{max}/dt: in the stricter case, $dT_{max}/dt = 0.2\,\text{K/s}$, while in the milder case, this value is three times higher. As expected, the control performance are worse in the stricter case, and the power setpoint is followed less accurately. Figure 6 shows the trends of the temperature of the solid membrane at the near the outlet of the SOFC, which respect the prescribed maximum rate of variation. Figure 7 shows the current I extracted from the fuel cell, which is the manipulated variable determined by the optimization problem.

Also in this much more challenging test case, Modelica2Pyomo is able to formulate a meaningful and working Pyomo model with no effort by the end user except for the manual inclusion of additional constraints for the membrane temperatures and the objective function. Compared to the previous example, Table 2 shows comparable compilation and instantiation times, while the time required by the solvers is higher, given the increased size of the optimization problem. Notice that the time employed to normalize the objective function is lower, since this simulation features less time steps, compared to the previous one. Moreover, IPOPT was not able to solve this model,

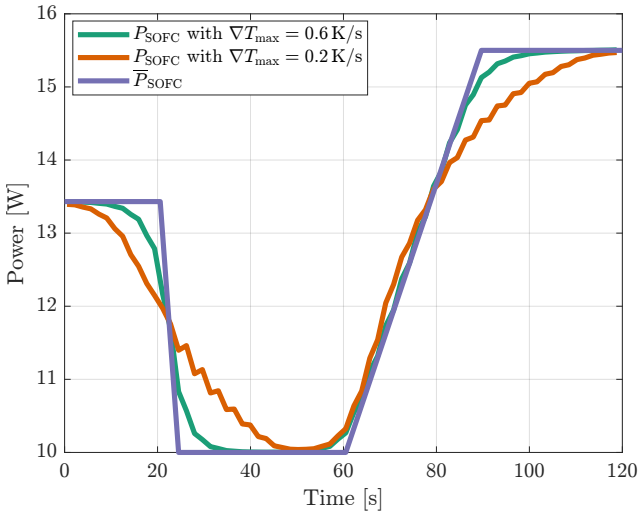


Figure 5. P_{SOFC} at varying dT_{max}/dt.

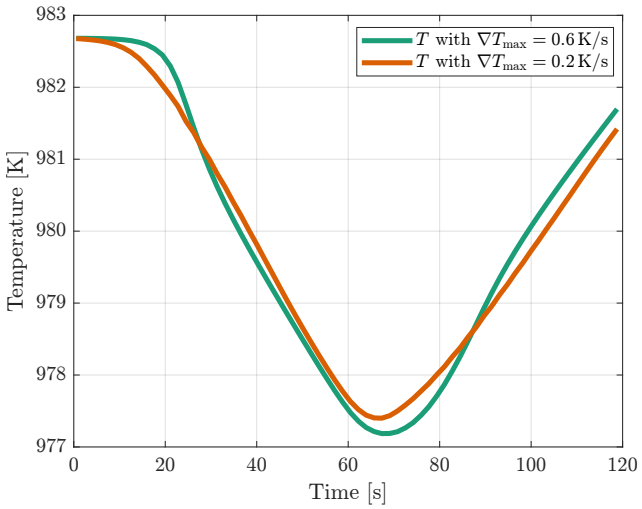


Figure 6. Membrane T at the SOFC discharge at varying dT_{max}/dt.

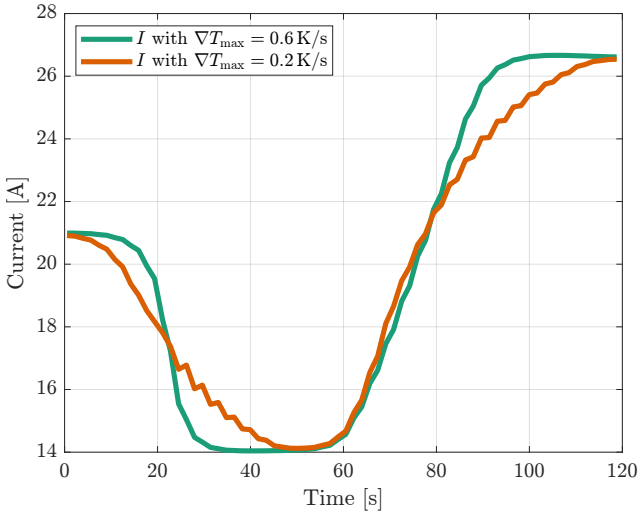


Figure 7. I extracted from the fuel cell at varying dT_{max}/dt.

Table 2. Performance results for the solid-oxide fuel cell test

	Time [s]
Modelica2Pyomo model compilation	3.7
Pyomo variables and Constraints instantiation	6.7
Constraints normalization	1.3
Objective function normalization	0.7
Solver: CONOPT 4	185

possibly indicating that a feasible path, active-set solver like CONOPT might be more suited for this class of problems, compared to an interior point method like IPOPT.

4 Conclusion

In this paper, the Modelica2Pyomo tool was presented. The aim of this tool is to create an open-source, flexible, and lightweight bridge between Modelica and the optimization world.

Modelica2Pyomo is a Python script that translates Modelica models into Pyomo models, exploiting the OpenModelica tool for flattening, the new Base Modelica language for flat Modelica model representation, and the Pyomo.DAE tool for writing DAE constraints for Pyomo in a high-level fashion. The flat, scalarized Base Modelica representation obtained through OpenModelica uses a very nimble set of language constructs, that can be efficiently handled by regular expressions to find Base Modelica language patterns that are converted into Pyomo expressions, without the need of embarking in the traditional workflow of compiler development, with parsers, lexers, abstract syntax trees, etc.

This approach allowed to drastically cut the required development and maintenance effort from the tens of person-years required for the development of a full-fledged Modelica compiler down to a few person-months effort, a nice demonstration of the potential that can be unlocked by the forthcoming Base Modelica standard.

The Modelica2Pyomo tool produces the Pyomo code corresponding to the set of DAE constraint equations, including their proper scaling, the transformation of some expression to more numerically robust equivalent ones, and the capability of importing intial guesses for the NLP solver from reference simulation results. The actual optimization problem (constraints and objective function) is then directly formulated using the Pyomo tool, which allows flexible access to a range of different solvers, both open-source and commercial.

The results obtained in two example test cases were reported, showing the capabilities of the proposed framework, which is able to successfully convert industrially-relevant complex Modelica models into nonlinear optimization programs with satisfying performance.

DOI
10.3384/ecp218117

The code of the tool is available as open source under GPL v3 license. Contributions for its further improvement and developments are welcome.

Future research directions will extend the range of Modelica constructs that can be handled by the tool, e.g., by handling conditional expressions and equations by regularization, improve the numerical robustness of the translated Pyomo.DAE model, as well as test the framework on even more challenging problems, e.g., optimizing transients of an innovative power plant that features the fuel cell model studied in this work as one of its components. One could also experiment with the use of optimization techniques for solving hard steady-state initialization problems, using Pyomo as a protoyping environment. It would also be interesting to explore the optimal control of Modelica systems including time delays and discrete variables, as well as optimization with embedded DAE solvers (shooting methods).

Modelica2Pyomo clearly demonstrates how Base Modelica can successfully enable the development of lean tools that employ Modelica models for uses other than simulation. The authors hope that this example further motivates the standardization effort of the Base Modelica language, as well as its improved support in Modelica tools.

Acknowledgements

This work was supported by the Italian Ministry of University and Research under Programma Operativo Nazionale "Ricerca e Innovazione" PONRI 2014-2020.

References

Åkesson, Johan (2008-03). "Optimica - An Extension of Modelica Supporting Dynamic Optimization". In: *Proc. 6th Intl. Modelica Conference*. Bielefeld, Germany, pp. 57–66.

Åkesson, Johan et al. (2010). "Modeling and optimization with Optimica and JModelica.org — Languages and tools for solving large-scale dynamic optimization problems". In: *Computers & chemical engineering* 34.11, pp. 1737–1749.

Andersson, Joel AE et al. (2019). "CasADi: a software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation* 11, pp. 1–36.

Casella, Francesco and Willi Braun (2017-12). "On the importance of scaling in equation-based modelling". In: *8th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools, EOOLT 2017*. Wessling, Germany, pp. 3–7. DOI: 10.1145/3158191.3158192.

Bynum, Michael L. et al. (2021). *Pyomo-optimization modeling in python*. Vol. 67. s 32. Springer.

De Pascali, Matteo Luigi and Francesco Casella (2024). "Modeling Innovative Thermal Power Generation Systems for the Energy Transition with Modelica: an Open Source and Flexible Solution". In: *2024 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, pp. 533–538.

De Pascali, Matteo Luigi and Francesco Casella (2025). *Modelica2Pyomo Repository*. https://github.com/looms-polimi/Modelica2Pyomo.

De Pascali, Matteo Luigi, Alessandro Donazzi, et al. (2023). "A control-oriented Modelica 1-D model of a planar solid-oxide fuel cell for oxy-combustion cycles". In: *2023 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, pp. 394–399.

Fritzson, Peter et al. (2020). "The OpenModelica Integrated Environment for Modeling, Simulation, and Model-Based Development". In: *Modelling, Identification, and Control* 4, pp. 241–285.

Jörg, Rädler (2011). *DyMat*. www.github.com/jraedler/DyMat.

Kurzbach, Gerd et al. (2023). "Design proposal of a standardized Base Modelica language". In: *Proc. 15th International Modelica Conference*. Aachen, Germany, pp. 469–478.

Matthew, Barnett (2025). *mrab-regex*. www . github . com / mrabarnett/mrab-regex.

Naik, Sakshi et al. (2025). *Variable aggregation for nonlinear optimization problems*. URL: https : / / arxiv . org / abs / 2502 . 13869.

Nicholson, Bethany et al. (2018). "pyomo.dae: A modeling and automatic discretization framework for optimization with differential and algebraic equations". In: *Mathematical Programming Computation* 10, pp. 187–223.

OpenModelica Consortium (2025). *Flattening models to BaseModelica*. www . openmodelica . org / doc / OpenModelicaUsersGuide/latest/basemodelica.html.

Parker, R. et al. (2022). "An implicit function formulation for optimization of discretized index-1 differential algebraic systems". In: *Computers & Chemical Engineering* 168.

Wächter, Andreas and Lorenz T. Biegler (2006). "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". In: *Mathematical programming* 106, pp. 25–57.

A Appendix: Source code of a simple first-order example

```
model Test
  parameter Real tau = 5;
  parameter Real mu = 30;
  Real u;
  Real x(nominal = 1, start = 1, fixed = true);
  Real y1(nominal = 50);
  Real y2(nominal = 30) = 0.5*y1;
equation
  u = 2;
  tau*der(x) = (u - x) + (u - x)^2;
  y1 = mu*x;
end Test;

//! base 0.1.0
package 'Test'
 model 'Test'
  final parameter Real 'tau' = 5.0;
  final parameter Real 'mu' = 30.0;
  Real 'u';
  Real 'x'(nominal = 1.0,
           fixed = true, start = 1.0);
  Real 'y1'(nominal = 50.0);
  Real 'y2'(nominal = 30.0);
 equation
  'y2' = 0.5 * 'y1';
  'u' = 2.0;
  5.0 * der('x') = 'u' - 'x' + ('u' - 'x') ^
      2.0;
  'y1' = 30.0 * 'x';
 end 'Test';
end 'Test';
```

```
import idaes
from pyomo.environ import *
from pyomo.core.expr import identify_variables, differentiate
from pyomo.dae import (ContinuousSet, DerivativeVar)
import numpy as np
import re

m = ConcreteModel()
m.scaling_factor = Suffix(direction=Suffix.EXPORT)
m.time = ContinuousSet(initialize = np.linspace(0,25,10+1))
dt = 25/(10+1)/3
# Writing the code for variable instantiation
m.u = Var(m.time, initialize = 2.0, within = Reals)
m.scaling_factor[m.u] = 1/max(1.0,2.0)
m.x = Var(m.time, initialize = 1.0, within = Reals)
m.scaling_factor[m.x] = 1/max(1.0,1.0)
m.y1 = Var(m.time, initialize = 30.0, within = Reals)
m.scaling_factor[m.y1] = 1/max(50.0,30.0)
m.y2 = Var(m.time, initialize = 15.0, within = Reals)
m.scaling_factor[m.y2] = 1/max(30.0,15.0)
# Instantiating the variables for the time derivatives
m.DERx = DerivativeVar(m.x, initialize = 0.4)
m.scaling_factor[m.DERx] = m.scaling_factor[m.x]*dt
# Applying time discretization through direct collocation to the problem
discretizer = TransformationFactory('dae.collocation')
discretizer.apply_to(m, nfe=10, ncp=3, scheme='LAGRANGE-RADAU')
timeSteps = [h for h in m.time]
# Instantiating the problem constraints
def _constr1(m,t):
return m.y2[t] == 0.5 * m.y1[t]
m.constr1 = Constraint(m.time, rule = _constr1)
m.u.fix(2.0)
def _constr2(m,t):
if t == 0:
return Constraint.Skip
return 5.0 * m.DERx[t] == m.u[t] - m.x[t] + (m.u[t] - m.x[t]) ** 2.0
m.constr2 = Constraint(m.time, rule = _constr2)
def _constr3(m,t):
return m.y1[t] == 30.0 * m.x[t]
m.constr3 = Constraint(m.time, rule = _constr3)
m.obj = Objective(expr = 1, sense = minimize)
# Code for the initial conditions of the dynamic optimization problem
def _init(m):
yield m.x[0] == 1.0
yield ConstraintList.End
m.init_conditions = ConstraintList(rule=_init)
# Constraint normalization
for constr in m.component_objects(Constraint, active=True):
varList = list(identify_variables(constr[list(constr.keys())[0]].body))
listF = []
for var in varList:
varName = re.sub(r'\[.*\]', '', str(var))
varName = "m." + varName
listF.append(abs(differentiate(constr[list(constr.keys())[0]].body, wrt=var)/m.scaling_factor[eval(
    varName)]))
m.scaling_factor[constr] = 1/max(listF)
# Solver settings and problem solution
scaled_model = TransformationFactory('core.scale_model').create_using(m)
solver = SolverFactory("ipopt")
results = solver.solve(scaled_model, tee=True)
TransformationFactory('core.scale_model').propagate_solution(scaled_model, m)
```


DOI 10.3384/ecp218117

Enhancing Collocation-Based Dynamic Optimization through Adaptive Mesh Refinement

Linus Langenkamp [1] Bernhard Bachmann [1]

[1]Institute for Data Science Solutions, Bielefeld University of Applied Sciences and Arts, Germany,
{first.last}@hsbi.de

Abstract

Direct collocation-based dynamic optimization plays an important role in the optimization of equation-based models. With this approach, continuous problems are transcribed into sparse nonlinear programs (NLPs) that can be solved efficiently. The open-source Modelica environment OpenModelica provides an implementation using Radau IIA collocation, but has major limitations, such as the lack of parameter optimization, no adaptive mesh refinement, and no support for higher-order integration schemes. This paper presents (1) a comprehensive reimplementation that addresses these limitations and (2) a novel h-method mesh refinement algorithm. Implemented in the custom Python / C++ optimization framework GDOPT, the approach demonstrates significant performance improvements, solving typical problems 2 to 3 times faster than OpenModelica under equivalent conditions. Using the proposed mesh refinement algorithm, the framework correctly identifies non-smooth regions and increases resolution accordingly, requiring only a small increase in computation time. The implementation lays the foundation for a future integration into the OpenModelica toolchain.

Keywords: Dynamic Optimization, Direct Collocation, Adaptive Mesh Refinement, Nonlinear Programming

1 Introduction

Dynamic Optimization has application in many different fields, including engineering, systems biology, economics, aerospace engineering and other disciplines where physical models are described by differential-algebraic equations (DAEs). A widely adopted strategy for solving such problems is direct collocation, a method that discretizes the time horizon and replaces the states by polynomials that satisfy the differential equations at discrete nodes. This approach leads to a sparse nonlinear program (NLP) that can be solved with state-of-the-art nonlinear optimizers. The accuracy of the approach relies on a well-designed discretization mesh. While smooth trajectories can be approximated with fewer intervals or higher degree polynomials, non-smooth behavior, e.g. rapid transitions, switching points or nonlinearities, demands high resolution and fine meshes. Adaptive mesh refinement methods, including h- (interval length adjustment), p- (polynomial degree adjustment), and hp-strategies (hybrid), are essential to efficiently obtain accurate solutions and are an active area of research (Liu, Hager, and Rao 2015; M. A. Patterson, Hager, and Rao 2015; Jain and Tsiotras 2008; Y. Zhao and Tsiotras 2009; J. Zhao and Shang 2018).

Several free and commercial tools implement direct collocation methods, including GPOPS-II (M. A. Patterson and Rao 2014), PSOPT (Becerra 2010), SPARTAN (Sagliano et al. 2018), JModelica (Magnusson and Åkesson 2015), and OpenModelica (Ruge et al. 2014; Fritzson, Pop, Abdelhak, et al. 2020). Among them, OpenModelica offers a fully open-source environment that combines expressive equation-based modeling in the Modelica language with optimization capabilities in a single tool. However, its current direct collocation implementation, using Radau IIA methods, has significant limitations, as it lacks support for adaptive mesh refinement, parameter optimization, higher-order collocation schemes and does not use exact second-order derivatives.

To address these limitations, this work proposes a novel h-method mesh refinement algorithm, implemented in a newly developed open-source framework called GDOPT, which is designed for future integration into the OpenModelica toolchain. The primary objective of this paper is to develop a general purpose mesh refinement algorithm that can accelerate the convergence of direct collocation as well as correctly identify non-smooth sections of control trajectories and increase the resolution appropriately.

This paper is organized as follows: Section 2 provides a general introduction and presents the mathematical formulation of dynamic optimization using direct collocation. Based on these considerations, we propose our novel mesh refinement algorithm L2-Boundary-Norm in Section 3, detailing its principal components and properties. Section 4 discusses the direct collocation implementation, focusing on its most important features and workflows. To demonstrate the capabilities of the algorithm and our implementation, Section 5 presents numerical results for a representative control problem, the fuel-optimal start-up of a diesel-electric power train (Sivertsson and Eriksson 2012; Mengist, Ruge, Gebremedhin, et al. 2013; Bachmann et al. 2012), comparing these results with the current OpenModelica implementation. Section 6 examines the limitations of both the framework and mesh refinement algorithm, while Section 7 summarizes the work and pro-

vides an outlook for future work.

2 Dynamic Optimization

2.1 Problem Definition

Consider the dynamic optimization problem

$$\min_{\boldsymbol{u}(t),\boldsymbol{p}} \; M(\boldsymbol{x}(t_f),\boldsymbol{u}(t_f),\boldsymbol{p},t_f) + \int_{t_0}^{t_f} L(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p},t)\,\mathrm{d}t \tag{1a}$$

s.t.

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p},t) \qquad \forall t \in T \tag{1b}$$

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{1c}$$

$$\boldsymbol{g}^L \le \boldsymbol{g}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p},t) \le \boldsymbol{g}^U \qquad \forall t \in T \tag{1d}$$

$$\boldsymbol{r}^L \le \boldsymbol{r}(\boldsymbol{x}(t_f),\boldsymbol{u}(t_f),\boldsymbol{p},t_f) \le \boldsymbol{r}^U \tag{1e}$$

$$\boldsymbol{a}^L \le \boldsymbol{a}(\boldsymbol{p}) \le \boldsymbol{a}^U \tag{1f}$$

over a fixed time horizon $T = [t_0, t_f]$ with time $t \in T$, referred to as the General Dynamic Optimization Problem (GDOP). The goal is to determine optimal control trajectories $\boldsymbol{u}(t) : T \to \mathbb{R}^{d_{\boldsymbol{u}}}$, time-invariant parameters $\boldsymbol{p} \in \mathbb{R}^{d_{\boldsymbol{p}}}$, and states $\boldsymbol{x}(t) : T \to \mathbb{R}^{d_{\boldsymbol{x}}}$ minimizing the cost functional (1a), subject to the constraints (1b) - (1f). The objective is divided into the *Mayer term* $M(\cdot)$, penalizing the final system state, and the *Lagrange term* $\int_{t_0}^{t_f} L(\cdot)\,\mathrm{d}t$, accounting for accumulated cost over time. The constraints include an initial value problem with *dynamic constraints* (1b) and an *initial condition* (1c), along with algebraic *path constraints* (1d), *final constraints* (1e), and *parametric constraints* (1f). All involved functions $M, L : \mathbb{R}^d \times T \to \mathbb{R}$, $\boldsymbol{f} : \mathbb{R}^d \times T \to \mathbb{R}^{d_{\boldsymbol{x}}}$, $\boldsymbol{g} : \mathbb{R}^d \times T \to \mathbb{R}^{d_{\boldsymbol{g}}}$, $\boldsymbol{r} : \mathbb{R}^d \times T \to \mathbb{R}^{d_{\boldsymbol{r}}}$, and $\boldsymbol{a} : \mathbb{R}^{d_{\boldsymbol{p}}} \to \mathbb{R}^{d_{\boldsymbol{a}}}$ with $d := d_{\boldsymbol{x}} + d_{\boldsymbol{u}} + d_{\boldsymbol{p}}$, are assumed to be twice continuously differentiable. The constraint bounds are given by $\boldsymbol{g}^L, \boldsymbol{g}^U \in \bar{\mathbb{R}}^{d_{\boldsymbol{g}}}$, $\boldsymbol{r}^L, \boldsymbol{r}^U \in \bar{\mathbb{R}}^{d_{\boldsymbol{r}}}$, and $\boldsymbol{a}^L, \boldsymbol{a}^U \in \bar{\mathbb{R}}^{d_{\boldsymbol{a}}}$ with $\bar{\mathbb{R}} := (\mathbb{R} \cup \{-\infty, \infty\})$. This formulation extends the Nonlinear Optimal Control Problem (NOCP) implemented in OpenModelica (Ruge et al. 2014) by including static parameters, which may be present in all model functions.

2.2 Direct Collocation

The continuous GDOP is reduced to an NLP by embedding a collocation scheme based on *flipped Legendre-Gauss-Radau (fLGR)* points. This discretization is equivalent to a transcription using the Radau IIA Runge-Kutta method, where the fLGR collocation nodes are scaled from $[-1, 1]$ to $[0, 1]$. For $[0, 1]$, the nodes c_j, $j = 1, \ldots, m$ are the m roots of the polynomial $(1-t)P^{(1,0)}_{m-1}(2t-1)$, where $P^{(1,0)}_{m-1}$ is an $(m-1)$-th Jacobi polynomial. The Radau IIA method has excellent stability properties, since it is A-, B-, and L-stable, and achieves order $2m-1$ for m collocation nodes.

The time horizon $[t_0, t_f]$ is divided into $n+1$ intervals $[t_i, t_{i+1}]$, $i = 0, \ldots, n$ with lengths $\Delta t_i := t_{i+1} - t_i$. On each interval $[t_i, t_{i+1}]$, collocation nodes $t_{ij} := t_i + c_j \Delta t_i$, $j = 1, \ldots, m_i$, along with the first grid point $t_{i0} := t_i + c_0 \Delta t_i$, where $c_0 = 0$, are introduced. Since $c_{m_i} = 1$ is always included in the Radau IIA scheme, the last grid point of interval $i-1$ matches the first of interval i, i.e. $t_{i-1,m_{i-1}} = t_{i0}$.

States and controls are approximated as $\boldsymbol{x}(t_{ij}) \approx \boldsymbol{x}_{ij}$ and $\boldsymbol{u}(t_{ij}) \approx \boldsymbol{u}_{ij}$, with the initial state fixed as $\boldsymbol{x}_{00} := \boldsymbol{x}_0$. In interval i, the state $\boldsymbol{x}(t)$ is replaced by a Lagrange interpolating polynomial $\boldsymbol{x}_i(t) = \sum_{j=0}^{m_i} \boldsymbol{x}_{ij}\, l_j(t)$ of degree m_i, where

$$l_j(t) := \prod_{\substack{k=0 \\ k \neq j}}^{m_i} \frac{t - t_{ik}}{t_{ij} - t_{ik}} \quad \forall j = 0, \ldots, m_i. \tag{2}$$

This polynomial must satisfy the differential equation (1b) at the collocation nodes t_{ij} for $j = 1, \ldots, m_i$ and match the initial condition $\boldsymbol{x}_{i0}$ from interval $i-1$. These conditions yield

$$\boldsymbol{0} = \begin{bmatrix} D^{(1)}_{10} I & \ldots & D^{(1)}_{1m_i} I \\ \vdots & \ddots & \vdots \\ D^{(1)}_{m_i 0} I & \ldots & D^{(1)}_{m_i m_i} I \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{i0} \\ \vdots \\ \boldsymbol{x}_{im_i} \end{bmatrix} - \Delta t_i \begin{bmatrix} \boldsymbol{f}_{i1} \\ \vdots \\ \boldsymbol{f}_{im_i} \end{bmatrix}, \tag{3}$$

with identity matrix $I \in \mathbb{R}^{d_{\boldsymbol{x}} \times d_{\boldsymbol{x}}}$, $\boldsymbol{f}_{ij} := \boldsymbol{f}(\boldsymbol{x}_{ij}, \boldsymbol{u}_{ij}, \boldsymbol{p}, t_{ij})$, and the first differentiation matrix $D^{(1)}_{jk} := \frac{\mathrm{d}\tilde{l}_k}{\mathrm{d}\tau}(c_j)$, where

$$\tilde{l}_k(\tau) := \prod_{\substack{j=0 \\ j \neq k}}^{m_i} \frac{\tau - c_j}{c_k - c_j} \quad \forall k = 0, \ldots, m_i. \tag{4}$$

Using the formulas provided in (Schneider and Werner 1986), these matrices can be computed efficiently *a priori*.

The Lagrange term is approximated via the corresponding Radau quadrature rule

$$\int_{t_0}^{t_f} L(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p},t)\,\mathrm{d}t \approx \sum_{i=0}^{n} \Delta t_i \sum_{j=1}^{m_i} b_j L_{ij}, \tag{5}$$

where $L_{ij} := L(\boldsymbol{x}_{ij}, \boldsymbol{u}_{ij}, \boldsymbol{p}, t_{ij})$ and the quadrature weights are given by

$$b_j = \int_0^1 \prod_{\substack{k=1 \\ k \neq j}}^{m_i} \frac{\tau - c_k}{c_j - c_k}\,\mathrm{d}\tau \quad \forall j = 1, \ldots, m_i. \tag{6}$$

2.3 Resulting NLP

Discretizing the remaining components of the GDOP is straightforward. The path constraints are replaced by constraints, which are evaluated at all grid points t_{ij} and the Mayer term and final constraints are approximated by an evaluation at the final grid point t_{nm_n}. The time-invariant parameters and their parametric constraints need not be

DOI 10.3384/ecp218127

discretized, since these are discrete. By flattening the collocated dynamics (3), the discretized GDOP becomes

$$\min_{\boldsymbol{x}_{ij},\boldsymbol{u}_{ij},\boldsymbol{p}} M(\boldsymbol{x}_{nm_n},\boldsymbol{u}_{nm_n},\boldsymbol{p},t_{nm_n}) + \sum_{i=0}^{n}\Delta t_i \sum_{j=1}^{m_i} b_j L(\boldsymbol{x}_{ij},\boldsymbol{u}_{ij},\boldsymbol{p},t_{ij}) \tag{7a}$$

s.t.

$$\boldsymbol{0} = \sum_{k=0}^{m_i} D^{(1)}_{jk}\boldsymbol{x}_{ik} - \Delta t_i \boldsymbol{f}(\boldsymbol{x}_{ij},\boldsymbol{u}_{ij},\boldsymbol{p},t_{ij}) \tag{7b}$$

$$\boldsymbol{x}_{00} = \boldsymbol{x}_0 \tag{7c}$$

$$\boldsymbol{g}^L \le \boldsymbol{g}(\boldsymbol{x}_{ij},\boldsymbol{u}_{ij},\boldsymbol{p},t_{ij}) \le \boldsymbol{g}^U \tag{7d}$$

$$\boldsymbol{r}^L \le \boldsymbol{r}(\boldsymbol{x}_{nm_n},\boldsymbol{u}_{nm_n},\boldsymbol{p},t_{nm_n}) \le \boldsymbol{r}^U \tag{7e}$$

$$\boldsymbol{a}^L \le \boldsymbol{a}(\boldsymbol{p}) \le \boldsymbol{a}^U \tag{7f}$$

where the dynamic constraints (7b) and path constraints (7d) must be satisfied for all $i = 0,\dots,n$ and $j = 1,\dots,m_i$.

This sparse, large-scale NLP can be implemented and solved efficiently with state-of-the-art nonlinear optimizers such as Ipopt (Wächter and Biegler 2006) or SNOPT (P. E. Gill et al. 2007; Philip E. Gill, Murray, and Saunders 2005), which require first and second order derivatives of the objective and constraints. By properly sorting the constraint vector, the cyclic block structure in the Jacobian and Hessian can be exploited. A detailed overview of the resulting blocks is given in (Langenkamp 2024).

3 Adaptive Mesh Refinement

An important modeling decision is the choice of grid points, i.e. the mesh $\mathcal{M} = \{t_0, t_1, \dots, t_n\}$, and the number of collocation nodes m_i per interval. Without prior knowledge of the problem, one might use an equidistant mesh $\Delta t_i \equiv$ const and a fixed number of nodes $m_i \equiv$ const. However, this often leads to large errors and unrealistic results, especially when the optimal solution exhibits discontinuities, switches, or steep gradients. Furthermore, the optimization often takes a significant amount of computation time due to poor initial guesses.

To address this, direct collocation methods typically employ adaptive mesh refinement algorithms, which balance accuracy and efficiency. In adaptive refinement, an initial NLP is solved on a coarse mesh, then the mesh is refined, and the NLP is solved again using updated initial guesses. This process continues until a termination criterion is met. The main classes of mesh refinement methods are h-, p-, and hp-methods.

h-methods (J. Zhao and Shang 2018; Y. Zhao and Tsiotras 2009; Jain and Tsiotras 2008; Betts and Huffman 1998) divide the time horizon into many intervals and use low-degree polynomials on each, i.e. $m_i \equiv m$. Convergence is achieved by increasing the number of intervals based on error criteria. These methods are robust and effective for non-smooth solutions but converge only polynomially fast for smooth problems. Several h-methods have been described in the literature. For example, in (J. Zhao and Shang 2018) a *multiresolution technique* is proposed that compares the current solution with a smoothed trajectory using non-oscillatory (ENO) interpolation. It detects non-smooth regions and merges intervals with constant control. Another method, presented in (Y. Zhao and Tsiotras 2009), encodes the curvature of the control trajectory into a density function that governs the distribution of mesh points, effectively capturing irregularities.

p-methods usually use a single interval and high-degree global polynomials. For smooth problems, they achieve *spectral*, i.e. exponential, convergence, but for non-smooth cases, errors grow significantly.

Modern hp- or ph-adaptive methods (M. A. Patterson and Rao 2014; M. A. Patterson, Hager, and Rao 2015; Liu, Hager, and Rao 2015; Sagliano et al. 2018; Garg, M. Patterson, Francolin, et al. 2011) combine both strategies. They adapt both the number of intervals and polynomial degrees. These hybrid methods also provide spectral convergence and are implemented in state-of-the-art tools such as GPOPS II (M. A. Patterson and Rao 2014). However, they may still struggle to detect switches and often generate large meshes with many collocation points (J. Zhao and Shang 2018; M. A. Patterson, Hager, and Rao 2015).

3.1 L2-Boundary-Norm

In this paper, the novel h-method *L2-Boundary-Norm (L2BN)* is proposed, which uniformizes the control trajectory in a similar way to density function-based approaches (Y. Zhao and Tsiotras 2009). L2BN applies consecutive bisection mesh refinement, following a workflow similar to the *multiresolution* technique (J. Zhao and Shang 2018), but without removing mesh points to maintain maximum stability. The method is designed to detect discontinuities, kinks, bends, and steep sections, while requiring limited computation time. L2BN is divided into 2 distinct phases:

3.2 Phase I

Phase I is used to accelerate the optimization by performing a simple multigrid refinement. Here the possibly poor initial guess of the control trajectory $\boldsymbol{u}_{init}(t)$, e.g. constant or linear, is only used for an initial optimization on an extremely coarse mesh. To ensure that the state trajectory guess $\boldsymbol{x}_{init}(t)$ is feasible, i.e. satisfies the initial value problem (1b) - (1c), the initial states are obtained by performing a simulation using the provided guesses $\boldsymbol{u}_{init}(t)$ and $\boldsymbol{p}_{init}$.

The initial optimization is exceptionally fast compared to one using the poor guess on a fine mesh. The optimal solution of the problem is then interpolated onto a mesh, where every interval has been bisected. This interpolated solution is then used as a new, more appropriate initial guess and optimized again. Repeating this process several times, yields a realistic, equidistant optimal solution

that solely suffers from steep sections, kinks or jump discontinuities. Overall, the first phase can be expressed as Algorithm 3.1:

Algorithm 3.1: Phase I: Multigrid Refinement

Input: GDOP, equidistant mesh $\mathcal{M}_0$, number of iterations k_{max}, initial guess on the control trajectory $\boldsymbol{u}_{init}(t)$ and parameters $\boldsymbol{p}_{init}$
Output: Equidistant solutions $\boldsymbol{x}^*(t)$, $\boldsymbol{u}^*(t)$, $\boldsymbol{p}^*$

$\boldsymbol{x}_{init}(t) \leftarrow$ Solve the ODE $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\cdot), \boldsymbol{x}(t_0) = \boldsymbol{x}_0$ using the guesses $\boldsymbol{u}_{init}(t), \boldsymbol{p}_{init}$
for $k = 0, \ldots, k_{max}$ **do**
  $\boldsymbol{x}_k^*(t), \boldsymbol{u}_k^*(t), \boldsymbol{p}_k^* \leftarrow$ Solve the NLP (7a) - (7f) on $\mathcal{M}_k$ using $\boldsymbol{x}_{init}(t), \boldsymbol{u}_{init}(t), \boldsymbol{p}_{init}$ as initial guesses
  if $k = k_{max}$ **then**
    return $\boldsymbol{x}_k^*(t), \boldsymbol{u}_k^*(t), \boldsymbol{p}_k^*$
  end
  $\mathcal{U}_k \leftarrow \left\{ \frac{t_i + t_{i+1}}{2} \mid i = 0, \ldots, |\mathcal{M}_k| - 1 \right\}$
  $\mathcal{M}_{k+1} \leftarrow \mathcal{M}_k \cup \mathcal{U}_k$
  $\boldsymbol{p}_{init} \leftarrow \boldsymbol{p}_k^*$
  $\boldsymbol{x}_{init}(t), \boldsymbol{u}_{init}(t) \leftarrow$ Interpolate $\boldsymbol{x}_k^*(t), \boldsymbol{u}_k^*(t)$ onto $\mathcal{M}_{k+1}$
end

3.2.1 Illustration of Phase I

To further illustrate phase I, we consider the *Oil Shale Pyrolysis*[1], studied in (Wen and Yen 1977) and (Langenkamp 2024), aiming to find an optimal temperature control. Using a poor, constant initial temperature guess two approaches are compared. The *Default* method involves optimization on a mesh with $|\mathcal{M}_0| = 80$ intervals and $m = 3$ collocation nodes. Alternatively, Algorithm 3.1 (Phase I) starts with a much coarser initial mesh of 5 intervals and performs $k_{max} = 4$ refinement iterations. Both methods ultimately yield the same optimal solution.

Method	$\lvert\mathcal{M}_0\rvert$	m	k_{max}	t_{opt}	Iter
Default	80	3	0	0.3492	161
Algorithm 3.1	5	3	4	0.1242	105

Table 1. Performance Evaluation of Phase I Refinement for an Oil Shale Pyrolysis Optimization

As detailed in Table 1, the optimization using Ipopt and MUMPS demonstrates that Algorithm 3.1 required fewer NLP iterations (Iter) and approximately one-third of the computation time t_{opt} in seconds compared to the default implementation. This significant speedup is attributed to the multigrid refinement strategy: instead of optimizing with a poor initial guess on a large mesh, the guesses progressively improve through successive refinements. Figure 1 depicts this process, showing how the temperature control trajectory evolves and approaches the final solution across refinement iterations.

[1]The GDOPT implementation can be found under `https://github.com/linuslangenkamp/GDOPT/blob/master/examples/oilShalePyrolysis.py`.

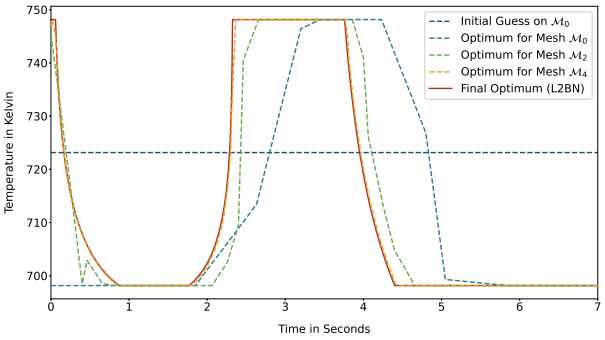


Figure 1. Multigrid refinement applied to an Oil Shale Pyrolysis (Wen and Yen 1977). The optimal solution after performing 3 additional phase II iterations is given as reference.

3.3 Phase II

Phase II addresses the problem of remaining steep sections, kinks or switches and iteratively refines the solution of phase I by investigating certain L^2-norms of the control trajectory (*on-interval condition*) as well as error terms of the trajectories on the boundary (*boundary condition*). It identifies non-smooth behavior and splits intervals, which violate the conditions, such that resolution is increased appropriately and switching times are identified correctly.

3.4 On-Interval Condition

The idea behind the *on-interval condition* is to individually inspect the current solution of the control variables $\boldsymbol{u}_{ij}$ within each mesh interval $[t_i, t_{i+1}]$ of the mesh $\mathcal{M}_k$, and to measure the *slope* and *curvature* of the control variables over the full interval. If these measures exceed a certain threshold, the interval is bisected, and the midpoint $\frac{t_i + t_{i+1}}{2}$ is added to the next mesh $\mathcal{M}_{k+1}$.

Consider an arbitrary interval $[t_i, t_{i+1}]$ from the mesh $\mathcal{M}_k$ and the current optimal solution of the l-th control variable $u^{(l)}$ on this interval. This solution is defined by $m+1$ sample points $u_{ij}^{(l)}$, $j = 0, \ldots, m$. The first step is to construct an interpolating polynomial $p_i^{(l)}(t)$ for the control variable $u^{(l)}$ on the normalized interval $[0, 1]$, such that $p_i^{(l)}(c_j) = u_{ij}^{(l)}$, $j = 0, \ldots, m$, with $p_i^{(l)} \in P^m$. This rescaling is performed because the condition should not be influenced by the length of the interval Δt_i and, moreover, only the magnitude of the controls should be relevant. In order to estimate the change on the interval, it is natural to consider a function norm of $\dot{p}_i^{(l)}$ and $\ddot{p}_i^{(l)}$ such as L^1, L^2 or L^∞. The L^2 norm

$$\|p\| := \sqrt{\int_0^1 p(\tau)^2 \, \mathrm{d}\tau} \tag{8}$$

is chosen, since it is a good middle ground and allows for

DOI 10.3384/ecp218127

a fast computation of the condition. All in all, the *on-interval condition* says:

Condition 3.1 (On-Interval Condition)**.** *If for a given interval* $[t_i, t_{i+1}]$ *there exists at least one control variable* $u^{(l)}$ *such that the interpolating polynomial* $p_i^{(l)}$ *on the nominal interval* $[0,1]$ *satisfies*

$$\left\|\dot{p}_i^{(l)}\right\| > TOL_1^{(l)} \quad \vee \quad \left\|\ddot{p}_i^{(l)}\right\| > TOL_2^{(l)},$$

for specified tolerances $TOL_1^{(l)}, TOL_2^{(l)} > 0$*, then the interval is bisected.*

3.4.1 Fast Computation

The following procedure enables an efficient and exact evaluation of the *on-interval condition* for any number of collocation nodes m. By construction, it holds that $\deg\left(\dot{p}_i^{(l)}(\tau)\right) = m-1$, $\deg\left(\ddot{p}_i^{(l)}(\tau)\right) = m-2$, which implies $\deg\left(\dot{p}_i^{(l)}(\tau)^2\right) = 2m-2$, $\deg\left(\ddot{p}_i^{(l)}(\tau)^2\right) = 2m-4$ for all $m \geq 2$. In the special case of $m = 1$, the second derivative of $p_i^{(l)}$ vanishes and the first derivative is constant. Since Radau quadrature with m collocation nodes integrates all polynomials of degree up to $2m-2$ exactly, both norms can be exactly evaluated using the quadrature rule. For simplicity, the calculations are only shown for the first derivative. These are analogous for the second derivative. Hence

$$\left\|\dot{p}_i^{(l)}\right\| = \sqrt{\int_0^1 \dot{p}_i^{(l)}(\tau)^2 \,\mathrm{d}\tau} = \sqrt{\sum_{k=1}^{m} b_k \dot{p}_i^{(l)}(c_k)^2}, \qquad (9)$$

where b_k are the Radau quadrature weights (6). By writing $p_i^{(l)}(\tau) = \sum_{j=0}^{m} u_{ij}^{(l)} l_j(\tau)$ as a Lagrange interpolating polynomial, we get $\dot{p}_i^{(l)}(c_k) = \sum_{j=0}^{m} u_{ij}^{(l)} D_{kj}^{(1)}$ using the first differentiation matrix. Finally, it follows that

$$\left\|\dot{p}_i^{(l)}\right\| = \sqrt{\sum_{k=1}^{m} b_k \left(\sum_{j=0}^{m} u_{ij}^{(l)} D_{kj}^{(1)}\right)^2}. \qquad (10)$$

Since applying the first differentiation matrix twice yields the second derivatives at the collocation nodes, a similar formula can be calculated for $\left\|\ddot{p}_i^{(l)}\right\|$. Therefore, we obtain Algorithm 3.2 by utilizing matrix-vector notation.

Because the algorithm has to be applied for each mesh interval and control variable, the overall runtime of the condition becomes $O(d_{\boldsymbol{u}} n m^2)$, which is just m times the number of discretized controls $d_{\boldsymbol{u}} n m$. This runtime is neglectable, compared to the several expensive NLP iterations that are performed for each mesh iteration.

3.4.2 Termination

An important idea of the Phase II iteration is that the final solution should satisfy Condition 3.1 for all intervals and control variables. This condition ensures that the local variability (slope and curvature) of the control trajectories remains below specific tolerances across all intervals. When this is achieved, the mesh points become equidistributed relative to this local variability. Since Condition 3.1 takes the same form as conditions used to define mesh density functions (e.g. ensuring an integral of some monitor function over each interval is below a tolerance), the resulting mesh distribution is at least as strong as, and can be characterized by, the density function implicitly defined by the satisfied condition itself. The derivation of this implicit density function is detailed in (Langenkamp 2024). Because of this, it is useful to have a criterion for the termination of a sequence of *on-interval* iterations. A sufficient condition for this termination can be deduced from Lemma 1, where the L^2 norm is again taken on $[0,1]$:

Algorithm 3.2: Fast *On-Interval* Computation

Input: Sample values $\hat{\boldsymbol{u}} = \left(u_{i0}^{(l)}, u_{i1}^{(l)}, \ldots, u_{im}^{(l)}\right)^T$, m-step Radau quadrature weights $\boldsymbol{b}$, differentiation matrix $D^{(1)}$

Output: $\left\|\dot{p}_i^{(l)}\right\|, \left\|\ddot{p}_i^{(l)}\right\|$

$\boldsymbol{p}' \leftarrow D^{(1)} \hat{\boldsymbol{u}}$

$\boldsymbol{q}' \leftarrow \left((p_1')^2, \ldots, (p_m')^2\right)^T$

$\left\|\dot{p}_i^{(l)}\right\| \leftarrow \sqrt{\boldsymbol{b}^T \boldsymbol{q}'}$

$\boldsymbol{p}'' \leftarrow D^{(1)} \boldsymbol{p}'$

$\boldsymbol{q}'' \leftarrow \left((p_1'')^2, \ldots, (p_m'')^2\right)^T$

$\left\|\ddot{p}_i^{(l)}\right\| \leftarrow \sqrt{\boldsymbol{b}^T \boldsymbol{q}''}$

return $\left\|\dot{p}_i^{(l)}\right\|, \left\|\ddot{p}_i^{(l)}\right\|$

Lemma 1. *Let* $p \in P^m$ *be a polynomial and* $p_\varepsilon(t) = p\left(\frac{t}{2}\right) + \varepsilon(t)$ *be a variation of* $p\left(\frac{t}{2}\right)$ *by* $\varepsilon \in P^m$*, then for a given* $\delta \in \mathbb{R}$ *with* $\frac{\sqrt{2}}{2} < \delta < 1$

$$\frac{\|\dot{p}_\varepsilon(t)\|}{\|\dot{p}(t)\|} < \delta, \quad \text{if} \quad \|\dot{\varepsilon}(t)\| < \frac{\sqrt{2}\delta - 1}{2} \left\|\dot{p}\left(\frac{t}{2}\right)\right\|.$$

A proof of Lemma 1 is given in the Appendix. Let $p(t)$ be a polynomial that violates Condition 3.1. The corresponding interval is bisected and both subintervals are examined in the next iteration. Without loss of generality, we consider the first subinterval, while the second follows analogously with $p_\varepsilon(t) = p\left(\frac{t+1}{2}\right) + \varepsilon(t)$. Rescaling the old polynomial from $[0, \frac{1}{2}]$ to $[0,1]$, as required by the *on-interval condition*, yields $p\left(\frac{t}{2}\right)$, so the new polynomial becomes $p_\varepsilon(t) = p\left(\frac{t}{2}\right) + \varepsilon(t)$, with a perturbation ε. By Lemma 1, the new norm on the subinterval is smaller if

$$\left\|\dot{p}_\varepsilon(t) - \dot{p}\left(\frac{t}{2}\right)\right\| = \|\dot{\varepsilon}(t)\| < \frac{\sqrt{2}\delta - 1}{2} \left\|\dot{p}\left(\frac{t}{2}\right)\right\|. \qquad (11)$$

Thus, we get a decreasing sequence of L^2-norms, provided the difference of the new and old polynomial derivatives is suitably bounded proportional to the derivative of the old polynomial itself. Assuming the convergence of the collocation method, the variations $\varepsilon, \dot{\varepsilon}$, and $\ddot{\varepsilon}$ should vanish for smooth problems. Since it is easy to find an analogous lemma for the second derivative with

$$\frac{\|\ddot{p}_\varepsilon(t)\|}{\|\ddot{p}(t)\|} < \delta, \text{ if } \|\ddot{\varepsilon}(t)\| < \frac{2\sqrt{2}\delta - 1}{4}\left\|\ddot{p}\left(\frac{t}{2}\right)\right\| \tag{12}$$

for $\frac{1}{2\sqrt{2}} < \delta < 1$, the same holds for $\ddot{p}$ in both subintervals. Hence, for smooth problems under suitable convergence assumptions, the refinement procedure terminates after finitely many iterations. However, for non-smooth solutions, e.g. bang-bang optimal controls, the method may never terminate, as the identical polynomial can be obtained on one of the subintervals, i.e. $p_\varepsilon(t) - p(t) \equiv 0$.

3.5 Boundary Condition

The described method could be implemented using only the *on-interval condition*. However, this approach has a critical weakness in detecting non-smooth behavior on the boundary of two adjacent intervals.

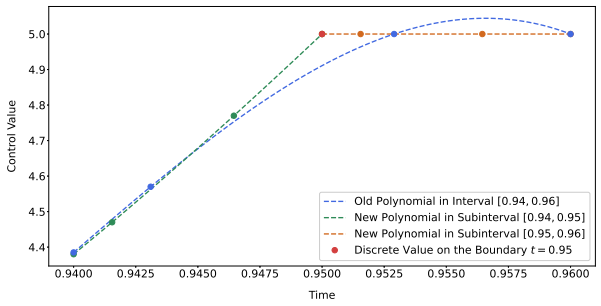


Figure 2. Poor corner detection

Consider Figure 2, where the interval $[0.94, 0.96]$ is split into $[0.94, 0.95]$ and $[0.95, 0.96]$ using the *on-interval condition*. The interpolating polynomials based on the 3-step Radau collocation method are shown. The first subinterval exhibits a steep slope throughout and will likely be bisected again. In contrast, the second subinterval appears nearly constant and will not trigger further refinement. However, this is problematic, as their shared boundary represents a sharp corner. To determine the switching time more accurately, the algorithm should further refine the mesh around this corner.

To resolve this, we introduce the *boundary condition*. It compares the first and second derivatives of adjacent polynomials at their shared boundary point. Rather than using purely relative or absolute error, we adopt the *plus-1 error* with $E(x,y) := \frac{|x-y|}{1+\min|x|,|y|}$, which combines both measures. Overall, the *boundary condition* says:

Condition 3.2 (Boundary Condition)**.** *If for two adjacent intervals i and $i+1$ there exists at least one control variable $u^{(l)}$, such that the interpolating polynomials $p_i^{(l)}, p_{i+1}^{(l)}$ on the nominal interval $[0,1]$ satisfy*

$$E(\dot{p}_i^{(l)}(t_{i+1}), \dot{p}_{i+1}^{(l)}(t_{i+1})) \geq CTOL_1^{(l)} \ \vee$$
$$E(\ddot{p}_i^{(l)}(t_{i+1}), \ddot{p}_{i+1}^{(l)}(t_{i+1})) \geq CTOL_2^{(l)}$$

for the common boundary point t_{i+1} and specified corner tolerances $CTOL_1^{(l)}, CTOL_2^{(l)} > 0$, then both intervals are bisected.

Evaluating the condition is inexpensive, since the values are side products of Algorithm 3.2.

3.6 Algorithm

Based on the previous considerations the full mesh refinement algorithm L2-Boundary-Norm (L2BN) can be formulated. By default, the constant corner tolerances are given as $CTOL_1^{(l)} = CTOL_2^{(l)} = 0.1$, but can be adjusted for a specific problem. The default tolerances for Condition 3.1 are given by

$$TOL_1^{(l)} = \frac{\text{range}(u^{(l)})}{|\mathcal{M}_0|} 10^{-\Lambda} \tag{13a}$$

$$TOL_2^{(l)} = \frac{\text{range}(u^{(l)})}{2|\mathcal{M}_0|} 10^{-\Lambda}, \tag{13b}$$

where $\text{range}(u^{(l)}) := \max_{t\in T} u^{(l)}(t) - \min_{t\in T} u^{(l)}(t)$ and $\Lambda \in \mathbb{R}$ is the level of refinement with default value $\Lambda = 0$.

Altogether we get Algorithm 3.3, which, for the b-th mesh refinement, requires $O(d_{\boldsymbol{u}} m^2 |\mathcal{M}_b|)$ operations. In this version of L2BN, Condition 3.1 enforces a bisection of the interval itself and both neighbors to prevent bisections in future iterations and enhance stability.

4 Implementation

The novel mesh refinement algorithm L2BN is integrated into a newly developed, open source dynamic optimization framework called GDOPT (General Dynamic Optimizer), which is publicly available on GitHub[2]. The framework is split into a Python modeling environment *gdopt*, which allows for expressive and accessible symbolic modeling of dynamic optimization problems and a C++ library *libgdopt* that performs the computationally intensive task of solving the large-scale NLPs with Ipopt (Wächter and Biegler 2006).

The framework offers a range of noteworthy features, of which the most relevant are presented in this paper. A comprehensive description of the GDOPT implementation can be found in (Langenkamp 2024) and an introduction to modeling with the framework is provided in the GDOPT User's Guide[3]. The principle workflows are illustrated in Figure 3. First, users can define their optimization problems in Python using purely symbolic ex-

[2] https://github.com/linuslangenkamp/GDOPT

[3] https://github.com/linuslangenkamp/GDOPT/blob/master/usersguide/usersguide.pdf

DOI 10.3384/ecp218127

Algorithm 3.3: L2-Boundary-Norm (L2BN)

Input: GDOP, phase I iteration count k_{max}, guesses $\boldsymbol{u}_{init}(t)$, $\boldsymbol{p}_{init}$, mesh $\mathcal{M}_0$, Radau IIA scheme with m nodes, phase II iteration count b_{max}, tolerances $TOL_1^{(l)}$, $TOL_2^{(l)}$, $CTOL_1^{(l)}$, $CTOL_2^{(l)}$

Output: $\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{p}^*$

$\boldsymbol{x}_0^*(t), \boldsymbol{u}_0^*(t), \boldsymbol{p}_0^* \leftarrow$ Perform k_{max} iterations of Algorithm 3.1 on $\mathcal{M}_0$ with m collocation nodes and guesses $\boldsymbol{u}_{init}(t)$, $\boldsymbol{p}_{init}$

for $b = 0, \ldots, b_{max} - 1$ **do**
  $\mathcal{U} \leftarrow \{\}$
  for $i = 0, \ldots, |\mathcal{M}_b| - 1$ **do**
    for $l = 1, \ldots, d_{\boldsymbol{u}}$ **do**
      if *Condition 3.1 is violated* **then**
        $\mathcal{U} \leftarrow \mathcal{U} \cup \{\max\{0, i-1\}, i, \min\{|\mathcal{M}_b| - 1, i+1\}\}$
        break
      end
      if $(i \neq |\mathcal{M}_b| - 1) \wedge$ *Condition 3.2 is violated* **then**
        $\mathcal{U} \leftarrow \mathcal{U} \cup \{i, i+1\}$
      end
    end
  end
  if $\mathcal{U} = \{\}$ **then**
    return $\boldsymbol{x}_b^*(t), \boldsymbol{u}_b^*(t), \boldsymbol{p}_b^*$
  end
  $\mathcal{U}_b \leftarrow$ Bisect the intervals in $\mathcal{U}$
  $\mathcal{M}_{b+1} \leftarrow \mathcal{M}_b \cup \mathcal{U}_b$
  $\boldsymbol{p}_{init} \leftarrow \boldsymbol{p}_b^*$
  $\boldsymbol{x}_{init}(t), \boldsymbol{u}_{init}(t) \leftarrow$ Interpolate $\boldsymbol{x}_b^*(t), \boldsymbol{u}_b^*(t)$ onto $\mathcal{M}_{b+1}$
  $\boldsymbol{x}_{b+1}^*(t), \boldsymbol{u}_{b+1}^*(t), \boldsymbol{p}_{b+1}^* \leftarrow$ Solve the NLP (7a) - (7f) on $\mathcal{M}_{b+1}$ using $\boldsymbol{x}_{init}(t)$, $\boldsymbol{u}_{init}(t)$, $\boldsymbol{p}_{init}$ as initial guesses
end
return $\boldsymbol{x}_{b_{max}}^*(t), \boldsymbol{u}_{b_{max}}^*(t), \boldsymbol{p}_{b_{max}}^*$

pressions for differential equations, constraints and objectives. The Python package *SymEngine*[4] is used to efficiently calculate the symbolic first and second derivatives and their common subexpressions to remove redundant calculations. These symbolic expressions are translated into efficient C++ code with exact sparsity patterns. After that, the code is compiled and linked with *libgdopt*, which implements the full collocation-based dynamic optimization pipeline, including discretization, solver interface, L2BN mesh refinement algorithm, and precomputed Radau IIA schemes for $m = 1, \ldots, 70$ accurate to machine precision.

GDOPT separates code generation and optimization, allowing parameters, solver settings, and initial guesses to be modified after compilation. This enables rapid repeated solves without regenerating and recompiling the model. At runtime, the executable reads all parameters, initial values, and solver settings, and solves the resulting NLP using Ipopt (Wächter and Biegler 2006) with a chosen linear solver, e.g. MUMPS (Amestoy et al. 2001) or one from the HSL suite (HSL 2013). Results are written to an output file, which is automatically read by the Python interface for post-processing and visualization using *matplotlib* (Hunter 2007).

4.1 Acceleration of Refinement Iterations

An important note on the implementation is, that Ipopt is an interior-point solver that uses logarithmic barriers. The implementation of L2BN heavily benefits from this concept, by setting the barrier parameter to a minimal value (e.g. $\mu_0 = 10^{-15}$) for repeated solves. This strategy significantly accelerates mesh refinement iterations even further, since the new initial guess, i.e. the previous optimal solution interpolated onto the new mesh, is known to be almost optimal by construction. Thus, Ipopt solves an unperturbed problem, avoiding unnecessary calculations and achieving rapid convergence.

[4] https://github.com/symengine/symengine

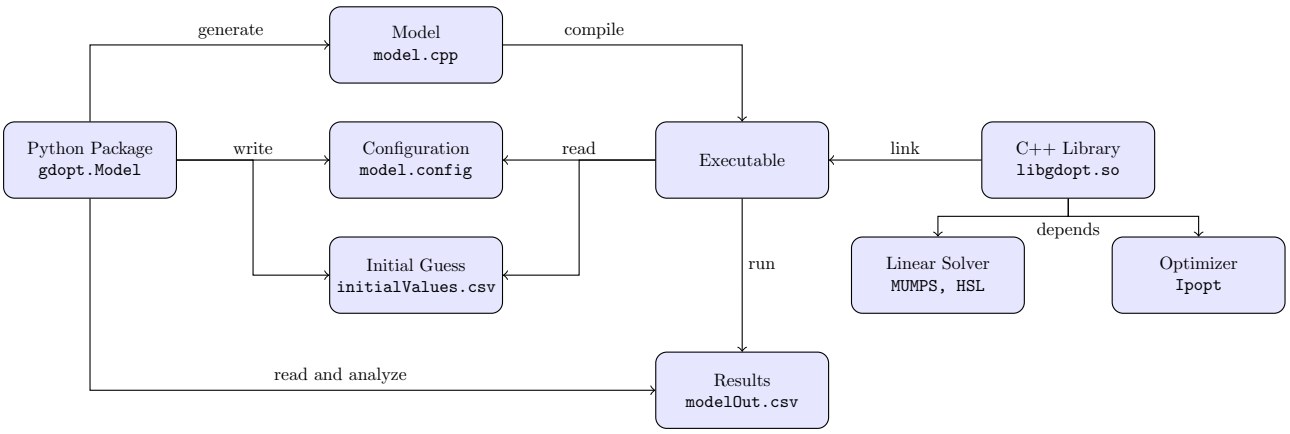


Figure 3. Principle workflows in GDOPT. For a detailed introduction to workflows in GDOPT, please refer to (Langenkamp 2024).

4.2 Optimization of DAEs

Currently, the framework does not support the direct handling of differential-algebraic equations (DAEs). While algebraic variables can be modeled as control inputs, this approach artificially bloats the NLP, making it more numerically unstable and computationally demanding. An integration of *libgdopt* into a proper modeling environment would clearly be beneficial. Previous work on OpenModelica (Ruge et al. 2014) has already demonstrated such an integration by performing a transformation to semi-explicit ODE form and solving for algebraic variables in each optimization step.

5 Results and Performance

To evaluate the performance of the proposed mesh refinement algorithm and the newly developed framework, a relevant dynamic optimization problem from the open literature is considered. For additional benchmarks of GDOPT, including demonstrations of the termination property for smooth problems, we refer to (Langenkamp 2024). All optimizations are performed on a laptop with a Intel Core i7-12800H, 32 GB RAM, running Ubuntu 24.04.2 and using GCC v13.3.0 with flags -O3 -ffast-math for compilation.

5.1 Diesel Electric Power Train

Consider the diesel-electric power train[5] that has been studied extensively in (Sivertsson and Eriksson 2012; Mengist, Ruge, Gebremedhin, et al. 2013; Bachmann et al. 2012). This optimization problem deals with the fuel optimal start-up of a diesel engine from an idling condition to a certain power level. The model features 2 control variables in relative units and 4 states, while each variable is associated with a given box constraint to guarantee physical validity. The diesel-electric power train has the following principle structure:

$$\min_{u_1(t),u_2(t)} \sum_{i=1}^{4} (x_i(t_f) - c_i)^2 \tag{14a}$$

s.t.

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} f_1(x_1, x_2, x_3, u_1) \\ f_2(x_1, x_2, x_4) \\ f_3(x_1, x_2, x_3, u_1, u_2) \\ f_4(x_1, x_2, x_3, x_4, u_1) \end{bmatrix} \forall t \in [t_0, t_f] \tag{14b}$$

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{14c}$$

$$\boldsymbol{x}^L \leq \boldsymbol{x} \leq \boldsymbol{x}^U \forall t \in [t_0, t_f] \tag{14d}$$

$$\mathbf{0} \leq \boldsymbol{u} \leq \mathbf{1} \quad \forall t \in [t_0, t_f] \tag{14e}$$

$$t_0 = 0, t_f = 0.5, \boldsymbol{c} \in \mathbb{R}^4 \text{ are constants.} \tag{14f}$$

The optimization problem is solved with both GDOPT and OpenModelica v1.24.0 using the same hardware and settings for various configurations. All runs use an Ipopt tolerance of 10^{-14}, the linear solver MUMPS, and identical constant initial guesses. All other solver parameters are left at default and the times are averaged over 5 runs.

[5]The GDOPT implementation can be found under `https://github.com/linuslangenkamp/GDOPT/blob/master/examples/dieselMotor.py`

5.2 Code Generation and Compilation

The code generation and compilation timings for GDOPT are given by: 0.0268 s for the derivative calculations and code generation, 1.1561 s for compiling the generated C++ code and 0.0335 s for obtaining the initial guess by simulating the dynamics. However, the OpenModelica compiler requires just 0.2011 seconds for the translation and compilation. Note that this overhead is independent of the specific configuration.

5.3 Results Without Mesh Refinement

For equidistant meshes with sizes $|\mathcal{M}_0| = 25, 100, 250$, the same optimization has been performed with both implementations and the results are displayed in Table 2.

Alg.	$\lvert\mathcal{M}_0\rvert$	m	$\phi^* \cdot 10^3$	t_{opt}	Speedup
OM	25	3	1.11171326	0.3592	1.0000
GD	25	3	1.11171326	0.1877	1.9136
OM	100	3	1.11155875	0.6730	1.0000
GD	100	3	1.11155875	0.3214	2.0940
OM	250	3	1.11155972	1.8256	1.0000
GD	250	3	1.11155972	0.6086	2.9997

Table 2. Performances of GDOPT (GD) and OpenModelica (OM) on equidistant meshes for the *Diesel Electric Power Train*. The columns are the algorithm (Alg.), initial mesh size ($|\mathcal{M}_0|$), number of collocation nodes (m), scaled final objective ($\phi^* \cdot 10^3$), total optimization time (t_{opt}), and speedup factor relative to OpenModelica.

It is noteworthy that GDOPT is 2 to 3 times faster than OpenModelica without using any mesh refinement. Moreover, both implementations provide the exact same objectives to all digits. This clearly shows that GDOPT efficiently deploys the relevant model information and additionally, performs internal calculations very fast.

5.4 Phase I Refinement

By incorporating $k = 2$ phase I refinements, the same optimal solution for an equidistant mesh size of $|\mathcal{M}| = 100$ is obtained in 0.2291 s as seen in Table 3, compared to 0.3214 s with GDOPT without refinement and 0.6730 s with OpenModelica (Table 2). Therefore, the multigrid refinement (Algorithm 3.1) accelerates the GDOPT procedure by roughly 40%.

5.5 Results Using L2-Boundary-Norm

Additional Phase II iterations enhance accuracy in regions with non-smooth control behavior. Since the optimal control remains constant over large parts of the time horizon, high resolution is not required everywhere. However, the problem also features sharp switching behavior, where accurately resolving the exact timing of the switches is crit-

DOI 10.3384/ecp218127

ical. This is illustrated in Figure 4, which shows the optimal control solution on the final refined mesh:

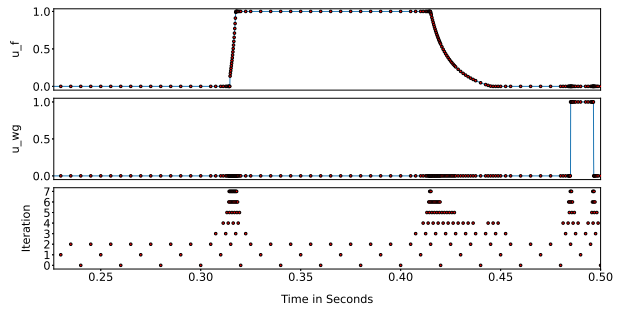


Figure 4. Mesh refinement history of the *Diesel Electric Power Train* for configuration: GDOPT, $k = 2$, $b = 5$, $|\mathcal{M}_0| = 25$, $m = 3$

To further investigate the impact of the refinement iterations, control trajectories in different stages of the refinement process are presented. In Figure 5 the controls are displayed for the configuration $k = 2$, $b = 5$, $|\mathcal{M}_0| = 25$, $m = 3$.

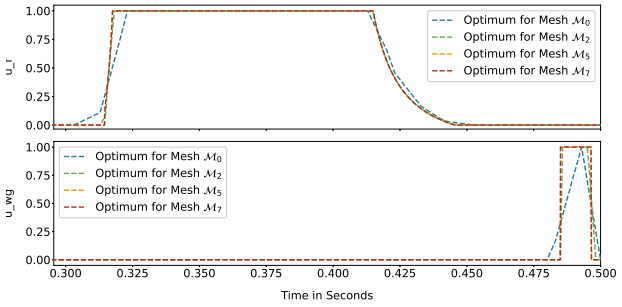


Figure 5. Refinement iterations for both control variables of the *Diesel Electric Power Train* for configuration: GDOPT, $k = 2$, $b = 5$, $|\mathcal{M}_0| = 25$, $m = 3$

Note that in this case the mesh subscript denotes the cumulative mesh iteration. By simply performing 2 multigrid iterations, the solution on $\mathcal{M}_2$ looks almost identical to the final solution using 5 additional phase II iterations. Nevertheless, by increasing the size of the plot, it becomes obvious that the switching times are not precisely located as seen for the second control variable in Figure 6. For this control, the placements of the switches are roughly $t \approx 0.485\,\mathrm{s}$ and $t \approx 0.496\,\mathrm{s}$ for $\mathcal{M}_2$, but the resolution is too low to capture the exact position of the switch. By performing additional iterations, the resolution of the switches increases drastically, such that the final solution on $\mathcal{M}_7$, which was obtained in just 0.9522 s (Table 3), describes a perfect rectangular shape. This final resolution is equivalent to that of an equidistant mesh with 3200 intervals. Performing the optimization on this grid without refinement takes 20.3825 s using GDOPT and 84.6323 s using OpenModelica. Furthermore, OpenModelica did not find the optimal solution but terminated with an oscillating solution.

Alg.	k, b	$\lvert\mathcal{M}_0\rvert$	m	$\lvert\mathcal{M}^*\rvert$	$\phi^* \cdot 10^3$	t_{opt}
GD	2,0	25	3	100	1.11155875	0.2291
GD	2,3	25	3	167	1.11155842	0.4301
GD	2,3	25	5	170	1.11155852	0.6849
GD	2,3	25	7	171	1.11155856	1.1687
GD	2,5	25	3	240	1.11155853	0.9522
GD	2,5	25	5	243	1.11155853	1.5512
GD	2,5	25	7	249	1.11155853	4.4676
GD	2,8	25	3	266	1.11155853	1.3536

Table 3. Performance of GDOPT (GD) using Algorithm 3.3 for the *Diesel Electric Power Train*. The columns are the algorithm (Alg.), maximum number of Phase I and II iterations (k, b), initial mesh size ($|\mathcal{M}_0|$), number of collocation nodes (m), final mesh size ($|\mathcal{M}^*|$), scaled final objective ($\phi^* \cdot 10^3$), and total optimization time (t_{opt}) in seconds.

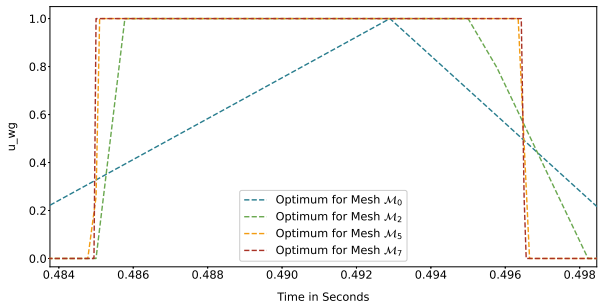


Figure 6. Refinement iterations for jump discontinuities of the *Diesel Electric Power Train* for configuration: GDOPT, $k = 2$, $b = 5$, $|\mathcal{M}_0| = 25$, $m = 3$

5.6 Resolution Under Runtime Constraints

To benchmark resolution efficiency, GDOPT was run on equidistant meshes under a fixed runtime limit of 1 s. The densest such mesh solved within this limit had 400 intervals and required 1.0273 s. In contrast, mesh refinement using L2BN with $k = 2$ Phase I and $b = 8$ Phase II iterations achieved an effective resolution equivalent to 25600 equidistant intervals in just 1.3536 s. For comparison, equidistant meshes with 550 and 600 intervals required 1.2349 s and 2.0783 s, respectively. These resolutions are over 40 times coarser than the refined result. This demonstrates that mesh refinement with L2BN significantly improves resolution, as needed for problems with switching controls, while maintaining low computational cost.

5.7 Discussion of Results

Overall, L2BN enables GDOPT to accelerate convergence from poor initial guesses, efficiently detect non-smooth regions, and adaptively increase resolution, far outperforming traditional equidistant mesh approaches. Combined with its rapid execution under equivalent conditions, GDOPT shows striking advantages over direct collocation in OpenModelica in both speed and solution quality.

6 Limitations

While effective, L2BN and GDOPT have several limitations that should be considered. First, L2BN does not take advantage of the spectral convergence properties of flipped Legendre-Gauss-Radau (fLGR) collocation points. Therefore, it is unable to achieve the convergence rates of hp-adaptive methods and may require excessive mesh iterations or intervals to reach high accuracy, potentially leading to numerical instabilities.

The performance of L2BN also depends on the initial mesh size and refinement parameters, such as the number of phase I and phase II iterations. Clearly, the initial mesh must resemble the principle behavior of the subsequent iterations, otherwise the phase I iterations can have negative effects. Poor choices for these settings can result in unnecessary computational effort or failure to resolve key features of the solution.

Another limitation is the absence of direct error estimation in the algorithm. Since L2BN does not explicitly compute or use error estimates, it may not always refine the mesh optimally. For example, regions with smooth but rapidly changing controls might be over-refined, because of the large change on the interval. This can lead to a lot of unnecessary computations and considerably larger meshes than sufficient.

The most significant limitation, however, is that GDOPT is unable to handle DAEs directly and must model algebraic variables as control inputs. An integration into an elaborate modeling environment like OpenModelica would allow solving for algebraic variables at every step, thus reducing the solver space. Furthermore, direct error estimates could be incorporated by leveraging established simulation capabilities of OpenModelica.

7 Discussion and Future Work

This work presents two significant contributions to direct collocation-based dynamic optimization: (1) a comprehensive re-implementation that addresses significant limitations in optimizing with OpenModelica, and (2) the novel L2BN mesh refinement algorithm. The results demonstrate that GDOPT achieves 2 to 3 times speedups over OpenModelica under equivalent conditions, while introducing additional, essential capabilities such as adaptive mesh refinement, parameter optimization, exact Hessians, and support for higher-order collocation schemes. Tested on a representative optimal control problem, the proposed mesh refinement algorithm proves to be highly efficient. It speeds up convergence from poor initial guesses, correctly detects non-smooth regions, and enhances solution accuracy with low computational effort. Additionally, it reaches a resolution at switching points that would be unrealistic to achieve with equidistant meshes in a reasonable amount of time.

Ongoing work focuses on integrating *libgdopt* into the OpenModelica toolchain to replace its legacy optimization runtime. This integration will bring several substantial improvements, including full parameter optimization capabilities, adaptive mesh refinement as presented in this work, variable polynomial degrees, and exact Hessian computation for improved convergence. Moreover, native DAE support, already available in OpenModelica, will be preserved through algebraic system resolution at each optimization step, expanding the capabilities of GDOPT, which is currently limited to ODEs. The performance gains demonstrated in this paper suggest that such an integration could significantly enhance the optimization capabilities of OpenModelica.

Acknowledgements

This work was conducted as part of the OpenSCALING project (Grant No. 01IS23062E) at the University of Applied Sciences and Arts Bielefeld, in collaboration with Linköping University. The authors would like to express their sincere appreciation to both the OpenSCALING project and the Open Source Modelica Consortium (OSMC) for their support, collaboration, and shared commitment to advancing open-source modeling and simulation technologies.

References

Amestoy, Patrick R. et al. (2001). “A Fully Asynchronous Multifrontal Solver Using Distributed Dynamic Scheduling”. In: *SIAM Journal on Matrix Analysis and Applications* 23.1, pp. 15–41.

Bachmann, Bernhard et al. (2012-09). “Parallel Multiple-Shooting and Collocation Optimization with OpenModelica”. In: DOI: 10.3384/ecp12076659.

Becerra, V. M. (2010). “Solving complex optimal control problems at no cost with PSOPT”. In: *2010 IEEE International Symposium on Computer-Aided Control System Design*, pp. 1391–1396. DOI: 10.1109/CACSD.2010.5612676.

Betts, John T. and William P. Huffman (1998). “Mesh refinement in direct transcription methods for optimal control”. In: *Optimal Control Applications and Methods* 19.1, pp. 1–21.

Fritzson, Peter, Adrian Pop, Karim Abdelhak, et al. (2020-10). “The OpenModelica Integrated Environment for Modeling, Simulation, and Model-Based Development”. In: *Modeling, Identification and Control: A Norwegian Research Bulletin* 41, pp. 241–295. DOI: 10.4173/mic.2020.4.1.

Garg, Divya, Michael Patterson, Camila Francolin, et al. (2011-06). “Direct Trajectory Optimization and Costate Estimation of General Optimal Control Problems Using a Radau Pseudospectral Method”. In: *Computational Optimization and Applications* 49, pp. 335–358. DOI: 10.1007/s10589-009-9291-0.

Gill, P. E. et al. (2007). *SNOPT 7.7 User’s Manual*. Tech. rep. CCoM Technical Report 18-1. San Diego, CA: Center for Computational Mathematics, University of California, San Diego.

Gill, Philip E., Walter Murray, and Michael A. Saunders (2005-01). “SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization”. In: *SIAM Rev.* 47.1, pp. 99–131. ISSN: 0036-1445. DOI: 10.1137/S0036144504446096. URL: https://doi.org/10.1137/S0036144504446096.

HSL (2013). *HSL: A collection of Fortran codes for large-scale scientific computation*. Available at http://www.hsl.rl.ac.uk. Accessed: 15-04-2025.

Hunter, J. D. (2007). "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3, pp. 90–95. DOI: 10.1109/MCSE.2007.55.

Jain, Sachin and Panagiotis Tsiotras (2008-09). "Trajectory Optimization Using Multiresolution Techniques". In: *Journal of Guidance Control and Dynamics - J GUID CONTROL DYNAM* 31. DOI: 10.2514/1.32220.

Langenkamp, Linus (2024-12). "Adaptively Refined Mesh for Collocation-Based Dynamic Optimization". MA thesis. DOI: 10.13140/RG.2.2.18499.72484.

Liu, Fengjin, William W. Hager, and Anil V. Rao (2015). "Adaptive mesh refinement method for optimal control using nonsmoothness detection and mesh size reduction". In: *Journal of the Franklin Institute* 352.10, pp. 4081–4106. ISSN: 0016-0032. DOI: https://doi.org/10.1016/j.jfranklin.2015.05.028. URL: https://www.sciencedirect.com/science/article/pii/S0016003215002045.

Magnusson, Fredrik and Johan Åkesson (2015). "Dynamic Optimization in JModelica.org". In: *Processes* 3.2, pp. 471–496. ISSN: 2227-9717. DOI: 10.3390/pr3020471. URL: https://www.mdpi.com/2227-9717/3/2/471.

Mengist, Alachew, Vitalij Ruge, Mahder Gebremedhin, et al. (2013-09). "Model-Based Dynamic Optimization with OpenModelica and CasADi". In.

Patterson, Michael A., William W. Hager, and Anil V. Rao (2015). "A ph mesh refinement method for optimal control". In: *Optimal Control Applications and Methods* 36.4, pp. 398–421. DOI: https://doi.org/10.1002/oca.2114. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/oca.2114. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/oca.2114.

Patterson, Michael A. and Anil V. Rao (2014-10). "GPOPS-II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using hp-Adaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming". In: *ACM Trans. Math. Softw.* 41.1. ISSN: 0098-3500. DOI: 10.1145/2558904. URL: https://doi.org/10.1145/2558904.

Ruge, Vitalij et al. (2014-03). "Efficient Implementation of Collocation Methods for Optimization using OpenModelica and ADOL-C". In: pp. 1017–1025. ISBN: 978-91-7519-380-9. DOI: 10.3384/ecp140961017.

Sagliano, Marco et al. (2018-01). "SPARTAN: A Novel Pseudospectral Algorithm for Entry, Descent, and Landing Analysis". In: pp. 669–688. ISBN: 978-3-319-65282-5. DOI: 10.1007/978-3-319-65283-2_36.

Schneider, Claus and Wilhelm Werner (1986-07). "Some new aspects of rational interpolation". In: *Math. Comput.* 47.175, pp. 285–299. ISSN: 0025-5718. DOI: 10.2307/2008095. URL: https://doi.org/10.2307/2008095.

Sivertsson, Martin and Lars Eriksson (2012). "Time and Fuel Optimal Power Response of a Diesel-Electric Powertrain". In: *IFAC Proceedings Volumes* 45.30. 3rd IFAC Workshop on Engine and Powertrain Control, Simulation and Modeling, pp. 262–269. ISSN: 1474-6670. DOI: https://doi.org/10.3182/20121023-3-FR-4025.00036. URL: https://www.sciencedirect.com/science/article/pii/S1474667015351673.

Wächter, Andreas and Lorenz T. Biegler (2006-03). "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". In: *Mathematical Programming* 106.1, pp. 25–57. ISSN: 1436-4646. DOI: 10.1007/s10107-004-0559-y. URL: https://doi.org/10.1007/s10107-004-0559-y.

Wen, C.S. and T.F. Yen (1977). "Optimization of oil shale pyrolysis". In: *Chemical Engineering Science* 32.3, pp. 346–349. ISSN: 0009-2509. DOI: https://doi.org/10.1016/0009-2509(77)80221-2. URL: https://www.sciencedirect.com/science/article/pii/0009250977802212.

Zhao, Jisong and Teng Shang (2018). "Dynamic Optimization Using Local Collocation Methods and Improved Multiresolution Technique". In: *Applied Sciences* 8.9. ISSN: 2076-3417. DOI: 10.3390/app8091680. URL: https://www.mdpi.com/2076-3417/8/9/1680.

Zhao, Yiming and Panagiotis Tsiotras (2009-04). "Mesh Refinement Using Density Function for Solving Optimal Control Problems". In: *AIAA Infotech at Aerospace Conference and Exhibit and AIAA Unmanned...Unlimited Conference*. DOI: 10.2514/6.2009-2019.

Appendix: Proof of Lemma 1

Lemma 1. *Let $p \in P^m$ be a polynomial and $p_\varepsilon(t) = p\left(\frac{t}{2}\right) + \varepsilon(t)$ be a variation of $p\left(\frac{t}{2}\right)$ by $\varepsilon \in P^m$, then for a given $\delta \in \mathbb{R}$ with $\frac{\sqrt{2}}{2} < \delta < 1$*

$$\frac{\|\dot{p}_\varepsilon(t)\|}{\|\dot{p}(t)\|} < \delta, \quad \text{if} \quad \|\dot{\varepsilon}(t)\| < \frac{\sqrt{2}\delta - 1}{2}\left\|\dot{p}\left(\frac{t}{2}\right)\right\|.$$

Proof.

$$\begin{aligned} \|\dot{p}(t)\|^2 &= \int_0^1 \dot{p}(t)^2 \,\mathrm{d}t = \int_0^{\frac{1}{2}} \dot{p}(t)^2 \,\mathrm{d}t + \int_{\frac{1}{2}}^1 \dot{p}(t)^2 \,\mathrm{d}t = \frac{1}{2}\left(\int_0^1 \dot{p}\left(\frac{t}{2}\right)^2 \mathrm{d}t + \int_0^1 \dot{p}\left(\frac{t+1}{2}\right)^2 \mathrm{d}t\right) \\ \implies \|\dot{p}(t)\| &= \frac{1}{\sqrt{2}}\left(\int_0^1 \dot{p}\left(\frac{t}{2}\right)^2 \mathrm{d}t + \int_0^1 \dot{p}\left(\frac{t+1}{2}\right)^2 \mathrm{d}t\right)^{\frac{1}{2}} = \frac{1}{\sqrt{2}}\left(\left\|\dot{p}\left(\frac{t}{2}\right)\right\|^2 + \left\|\dot{p}\left(\frac{t+1}{2}\right)\right\|^2\right)^{\frac{1}{2}} \end{aligned} \tag{15}$$

Also $\dot{p}_\varepsilon(t) = \frac{1}{2}\dot{p}\left(\frac{t}{2}\right) + \dot{\varepsilon}(t)$ and by the *Minkowski inequality* $\|\dot{p}_\varepsilon(t)\| \leq \frac{1}{2}\left\|\dot{p}\left(\frac{t}{2}\right)\right\| + \|\dot{\varepsilon}(t)\|$, implying

$$\frac{\|\dot{p}_\varepsilon(t)\|}{\|\dot{p}(t)\|} \leq \frac{\frac{1}{2}\left\|\dot{p}\left(\frac{t}{2}\right)\right\| + \|\dot{\varepsilon}(t)\|}{\frac{1}{\sqrt{2}}\left(\left\|\dot{p}\left(\frac{t}{2}\right)\right\|^2 + \left\|\dot{p}\left(\frac{t+1}{2}\right)\right\|^2\right)^{\frac{1}{2}}} \leq \frac{1}{\sqrt{2}}\frac{\left\|\dot{p}\left(\frac{t}{2}\right)\right\| + 2\|\dot{\varepsilon}(t)\|}{\left\|\dot{p}\left(\frac{t}{2}\right)\right\|} = \frac{1}{\sqrt{2}}\left(1 + \frac{2\|\dot{\varepsilon}(t)\|}{\left\|\dot{p}\left(\frac{t}{2}\right)\right\|}\right). \tag{16}$$

By setting $\frac{1}{\sqrt{2}}\left(1 + \frac{2\|\dot{\varepsilon}(t)\|}{\|\dot{p}\left(\frac{t}{2}\right)\|}\right) < \delta$ for $\delta \in \mathbb{R}$ with $\frac{\sqrt{2}}{2} < \delta < 1$ and rearranging terms, the proposition follows. □

DOI 10.3384/ecp218127

Validation of Dynamic Simulation Models Using Metamorphic Testing and Given-When-Then Patterns

Gaadha Sudheerbabu[1] Dragos Truscan[1] Mikael Manngård[2] Kristian Klemets[3]

[1]Åbo Akademi University, Finland, firstname.lastname@abo.fi
[2]Novia University of Applied Sciences, Finland mikael.manngard@novia.fi
[3]Univeristy of Turku, Finland kristian.klemets@utu.fi

Abstract

As the maritime industry evolves, there is a focus on simulation-driven design, testing, and validation using novel technology solutions. Simulation models designed to represent the behaviour and features of real systems are increasingly available for testing during the early phase of the full development, but in many cases, their testing suffers from the availability of test oracles. Metamorphic testing has become increasingly used in different application domains as an approach to test systems when an explicit test oracle is unavailable. In order to increase its adoption by domain experts, we combine metamorphic testing with Behaviour Driven development for the verification and validation of simulation models. The tool-based approach facilitates automated test generation based on domain-specific custom metamorphic transformations to generate meaningful test inputs for metamorphic input relations. The method also uses features and scenarios extracted from system requirements and domain expertise to define metamorphic output relations. By automating test generation based on system behaviours as features, scenarios, metamorphic transformations, and output candidates for metamorphic relations in a Gherkin-like format, the tool enables practitioners to verify models based on domain-specific constraints and metamorphic relation checks. Our preliminary evaluation shows that the tool can detect metamorphic relations violations in the simulation models under test and that automated test generation provides improved coverage.

Keywords: Metamorphic testing, Validation, Simulation, FMI, FMU

1 Introduction

In recent times, maritime vessels have evolved into complex systems of systems with advanced automation and control systems. This evolution has made the verification and validation of maritime systems more challenging and has increased the demand for new methods for early simulation-based testing. The simulation model of a system is designed based on the properties and behaviour of the real system it represents. The *Functional Mock-Up Interface* (FMI) (Blochwitz et al. 2012) is a standard that defines a container and an interface to exchange dynamic simulation models. The container packs simulation models into an archive file called *functional mock-up unit* (FMU), which contains metadata, interface specification, binary files, data files, documentation and, optionally, the source code of the model. The goal of packing simulation models as FMUs is to exchange them between different stakeholders, such as suppliers and manufacturers, and this opens up the possibility of providing black-box models of the systems and subsystems for acceptance and integration testing. Besides the fact that, in many cases, dynamic simulation models do not come accompanied by explicit test oracles to allow automated testing, this encapsulation complicates the acceptance and integration testing processes.

Metamorphic testing (Tsong Y Chen et al. 2020) is an increasingly popular testing technique for generating test cases and verifying test results based on the properties or functional behaviour of the system. This technique also effectively addresses one of the fundamental challenges in software testing, the test oracle problem (Barr et al. 2014). The test oracle problem is the challenge of determining the expected output or behaviour of the *system under test* (SUT) for a given input. Often, when an explicit test oracle is unavailable to validate the correctness of the execution results of simulation models, they are considered non-testable (Balci 2003; Weyuker 1982).

A recent study about the research directions in metamorphic relation generation (Li et al. 2024) discusses the possibility of extending the applicability of MT in automated test generation to wider application domains that face the oracle problem. Metamorphic testing has been applied to the verification of simulation software (Murphy et al. 2011) and to the verification (Jiang et al. 2014; Lindvall et al. 2017) and validation (M. M. Olsen and M. S. Raunak 2016; M. Olsen and M. Raunak 2018) of simulation models. Recent research on the validation of simulation models using metamorphic testing shows that it has been applied to agent-based and discrete event simulation models, as well as to hybrid simulation models that include combinations of agent-based models, discrete event simulation models, and system dynamics models.

Behaviour-Driven Development (BDD) is a software development approach that extends Test-Driven Development (TDD) by focusing on the behaviour of an applica-

tion from the end user's perspective. The key idea is to write tests in a natural, readable language that describes how the application should behave under specific scenarios. BDD helps teams create a shared understanding of requirements and ensures software is aligned with business goals. BDD uses requirements patterns, such as Given-When-Then, to describe the expected behaviour of a system in a clear, consistent way. It breaks down each scenario into three logical parts: preconditions, action, and expected outcome, allowing domain experts to specify executable testing scenarios.

In this work, we propose the integration of BDD with metamorphic testing to facilitate specifying the test design by domain experts to automate the test case generation, execution, and verdict assignment for the validation of dynamic simulation models. The approach can streamline the testing process by generating the test design documentation in a practitioner-friendly BDD format and automating the different steps of the testing cycle with the following contributions:

- We propose a set of metamorphic input and output relational operators to test system dynamics simulation models.
- We provide a framework to allow domain experts to express test actions and expected behaviour in BDD format by selecting metamorphic transformations and output relations.
- We provide the tool support for automating different steps of the metamorphic testing process: the metamorphic BDD test case generation, input transformation, test execution and test verdict assignment.

The rest of the paper is divided into six sections. Section 2 describes concepts such as metamorphic testing and behaviour-driven development testing. Section 3 introduces the overview of our approach. We exemplify our approach on the simulation model of a lubricating oil cooling system in Section 4 and discuss the results in Section 5. We overview related work in Section 6, and draw the conclusions in Section 7.

2 Prerequisites

This section describes in more detail the metamorphic testing technique and behaviour-driven development testing.

2.1 Metamorphic testing

Metamorphic testing (MT) was introduced by Chen et al. (Tsong Y Chen et al. 2020) as a solution to test systems without explicit specification of the test oracle. In MT, the behavioural or functional properties of the system are defined by posing a hypothesis about using generic relations, known as *metamorphic relations (MRs)*, between different sets of inputs and their expected outputs.

The typical MT process is shown in Figure 1. A *source test case* is the first set of tests performed using *seed inputs (initial inputs)* denoted as x in Figure 1. The *seed inputs* are then transformed into *morphed inputs* denoted as x' based on a defined *metamorphic transformation*. The *follow-up test cases* are performed using the resulting *morphed inputs*.

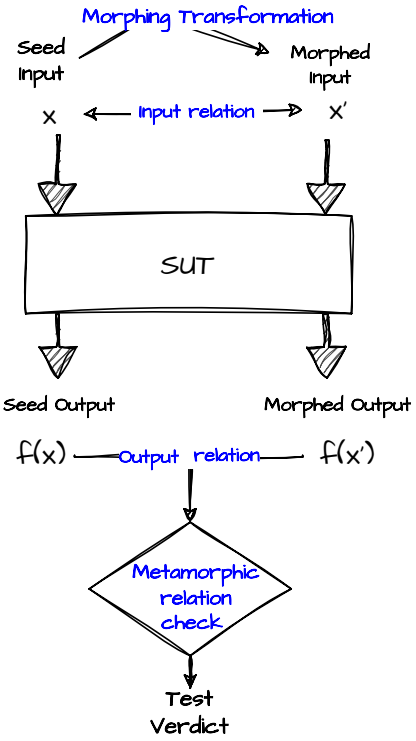


Figure 1. Generic metamorphic testing process

An MR is composed of two parts: an *input relation* and *output relation* (Liu et al. 2012). The *input relation* represents the relation between the inputs of the source test case and a follow-up test case, whereas an *output relation* represents the relation between the expected outputs of the *source and follow-up test cases*. In addition, an *implication* between the outputs of *source and follow-up test cases* is needed to specify the impact of input transformations on their corresponding outputs. The output relation, often specified using a relational operator such as equality, subset, disjoint, complete, and difference (Segura et al. 2017), should hold for any corresponding output of the system. The test verdict is assigned based on the *violation* or *non-violation of output relation*.

Several studies show that metamorphic testing has become a popular choice for testing systems without explicit test oracles and that it has been applied successfully to many application domains, including embedded systems, web applications, computer graphics, and simulation and

DOI
10.3384/ecp218139

modelling (Segura et al. 2016; Tsong Yueh Chen et al. 2018).

2.2 Behaviour Driven Development testing

Behaviour-driven development (Smart and Molak 2023) testing aims to build a development process in which development and testing activities occur concurrently throughout the product life cycle, facilitating faster and more reliable releases. The Gherkin language (Wynne and Hellesoy 2012) serves a key role in BDD and test automation by allowing developers and other stakeholders to describe software behaviour clearly that is both human-readable and executable by machines. As a structured plain-text language, Gherkin uses keywords such as *Feature, Scenario, Given, When, Then* to define the abstract description of the feature or behaviour of the system, scenarios associated with it and the steps that describe the initial conditions, test actions, and acceptance criteria. These steps ensure precise documentation of system requirements and test design as *user scenarios*. An example of a scenario specified using *Given-When-Then (GWT)* patterns is presented in Listing 1.

Listing 1. Example test specified using GWT template

```
Feature: User login
  Scenario: Successful login with valid
      credentials
    Given the user is on the login page
    When the user enters valid credentials
    Then the user should be redirected to
          the dashboard
```

3 Overview of the approach

We propose a metamorphic behaviour-driven development (Metamorphic BDD) approach for testing simulation models. The approach is intended to allow domain experts to formulate easily metamorphic tests for FMUs, by mapping concepts of MT to the GWT templates (as shown in Table 1).

Table 1. Mapping MT concepts to GWT patterns

Metamorphic concept	GWT pattern
Seed input & output	Given
Morphing Transformation Follow-up input	When
Output Metamorphic Relation	Then

Let x be an input to the SUT, and $x' = \mathrm{MT}(x)$ its metamorphic transform. The output of the SUT is $f(x)$. The metamorphic relation is evaluated by a function $\mathrm{MR}_{\mathrm{out}}(f(x), f(x'))$, which checks whether the expected relationship between the outputs holds. Consider the following GWT scenario:

$$\begin{aligned} &GIVEN \quad x \quad with \quad initial \quad value \quad x_0 \\ &WHEN \quad x' = \mathrm{MT}(x) \\ &THEN \quad \mathrm{MR}_{\mathrm{out}}(f(x), f(x')), \end{aligned}$$

In this case, all the inputs and outputs are time series over a specified time interval. As such, a metamorphic test consists of four steps:

1. *Execute source test*, with seed input x;
2. *Apply metamorphic transformation* to seed input $x' = \mathrm{MT}(x)$;
3. *Execute follow-up test* with morphed input x';
4. *Verify the metamorphic relation* between the seed output $f(x)$ and morphed output $f(x')$.

Although the approach can be applied for defining tests for any kind of simulation system, in this paper, we take advantage of the information provided in the FMU package to streamline and automate the tool support. To that extent, we use the interface specification of the SUT extracted from the FMU package and the necessary properties of the system from the requirements specification as input. This will allow us to facilitate MR definitions and to automate test generation, execution, and verdict assignment. Finally, a test report is generated as output.

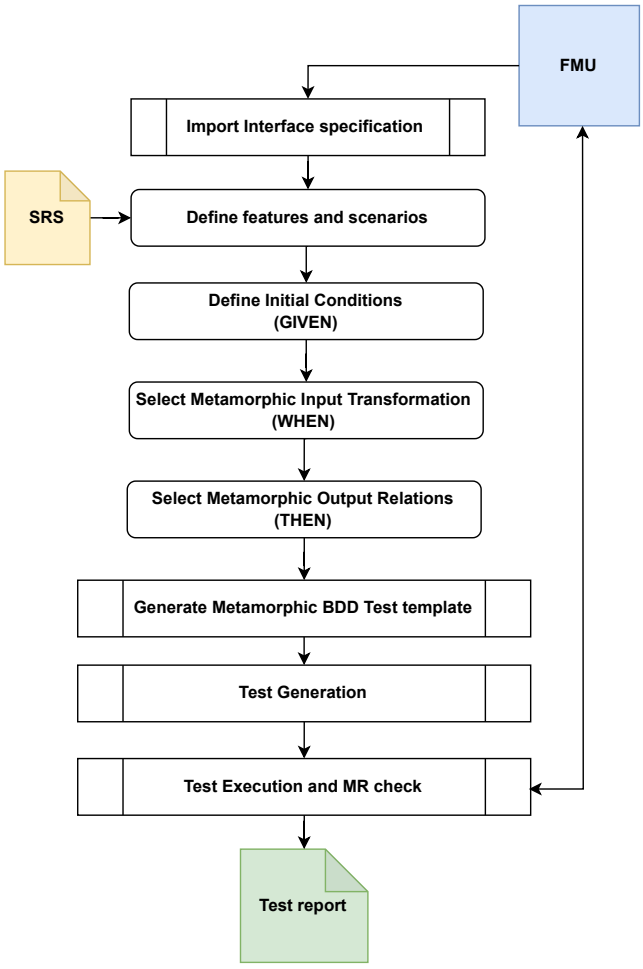


Figure 2. Metamorphic BDD testing workflow

The workflow of the approach is depicted in Figure 2. In the following, we will discuss the main steps of the approach.

Import interface specification. In the first step of our approach, the interface specification of the SUT is imported from the FMU file (*modelDescription.xml*). This includes the input and output signals, the initial values for the inputs and the ranges of the input.

Define features and scenarios. Then, the domain expert defines the features and scenarios for validating the relevant properties of the SUT.

Define initial conditions The user defines the initial condition values for the inputs of the system. These conditions correspond to the GIVEN statement of the GWT pattern.

Select Metamorphic Input Transformation. For the WHEN statement, the user selects the metamorphic transformation that should be applied to each input signal in the scenario defined. A given *metamorphic transformation* is defined as a tuple $(transformation, relation)$. The *transformation* defines the pattern based on which the input signal is changed (e.g., *None*, *Steps*, *Drift*), whereas the *relation* specifies the direction of the change (e.g., *constant*, *increase*, *decrease*). In case of *Drift* transformation, the input signals vary by a *ramp_rate* value selected from the range ± 2 and for *Steps*, they vary from the initial condition value by $\pm(value)$ until it reaches the lower and upper limits of the inputs they are applied upon. The actual value of the *ramp_rate* can be defined manually or generated using a random uniform sampling method, which is used to create the morphed input values for the corresponding input variable.

Metamorphic Output Relation Selection. The output metamorphic relation can be defined in the THEN statement, for one or several of the output signals extracted from the interface specification of the FMU. The output relations that are applicable to validate the expected properties of the output signal can be selected from the list of temporal and relational operator combinations enumerated below:

1. *Always_Greater than*: Implies that morphed outputs of the selected output variable should always be greater than the corresponding seed outputs.
2. *Always_Greater than or equal to*: Implies that morphed outputs of the selected output variable should always be greater than or equal to the corresponding seed outputs.
3. *Always_Less than*: Implies that morphed outputs of the selected output variable should always be less than the corresponding seed outputs.
4. *Always_Less than or equal to*: Implies that morphed outputs of the selected output variable should always be less than or equal to the corresponding seed outputs.
5. *Always_Equal to*: Implies that morphed outputs of the selected output variable should always be equal to the corresponding seed outputs.
6. *Eventually_Increases than*: Implies that morphed outputs of the selected output variable should eventually increase than the corresponding seed outputs.
7. *Eventually_Decreases than*: Implies that morphed outputs of the selected output variable should eventually decrease than the corresponding seed outputs.
8. *Always in given range [lower_limit upper_limit]*: Implies that morphed outputs of the selected output variable should always stay in the range defined as its lower limit and upper limit in the requirements specification.

All these metamorphic relations assume that the seed and morphed outputs will be checked at each time step of the simulation.

Metamorphic BDD test template generation. In this step, a feature file with the metamorphic BDD format is generated based on the metamorphic relations selected for the inputs and outputs in the previous step. This step takes as input the number of scenario instances to be generated and the number of time steps per scenario instance to create the parametrized BDD test file in Gherkin format.

Test Generation. In this step, the morphed test input values are generated for the follow-up tests based on the transformation relation selected and the parametrized values in the BDD test file. These test inputs are also saved to a test suite file in '.json' format.

The test inputs are generated using the *ramp_rate* added as a parametrized value in the BDD test file. The seed input is transformed into morphed input by applying the selected transformation on the initial condition value added in the GIVEN condition for the corresponding input. If the transformation selected is *None*, the value remains the same as the initial condition value throughout the simulation.

Test Execution and MR checking. In this last step, the morphed test inputs are executed on the FMU file. For this case study, the Python package `PyFMI` (Andersson, Åkesson, and Führer 2016) was used to simulate FMUs. The simulations run for the source test execution using the initial condition values for inputs and the follow-up scenario instances using the morphed input values generated. Then, each set of morphed outputs from follow-up test executions is compared with seed outputs from the source test execution, using the output MRs to assign the test verdict. The test verdict of each follow-up test execution and the comparison plots are summarized and presented in a test report for further analysis.

One should notice that each generated scenario instance would require executing a seed test and a follow-up test. However, since the seed test would be identical for each scenario instance, only one execution of the seed test is needed in order to collect the corresponding seed output. That is, if from a given scenario, *n* scenario instances are generated, the number of executions of the SUT equals *n+1*. This considerably reduces the test execution time.

DOI 10.3384/ecp218139

4 Case Study

We will exemplify the approach presented in Section 3 on a simplified version of a *Lubricating Oil Cooling* (LOC) system that transfers heat from the lubrication oil to the cooling water circuit of a ship engine unit (NoviaRDISeafaring 2024). The LOC system has a control valve which controls the lubrication oil temperature at a constant setpoint at the engine inlet using a PI controller. The controller aims to keep the lubrication oil temperature at the outlet within the boundary values specified at all conditions.

To test the LOC system using our approach, we start by importing the interface specification of the system. Then, the relevant features to be tested should be defined in the framework. For instance, a feature and scenario added to test the LOC system can be seen in Figure 3.

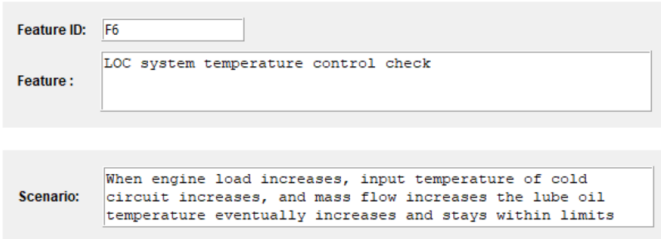


Figure 3. Defining the Feature and Scenario to test LOC system

The tool automatically extracts inputs and outputs from the model description and presents them in the GUI together with predefined data values. The initial condition values for the inputs of the system can be provided in the GIVEN tab, as in Figure 4.

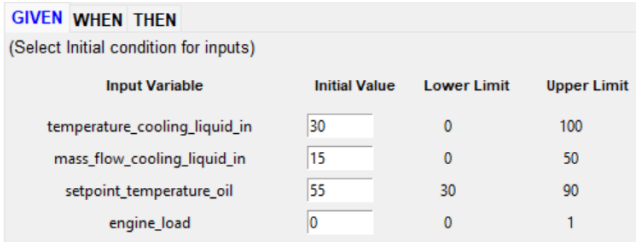


Figure 4. Setting Initial condition values

For the WHEN statement, the user selects the metamorphic transformation that should be applied for all inputs of the FMU (see Figure 5).

Figure 6 shows an example resulting input signal generated upon selecting *Steps_Increase* using a *ramp_rate* of `3.64` at a time interval of `5` for the input *temperature_cooling_liquid_in* with an initial condition value `30`. The transformation will preserve the original limits of the signal, `0` and `100`, respectively.

Figure 7 shows the output signals of the LOC system and the set of predefined output relations that can be selected for each signal. From the listed set of relations, the

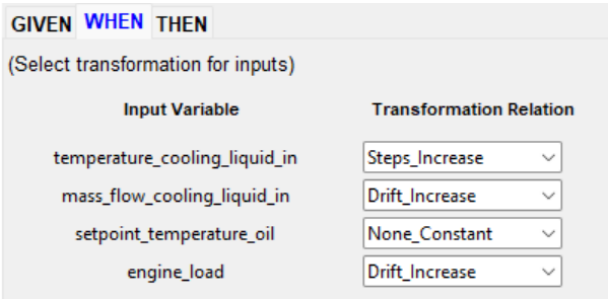


Figure 5. Metamorphic Input Transformation Selection options

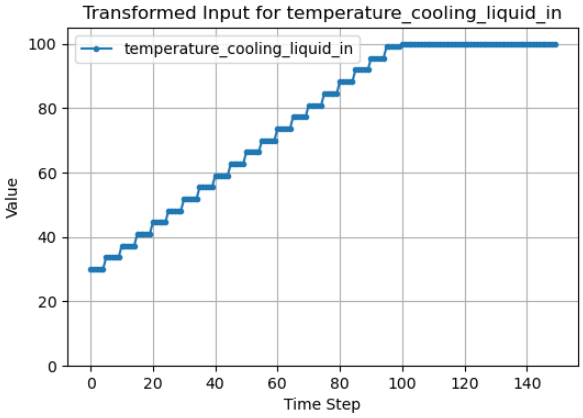


Figure 6. Example of transformed input using *Steps_Increase* transformation.

ones corresponding to the output variable selected and the input transformation applied can be chosen to facilitate the automatic generation of the metamorphic BDD test file.

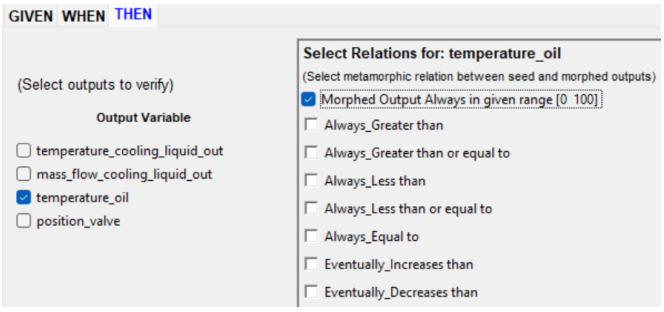


Figure 7. Metamorphic Output Relation Selection options

Figure 8 shows the first few scenario instances of the parametrized BDD test file created based on the inputs and outputs selected in Section 4, and the test generation specific parameters such as number of scenario instances to be generated as `5`, and the number of time steps per scenario instance as `1000`.

For the BDD test scenario in Figure 8, the follow-up test inputs generated for one of the scenario instances are depicted in Figure 9. Figure 10, shows the seed and mor-

```
Feature: F6_LOC system temperature control check
  Scenario: F6_scenario When engine load increases, input temperature of cold
circuit increases, and mass flow increases the lube oil temperature eventually
increases
    Given 'F6_scenario' has seed_inputs with initial values '[30, 15, 55, 0]'
generating seed_outputs in '1000' time steps

    When 'engine_load' transform by 'Drift_Increase' with ramp-rate '0.0027'
     And 'mass_flow_cooling_liquid_in' transform by 'Drift_Increase' with
ramp-rate '0.0027'
     And 'temperature_cooling_liquid_in' transform by 'Steps_Increase' with
ramp-rate '3.64'
    Then morphed_outputs of 'temperature_oil' 'Always in given range [0  100]'
     And morphed_outputs of 'position_valve' 'Always in given range [0  1]'
```

Figure 8. Metamorphic BDD test specification in GWT format

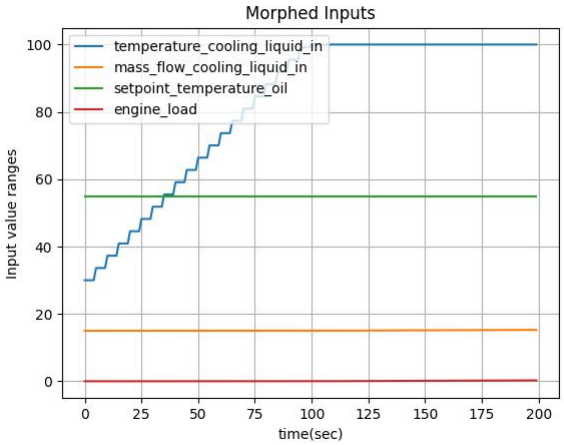


Figure 9. Plot of follow-up test generated for a scenario instance in Figure 8

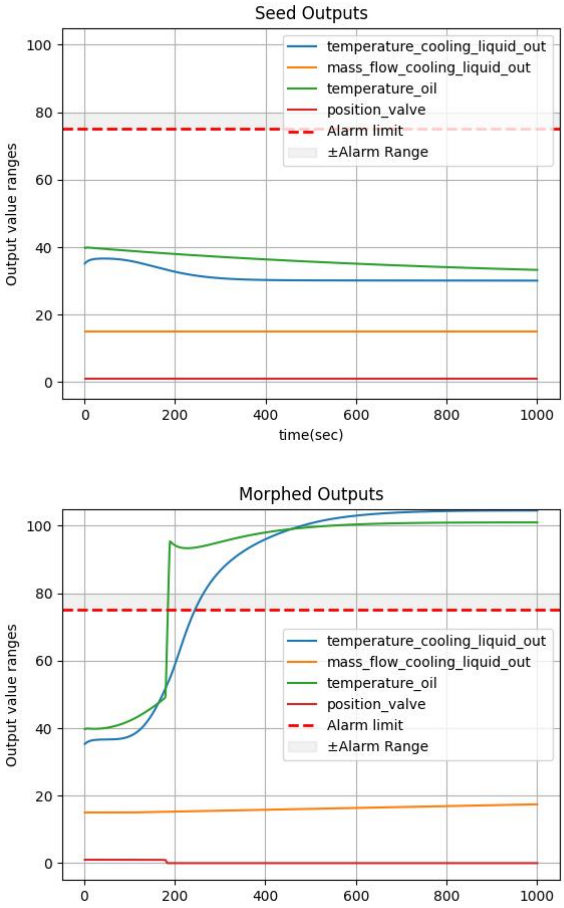


Figure 10. Example of seed and morphed output plots included in the test report

phed outputs of the scenario presented in Figure 8 where the expected behaviour of the system is that the lubricating oil temperature at the outlet eventually increases compared to the seed outputs and stays within the boundary limits in response to the input transformations depicted in Figure 9. The verdict is assigned by checking the values of *temperature_oil* from the simulation results of the source and follow-up test execution of the system using the metamorphic output relation.

4.1 Tool support

Tool support has been in Python using the GUI toolkit `Tkinter` (Lundh 1999) for the user interface (see Figure 11). Most of the steps in the metamorphic testing process are fully automated based on the input and output MR selection using the framework. To implement automation, we have used Python, Behave library, a BDD framework for Python (Behave 2012), and `PyFMI`, which interacts with the FMU generated using `Simulink` (MATLAB/Simulink 2024) to perform automated test execution. Allure (Allure 2025), an open-source tool for visualising the results of a test run, is used to generate the test report (see Figure 12), which is integrated with Behave and can also be integrated with task management tools like Jira.

5 Results and discussion

We have defined *6* features and their corresponding scenarios, and provided *1000* time steps to generate the test inputs for each execution of the SUT. The number of steps executed per feature and scenario depends on the number of scenario instances and the number of Given-When-Then templates in the BDD test file. The execution time of the tests was *1m14.584s*. The test results, comprising the features and behaviours tested, and the test verdict, are summarised in a test report in HTML format. The test suite with test inputs and outputs is saved in *'.json'* format. A domain expert performs further analysis to fix the defects identified by the failed tests, based on the details in the test report and test inputs that caused the system to fail. The failed scenario instance of *Feature 6* highlighted in Figure 12 checks if the output relation, *Always in given range [lower_limit upper_limit]*, is satisfied for the *temperature_oil*. Based on the evaluation of observed outputs, the MR is not satisfied, and the input transformations and the ramp_rates used in the scenario instance can be further analysed to fine-tune the model parameters to fix the issue.

DOI 10.3384/ecp218139

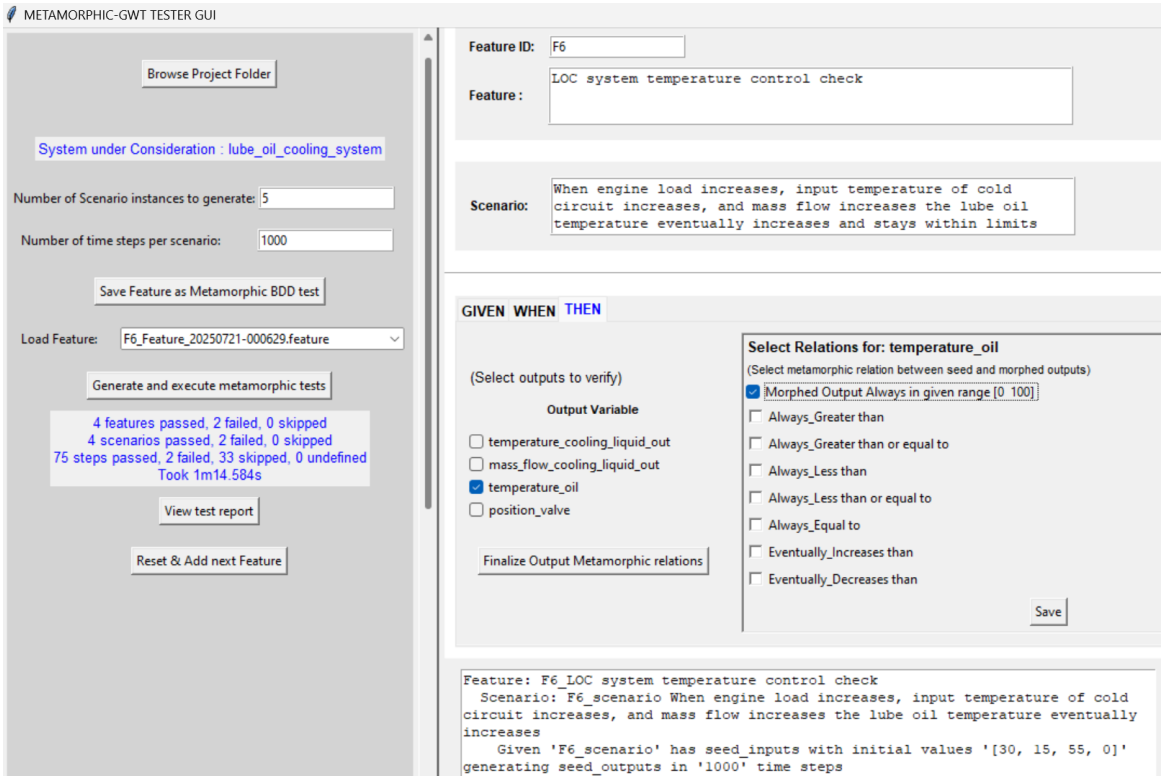


Figure 11. GUI of the Metamorphic GWT tool

6 Related work

Yao, Deng, et al. (Deng et al. 2021) proposed a declarative metamorphic testing approach, BMT, for testing autonomous driving models. The approach designed for generating MRs using custom traffic behaviours is primarily applied to image processing to verify deep learning models on their erroneous predictions. Their work also used behaviour-driven development based domain-specific language to build the metamorphic relation and to generate a test template similar to our approach. However, they target image processing systems for autonomous driving models, whereas our work focuses on testing the dynamic simulation models exported as FMUs in the maritime application domain.

7 Conclusions

We proposed a metamorphic testing approach combined with behaviour-driven development for the validation of simulation models. The main contributions of our approach were: specifying the metamorphic tests as scenarios using GWT patterns, and defining a set of metamorphic transformations and metamorphic relations to validate dynamic simulation systems. Tool support for the approach has been provided in the form of a GUI-based tool that allows the domain expert to import FMU specifications and to formulate metamorphic tests as parametrizable scenarios. The tool also allows the generation of additional tests from these scenarios. Currently, the input-output combinations in metamorphic relations are selected based on domain expertise to facilitate the test generation process. However, this can be done semi-automatically based on the availability of artefacts such as SysML diagrams, causal graphs or cause-effect diagrams (Clark et al. 2023).

Future work will focus on customizing the framework to utilize information mined from causal modelling approaches for the systematic generation of metamorphic relations. At the moment, the test generation uses random generation, but future work will replace it with a guided approach, based on our previous work on combining metamorphic testing and generative adversarial networks (Sudheerbabu et al. 2024). Future work also includes a study with practitioners on the usability and practical effectiveness of the tool to both researchers and practitioners in the context of acceptance and integration testing.

Acknowledgment

This work has been supported by Business Finland via the Virtual Sea Trial project (VST), under grant 7187/31/2023.

References

Allure (2025). *Allure report*. https://allurereport.org/docs/. Accessed: 2025-03-16.

Andersson, Christian, Johan Åkesson, and Claus Führer (2016). "Pyfmi: A python package for simulation of coupled dynamic

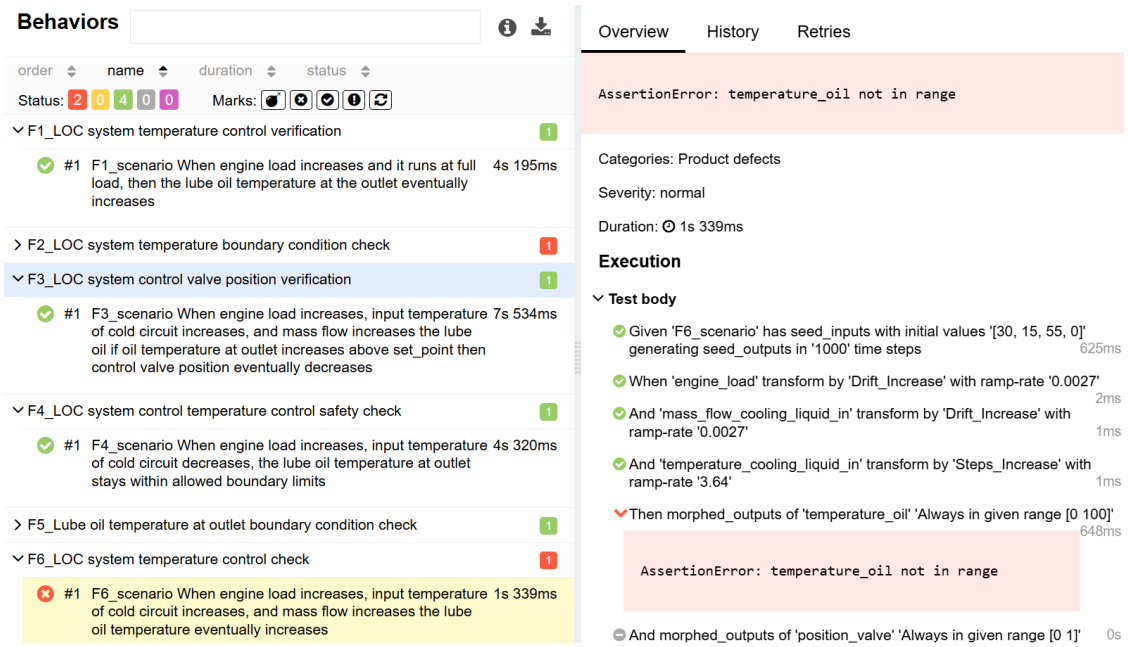


Figure 12. Test report with summary of test verdicts

models with the functional mock-up interface". In: *Technical Report in Mathematical Sciences* 2.

Balci (2003). "Verification, validation, and certification of modeling and simulation applications". In: *Proceedings of the 2003 Winter Simulation Conference, 2003.* Vol. 1. IEEE, pp. 150–158.

Barr, Earl T et al. (2014). "The oracle problem in software testing: A survey". In: *IEEE transactions on software engineering* 41.5, pp. 507–525.

Behave (2012). *Behave*. https://behave.readthedocs.io/en/latest/. Accessed: 2025-02-15.

Blochwitz, Torsten et al. (2012). "Functional mockup interface 2.0: The standard for tool independent exchange of simulation models". In: *9th international modelica conference*. The Modelica Association, pp. 173–184.

Chen, Tsong Y et al. (2020). "Metamorphic testing: a new approach for generating next test cases". In: *arXiv preprint arXiv:2002.12543*.

Chen, Tsong Yueh et al. (2018). "Metamorphic testing: A review of challenges and opportunities". In: *ACM Computing Surveys (CSUR)* 51.1, pp. 1–27.

Clark, Andrew G et al. (2023). "Metamorphic testing with causal graphs". In: *2023 IEEE Conference on Software Testing, Verification and Validation (ICST)*. IEEE, pp. 153–164.

Deng, Yao et al. (2021). "BMT: Behavior driven development-based metamorphic testing for autonomous driving models". In: *2021 IEEE/ACM 6th International Workshop on Metamorphic Testing (MET)*. IEEE, pp. 32–36.

Jiang, Mingyue et al. (2014). "Testing model transformation programs using metamorphic testing". In.

Li, Rui et al. (2024). "Metamorphic Relation Generation: State of the Art and Research Directions". In: *ACM Transactions on Software Engineering and Methodology*.

Lindvall, Mikael et al. (2017). "Metamorphic model-based testing of autonomous systems". In: *2017 IEEE/ACM 2nd International Workshop on Metamorphic Testing (MET)*. IEEE, pp. 35–41.

Liu, Huai et al. (2012). "A new method for constructing metamorphic relations". In: *12th Intl. Conference on Quality Software*. IEEE, pp. 59–68.

Lundh, Fredrik (1999). "An introduction to tkinter". In: *URL: www. pythonware. com/library/tkinter/introduction/index. htm* 539, p. 540.

MATLAB/Simulink (2024). *Simulink(R2024a)*. Natick, Massachusetts, United States. URL: https://www.mathworks.com.

Murphy, Christian et al. (2011). "On effective testing of health care simulation software". In: *Proceedings of the 3rd workshop on software engineering in health care*, pp. 40–47.

NoviaRDISeafaring (2024). *VST*. https://github.com/Novia-RDI-Seafaring/fmu-opc-hackathon/tree/main/fmus/loc.

Olsen, Megan and Mohammad Raunak (2018). "Increasing validity of simulation models through metamorphic testing". In: *IEEE Transactions on Reliability* 68.1, pp. 91–108.

Olsen, Megan M and Mohammad S Raunak (2016). "Metamorphic validation for agent-based simulation models." In: *SummerSim*, p. 33.

Segura, Sergio et al. (2016). "A survey on metamorphic testing". In: *IEEE Transactions on software engineering* 42.9, pp. 805–824.

Segura, Sergio et al. (2017). "Metamorphic testing of RESTful web APIs". In: *IEEE Transactions on Software Engineering* 44.11, pp. 1083–1099.

Smart, John Ferguson and Jan Molak (2023). *BDD in Action: Behavior-driven development for the whole software lifecycle*. Simon and Schuster.

Sudheerbabu, Gaadha et al. (2024). "Iterative Optimization of Hyperparameter-based Metamorphic Transformations". In: *2024 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, pp. 13–20.

Weyuker, Elaine J (1982). "On testing non-testable programs". In: *The Computer Journal* 25.4, pp. 465–470.

Wynne, Matt and Aslak Hellesoy (2012). *The cucumber book: behaviour-driven development for testers and developers*. Pragmatic Bookshelf.

The Value of Enforcing a Strict Modeling Methodology within Modelica

Dirk Zimmer[1]

[1]Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Germany,
dirk.zimmer@dlr.de

Abstract

Self-restriction to a certain modeling style can enable the modeling of large-scale systems and the robust modeling of complex system architectures. This paper discusses how such a self-restriction can be achieved within the Modelica language and provides a corresponding example.

Keywords: Modeling Methodology, Language Design, Object-Oriented

1 Motivation

Modelica provides the modelers with a lot of freedom on how to create and compose their models. This freedom contributed significantly to the appeal of the language. It enabled many modelers to solve even exotic problems while the powerful algorithms for symbolic index-reduction in Modelica compilers still enable the generation of efficient simulation code. Hence Modelica has become the language of choice for many experts in their fields, especially in thermal and energy related fields where many specialized solutions are needed.

As always, this freedom is associated with a price to be paid for it. First, what is appealing to an expert maybe overwhelming for a novice. Beginners in Modelica often struggle with decrypting the error-messages and often yearn for more guidance. Even after 25 years, the entry barrier to the world of equation-based modeling is high. Second, since equations can be entered in any form, the code generation has to be relegated to the very last stages in processing. For monolithic simulation code, this is fine but code for large-scale systems on GPUs or for variable structure systems require differently structured code that becomes available before final system assembly. This is difficult to fulfill for current Modelica models, despite notable efforts from (Benveniste 2023, Neumayr 2023).

Being self-aware of its flaws, the Modelica community has always remained open to take inspiration from new language developments. Whether these are experimental languages like MOSILAB (Nytsch-Geusen 2006), SOL (Zimmer 2010) or larger efforts such as Modia (Elmqvist 2021). However, these developments also revealed how strong the Modelica legacy is and that maybe a new language or a major extension is not what is needed.

Indeed, this paper argues that not an extension of the Modelica language is needed but that a self-imposed restriction would benefit a significant set of applications. The conflict between an expert and a novice does not need to exist. We can formulate basic models by restricted means and then let the experts extend from this giving them free choice of their tools. The conflict between different possible compile paths also does not need to exist. We can provide models in a restricted form that enable local analysis and local compilation and when these restrictions are lifted, full global analysis for elaborate index-reduction is enabled.

We shall remind ourselves that it is the strength of a declarative language that it does not stipulate its computational realization. When we are stricter about the declarative form, we typically gain freedom on how to process the content.

2 Robust Modeling Methodology

The greatest source of complexity in the processing of equation-based models are systems that are over-idealized. Over-idealization means that the system has been idealized so strongly that the information on how to solve this system has been lost in the process of modeling. A great educational collection of such examples has been provided by Peter Junglas (Junglas 2025):

- A singular model of gear shift for a car transmission: the over-idealization neglected the necessary synchronization during the shift.
- A singular model of an RS Flip-Flop: the over-regularization neglected the signal delays within the components
- A model for the distribution of compressed air with unsolvable non-linear equations: the over-idealization neglected the kinetic energy of the fluid that would lead to the flow balance.
- etc.

Unfortunately, the formulation of over-idealized systems has a strong legacy in text-books and modeling. To make matters worse the awareness of this phenomenon is often little to none. Over-idealized systems not necessarily lead to singular systems but often also to structural regular systems involving non-linear equation system in implicit

form and states with non-linear constraints between potential state variables.

Because the information on how to solve these models has been removed, a compiler for such Modelica models has essentially to perform blunt guessing (often euphemistically formulated as "applying clever heuristics"). No matter how elaborated such guesswork may be, it involves severe disadvantages:

- It needs be performed on a global level to exploit the full context and hence a global analysis of the model is needed.
- It does not scale. While guessing may work for system of low dimensions, it will eventually fail for increasingly large dimensions.
- The compile process becomes non-deterministic. Different compilers may have different guessing strategies.

It is important to emphasize that this is not a problem of the compiler's algorithms or of any of the simulation tools. It is a problem of the input to the compiler. If the information entering the system is incomplete, no algorithm can magically retrieve this missing information. This is basic information theory.

The only solid solution to this problem is thus to insist on input that is information complete, meaning models that are idealized (as all models are idealizations) but not over-idealized.

Fortunately, a suitable model class has been recently identified. We call them Linear Implicit Equilibrium Dynamics (LIED), (Zimmer 2024)

A DAE system with potential state derivatives $\dot{\boldsymbol{x}}$, time t and algebraic variables $\mathbf{w}$

$$\mathbf{0} = \boldsymbol{F}(\dot{\boldsymbol{x}}, \boldsymbol{x}, \mathbf{w}, t)$$

is defined as LIED system when it can be transformed into the following form:

$$\begin{bmatrix} \mathbf{w}_E \\ \dot{\boldsymbol{x}}_E \end{bmatrix} = \mathrm{g}(\boldsymbol{x}_I, \boldsymbol{x}_E, t)$$

$$\mathbf{A}(\boldsymbol{x}_I, \boldsymbol{x}_E, \mathbf{w}_E) \begin{bmatrix} \mathbf{w}_I \\ \dot{\boldsymbol{x}}_I \end{bmatrix} = \mathrm{f}(\boldsymbol{x}_I, \boldsymbol{x}_E, \mathbf{w}_E, t)$$

We see that both the algebraic variables as well as the state derivatives can be split into a fully explicit part $(\dot{\boldsymbol{x}}_E;\ \mathbf{w}_E)$ and a part $(\dot{\boldsymbol{x}}_I;\ \mathbf{w}_I)$ with a linear system in implicit form expressed by the regular matrix $\mathbf{A}$. Furthermore, the following conditions shall hold true:

$$\dot{\boldsymbol{x}}_E \cap \dot{\boldsymbol{x}}_I \subseteq \dot{\boldsymbol{x}}$$

$$\mathbf{w}_E \cap \mathbf{w}_I \supseteq \mathbf{w}$$

$$\dot{\boldsymbol{x}}_E \cap \dot{\boldsymbol{x}}_I \cap \mathbf{w}_I \supseteq \dot{\boldsymbol{x}}$$

$$\dot{\boldsymbol{x}}_E, \dot{\boldsymbol{x}}_I, \mathbf{w}_E, \mathbf{w}_I \text{ are all disjoint}$$

These conditions essentially mean that it is allowed to perform certain symbolic mechanism of index reduction such as the dummy derivative method (Mattsson1993) originating from Pantelides (Pantelides 1988). Using this method, states variables of $\boldsymbol{x}$ can be transformed to algebraic variables in $\mathbf{w}_I$ and further derivatives may be added to $\mathbf{w}_I$ or $\mathbf{w}_E$. In practice, this is important because it means that the linear implicit dynamics can be expressed by far fewer states than suggested by the vector $\boldsymbol{x}$ of the original DAE formulation.

Whereas this model class may seem to be overly restrictive in the first place we could demonstrate that it is not only applicable but also highly attractive for the modeling of stiff mechanical systems as in Figure 2 (Zimmer 2023) and complex thermo-fluid architectures (Zimmer 2022, Junglas 2023) as in Figure 1. Further library development for energy and electric systems is currently ongoing.

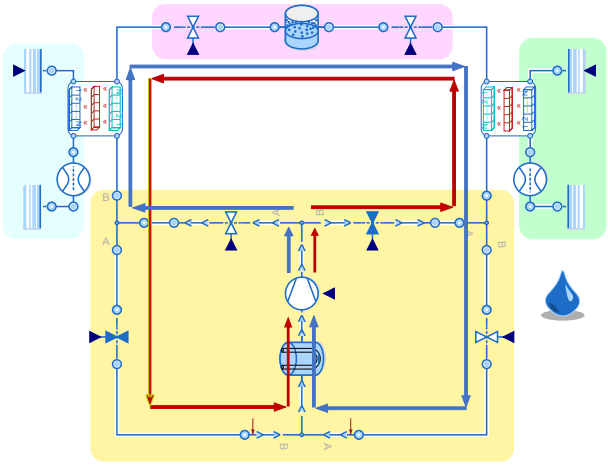
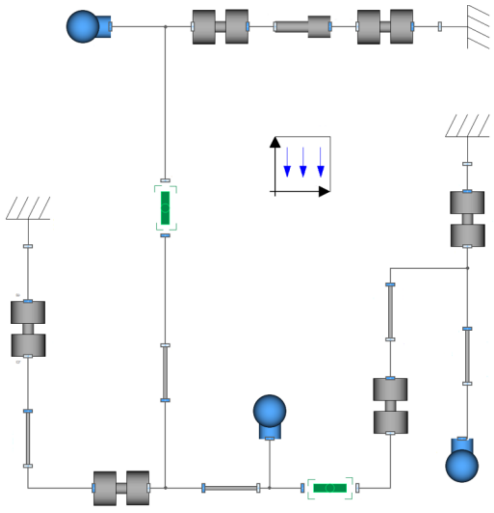


Figure 1. Model diagram of a heat pump using the DLR ThermoFluid Stream library (Zimmer 2022)



DOI 10.3384/ecp218147

Figure 2. Model diagram of a mechanical system for a kinematic using the Dialectic Mechanics library (Zimmer 2023).

The goal is now to define a language that is restricted for such LIED systems and that also represent valid Modelica code. The expected advantages are naturally contrarian to the previous bullet list for over-idealized models:

- Models can be analyzed locally and early compilation of components is enabled.
- It scales. The dimension of the system will principally not impact its solvability barring other issues such as numerics or memory consumption.
- The compile process can be formulated in fully deterministic way increasing the value of a reference implementation.

3 Demonstration Example

Whereas the two preceding sections made very abstract arguments, we shall now provide a concrete example and then discuss the design ideas based on this code. Listing 1 was placed in the Appendix and presents the model of a 2D multibody library implementing components according to dialectic mechanics. The code has been stripped from most annotations, occasionally shortened and simplified for presentation purposes.

3.1 Assumed Causality

A key point of the robust modeling approach is that models are formulated under the assumption of causality. This enables two major benefits:

1. Code generation for individual components is enabled.
2. Error messages point to a single line holding either a equation whose causality is not correct or to a variable definition that is not causalized.

To achieve this, the connectors need to have an assumed causality and what derivates are needed. Also, we need to be able to formulate what is a state and what variables can be used to tear a linear equation system. Let us go through each of these points:

Connectors with assumed causality

Input and outputs are the natural way to formulate a causal connector. Whenever inputs and outputs are used we will use them as indicator of causality: the input is assumed to be known to its component and the output is unknown (and the reverse is true to the outside).

For the modeling of physical systems, conjugated pairs of potential and flow variables have proven to be useful. In order to formulate these pairs with an assumed causality, they are not allowed to stand on their own but need to be associated with a signal connector.

A potential variable may be declared in a connector after a signal and then inherits the assumed causality from this signal. The flow variable must then be declared after the potential variable and is assumed to be of inverse causality.

Two examples of such connectors are found in Listing 1, lines 35-51. The `FlangeOn` is supposed to be on the root side of a kinematic chain and hence its variable for position forms an output signal. The corresponding potential is the velocity and shares this causality. The flow is the force (or torque respectively and of inverse causality). The `FlangeTo` forms then the connector of opposite sex.

The example demonstrates another feature defined on Listing 1, line 14 and used in lines 30-34. A new special predefined type called `RealWithDerivative` that indicates that not only the potential variable is available in the connector but also its derivative. Providing this information means that differential index-reduction can be performed on the component level.

For a standard Modelica compiler, it is not necessary to have such a special type hence the definition on line 15 simply defines a `Real`. The definition exists so that other compilers or a compliance checker can replace this default definition by a special inner representation.

Given such causalized connectors, the model equations can now be provided in partially ordered form: each equation can be causalized based on the previous equations. A simple example for this is the body model in Listing 1, lines 130 – 150. It assumes the position and the derivatives to be known and calculates the forces and torques from the acceleration.

Declaring Tearing of Linear Equation Systems

Since the modeler shall formulate everything under the assumption of causality, this also impacts how implicit linear equation systems across components have to be formulated. Often the modeler expects the model to react to a variation by a linear response. This observation enables to formulate such a system in causalized and torn form. To indicate a tearing of a linear equation system, a special predefined model is being declared and then applied. The model is called `LinearResponse`.

The revolute joint element shows how to use it. The modeler expects that a linear response in force results from a change in acceleration. He formulates it accordingly by declaring the model `LinearResponse`. The assumed causality of the model now naturally follows from this.

Within Modelica the `LinearResponse` model can be formulated as in Listing 1, line 6-12. This definition ensures that the model will be correctly understood by any existing Modelica tool. Special tools for this modeling

methodology may replace this definition with an internal representation.

Declaring States of the System

A predefined type is also used to select the state of the system. In this case the state variable is declared as `ContinuousState`.

The obvious question is why using this mechanism and not the already available `StateSelect` attribute? The reason is to enable an orthogonal mechanism for state selection.

The special model is used for the restricted modeling methodology presented here and aims to enable component-wise compilation. The state-select attribute is to be set by an expert for full complex (non-restricted) Modelica models. The motives for selecting states can be quite different and hence it seems a good idea to avoid any potential conflict.

When translating a model, the state variable is assumed to be known and the derivative is assumed as unknown. If you take the derivative of any non-state variable than this variable has to be known at this point and the derivative is derived from this. The `Body` component is an example from this. Its derivative does not define a state.

3.2 No Look-ahead

In our restricted modeling methodology, we shall declare or define items before we use them. Basically, a compiler or a compliance checker for this restricted set shall be able to "understand" each line without having to read ahead. This concession significantly reduces the complexity of a compiler since many processes can now be performed directly on the stack instead of having to work on intermediately stored parts. Far less internal data-structures are thus needed.

One result of this requirement is that packages must not be mutually depending on each other anymore. The common pattern that is used by many existing Modelica libraries does not match this requirement. We have to abandon this pattern and separate interfaces from components from examples in order to shed unnecessary dependencies.

We can see this in Listing 1 where packages for interfaces, planar components and examples are separated and all encapsulated. Import statements are restricted to encapsulated packages. By separating interfaces, components and examples the interdependencies are reduced.

While providing a short-term inconvenience, this restriction leads to code that is better maintainable in the long-term and avoids "spaghetti" packages. Even a modern language like Go enforces such a rule for good reasons.

3.3 Feature Elimination

When we restrict ourselves to a certain modeling style, this also offers an opportunity to trim the feature set of the language. There are a number of small features which have only very specific purpose or application in the standard Modelica language that we shall abandon for this modeling methodology:

- connections.branch/root (used mostly in MultiBody)
- stream (used in Modelica.Fluid)
- operator overload
- expandable connectors
- delay
- homotopy
- inline
- etc.

As the reader can see from the above list of abandoned language features, it is back to basics again. Another significant simplification is that annotations can be part of the code but are completely ignored for the process of compliance checking or compiling. The information within the annotation is hence only regarded as auxiliary meta-information for non-compilation processes as it was the original intention in early Modelica (this is also an additional reason why to use special predefined types and models such as in lines 2-19 rather than custom annotations)

For the moment, the focus is also on continuous systems but of course, discrete events also need to be handled. This shall follow the same guidelines yielding a fully deterministic process that can be locally analyzed. However, this is still upcoming work.

3.4 Compatiblity to Modelica

Evidently, the restrictions that are imposed to the modeler are significant. The Modelica Standard Library clearly does not comply to these restrictions and other existing libraries are also unlikely to do so.

We hence need new libraries developed under this methodology. This is no surprise because there was previously no sufficiently practical guidance on how to model in an object-oriented way that is also information complete.

What is important however that libraries that are developed under these restrictions are fully usable with any tool supporting the Modelica standard. Even if these tools do not exploit the special characteristics of the restricted modeling methodology, they still profit from the fact that these models are information complete. Not only shall these libraries work in any Modelica tool, they shall also be (automatically without further tool support) be very robust in their usage.

DOI 10.3384/ecp218147

We currently engage in the development of 4 libraries:

1. Signal Blocks
2. Dialectic Mechanics
3. Controlled Energy Flows
4. ThermoFluid Stream Lite

4 Implementation

To support the development of these new libraries, a Compliance checker is being implemented. It serves two functions:

1. The implementation formalizes the restrictions that have been informally stated above.
2. It provides instant feedback to modelers whether their library abides to these restrictions.

Formalizing the restricted modeling methodology will ultimately result in a new language that forms a sub-set of Modelica. The compliance checker is thus not developed from an existing Modelica compiler or tool but as a new tool from scratch. This is the most effective way to shed complexity and question all the existing requirements.

The compliance checker is implemented as a stand-alone console application in C++17 using only the Standard Template Library (STL). Given that the portability of C++ code between operating systems and compilers has significantly improved over the last decade the application may be applied on different operating systems.

We can use the compliance checker to keep track of the complexity of the new restricted language. Figures 3 and 4 measure the complexity of the C++ code compared to the Modelica specification which is written in English language. The counting for the code includes comments and the counting for the specification includes also the examples.

Although we compare an executable code against a mere description in natural language, we still have far less complexity. This shows that (at least for now) the reduction in complexity is very effective.

How is this compliance checker supposed to be used? As a console application it runs next to the Modelica development environment of your choice and automatically checks whenever a modification is saved whether the file(s) comply to the restricted modeling methodology. Otherwise it provides an error message that always refers to a specific line of code.

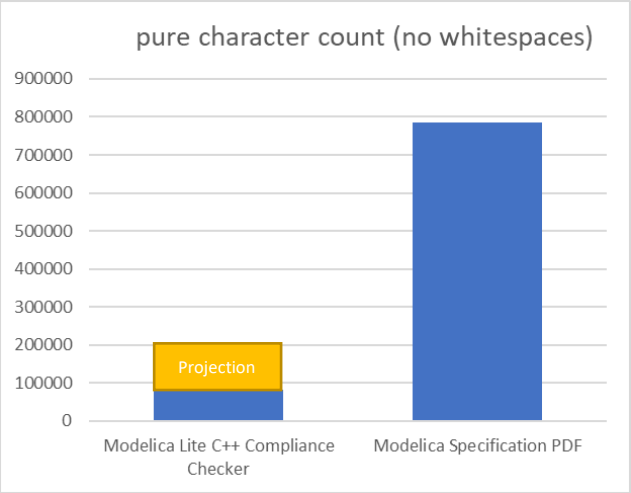


Figure 3. Complexity of the compliance checker implementation in C++ for a restrictive Modelica vs Modelica Specification PDF version in character count. Projection indicates an estimate for the complete compliance checker.

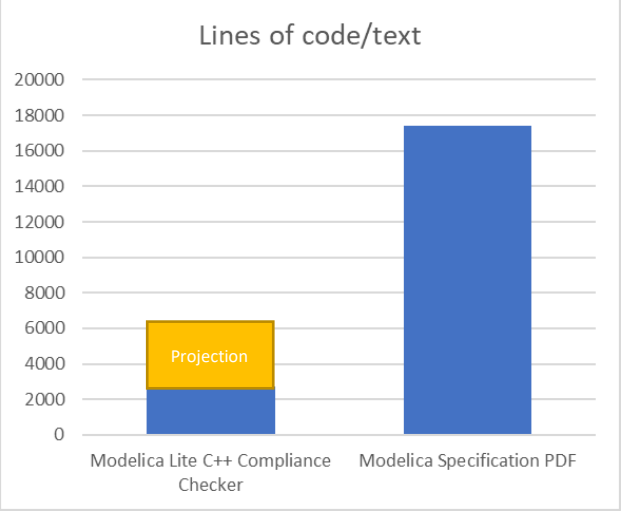


Figure 4. Complexity of the compliance checker in C++ for a restrictive Modelica vs Modelica Specification PDF version in lines of code.

In this way, the existing IDEs for Modelica can be effectively reused but the modeler is restricted with instant feedback. In future, the compliance checker can be extended to enable different compile paths. After all the compliance checker already performs many processing steps of a full compiler such as: lexer, parser, type generation, identifier resolution, partial casualization, etc.

5 Conclusions

Modelica has gained its popularity by providing the modeler ample freedom in the formulation of its models.

However, there are many applications that profit from a more restrictive modeling methodology:

- For complex architectures, (especially when modeled across different organizational entities) the robustness of the model often becomes the limiting factor in model development. If a strict modeling methodology allows stronger a-priori statements on solvability and pinpoint error messages, it will be highly welcome.
- For large scale systems or variable structure systems, a global analysis of the system increasingly becomes infeasible. If a strict modeling methodology requires only a local analysis and enables compilation of components, it will be of high value.
- Modern compute architectures like GPUs (Kirk 2022) demand different forms of code generation. Modern numerical methods such as multi-derivative methods (Krivovichev 2024) also require different compilation schemes. If a strict modeling methodology eases the compiler development by massively reducing complexity, it will find quick adaptation.

As shown in the previous section, it is possible to support such a strict methodology fully within the current version of Modelica without demanding any changes to the existing language. It is basically an act of self-discipline. Some models may seem unusual for a Modelica purists but in view of the listed advantages, one should advocate for pragmatism.

The currently undergoing implementation of a compliance checker will ease this discipline, formalize the restrictions and open up the development of independent compile paths. We plan to release the compliance checker as open-source software.

In this way, we can have the best of both worlds. Confined modeling efforts for special systems can be performed by experts using the full feature-set of Modelica. Modeling efforts of larger architectures or with versatile compute architectures can be performed using a strict modeling methodology such as the one presented here.

Acknowledgements

This work was partially supported by the ITEA4 research project OpenScaling with financial support of the German federal ministry of education and research referring to the grant number: 01IS23062C

SPONSORED BY THE

Federal Ministry of Education and Research

References

Benveniste, A. B. Caillaud, M. Malandain, J. Thibault (2023) Towards the separate compilation of Modelica: modularity and interfaces for the index reduction of incomplete DAE systems. *Proceedings of the 15th International Modelica Conference*, Aachen.

Elmqvist H., M. Otter, A. Neumayr and G. Hippmann (2021) Modia - Equation Based Modeling and Domain Specific Algorithms *Proceedings of the 14th International Modelica Conference*, Linköping.

Junglas, P. (2025) Mathematical Problems due to Oversimplification *Keynote at the ASIM Workshop* STS, GMMS, EDU. https://www.peter-junglas.de/fh/talks/2025-oberpfaffenhofen/oversimp/html/index.html

Junglas, P. (2023) Implementing Thermodynamic Cyclic Processes Using the DLR Thermofluid Stream Library. *Simulation News Europe* E 33(4)

Kirk, David B., Izzat El Haji, Wen-mei W. Hwu (2022). *Programming Massively Parallel Processors: A Hands-on Approach*. Morgan Kaufmann, 4th Edition.

Krivovichev, G. V. (2024). Stability Optimization of Explicit Runge–Kutta Methods with Higher-Order Derivatives. *Algorithms*, 17(12), 535. https://doi.org/10.3390/a17120535

Mattsson, S.E., Gustaf Söderlind (1993). "Index Reduction in Differential-Algebraic Equations Using Dummy Derivatives" In: *SIAM Journal on Scientific Computing* 1993 14:3, 677-692

Neumayr, A.; Otter, (2023) M. Modelling and Simulation of Physical Systems with Dynamically Changing Degrees of Freedom. Electronics, doi.org/10.3390/electronics12030500

Nytsch-Geusen, C. et al. (2006) Advanced Modeling and Simulation Techniques in MOSILAB: A System Development Case Study. In *Proc. of the 5th International Modelica Conference*, Vienna Austria, Vol. 1, pp. 63-71.

Pantelides, C. (1988), The consistent initialization of differential-algebraic systems, *SIAM J. Sci. Statist. Comput*., 9, 213–231

Zimmer, D. (2010), *Equation-Based Modeling of Variable Structure Systems*, PhD-Thesis, ETH Zurich.

Zimmer, D. (2020), Robust Object-Oriented Formulation of Directed Thermofluid Stream Networks . *Mathematical and Computer Modelling of Dynamic Systems*, Vol 26, Issue 3.

Zimmer, D., N. Weber, M. Meißner (2022) *The DLR ThermoFluid Stream Library*. MDPI Electronics - Special Issue.

Zimmer, D., C. Oldemeyer (2023). "Introducing Dialectic Mechanics". *Proceedings of the 15th International Modelica Conference*, Aachen.

Zimmer (2024) Object-Oriented Modeling of Classic Physical Systems using Linear Implicit Equilibrium Dynamics *Preprints* 2024, 2024031139

Appendix

Listing 1. Example code

```
encapsulated package Base "Base elements”

  type ContinuousState = Real;

  model LinearResponse "continuous state"
    input Real residual;
    output Real balance;
  equation
    residual = 0;
     annotationdefaultConnectionStructurallyInconsistent=true);
  end LinearResponse;

  type RealWithDerivative = Real;
  type RealWith2Derivatives = Real;

  package Units
    [...]
  end Units;

end package Base;


encapsulated package Mechanical

  encapsulated package Interfaces
    import B = Base;
    import SI = Base.Units.SI;

    type VelocityInt = B.RealWithDerivative
                         (unit = "m/s")
    type AngularVelocityInt =
      B.RealWithDerivative(unit = "rad/s")

    connector FlangeOnPlanar
      output SI.Position r0[2];
      output SI.Angle phi;
      VelocityInt v0[2];
      AngularVelocityInt w;
      flow SI.Force f0[2];
      flow SI.Torque t;
    end FlangeOnPlanar;

    connector FlangeToPlanar
      input SI.Position r0[2];
      input SI.Angle phi;
      VelocityInt v0[2];
      AngularVelocityInt w;
      flow SI.Force f0[2];
      flow SI.Torque t;
    end FlangeOnPlanar;

    [...]
  end package Interfaces;

  encapsulated package Planar
    import B = Base;
    import Interfaces = Mechanical.Interfaces;
    import SI = package Base.Units.SI;
```

```
    package Motion

      model Root "Frame fixed"

        parameter SI.Position r0_abs[2] = {0,0}
        "Fixed absolute x,y-position";
        parameter SI.Angle phi_abs = 0
        "Fixed angle";
        Interfaces.FlangeOnPlanar flangeOnP;

      equation
        flangeOnP.r0 = r0_abs;
        flangeOnP.phi = phi_abs;
        flangeOnP.v0 = {0,0};
        flangeOnP.w = 0;
      end model Root;

      model RevoluteJoint "Ideal Revolute Joint"
        Interfaces.FlangeToPlanar flangeToP;
        Interfaces.FlangeOnPlanar flangeOnP;
        parameter SI.Time TD
        "dialectic time constant";

      protected
        B.LinearResponse Cut;

        B.ContinuousState phi(unit="rad")
        "position of component";
        B.ContinuousState w(unit="rad/s")
        "(kinetic) velocity of component";
        SI.AngularAcceleration z
        "acceleration of component";
        SI.Torque t "actuation torque";

      equation

        z = Cut.balance;

        //dialectic damping term
        der(phi) = w + z*TD;
        der(w) = z;

        //rigidly connect positions
        flangeOnP.r0 = flangeToP.r0;
        flangeOnP.v0 = flangeToP.v0;
        flangeOnP.phi = flangeToP.phi + phi;
        flangeOnP.w = flangeToP.w + w;

         //balance forces
        flangeToP.f0 = -flangeOnP.f0;
        flangeToP.t = -flangeOnP.t;
        t = flangeOnP.t;
        Cut.residual = t;

      end RevoluteJoint;

    end Motion;
```

```
128    package Parts
129
130      model Body
131        Interfaces.FlangeToPlanar flangeToP;
132        parameter SI.Mass m  "Mass of the body";
133        parameter SI.Inertia J "Inertia";
134        parameter SI.Acceleration g[2]
135        "Gravity";
136        parameter Boolean enableGravity = true;
137
138      protected
139        SI.Acceleration a0[2] "Acceleration";
140        SI.AngularAcceleration z
141        "Angular acceleration";
142
143      equation
144        a0 = der(flangeToP.v0);
145        z = der(flangeToP.w);
146        flangeToP.f0 =
147          if enableGravity then m*a0-m*world.g
148          else m*a0;
149        flangeToP.t = J*z;
150      end Body;
151
152    end Parts;
153
154  end Planar;
155
156  encapsulated package Examples
157    import Planar = Mechanical.Planar;
158    [...]
159
160  end package Examples;
161
162end package Mechanical;
```


DOI
10.3384/ecp218147

Context-Oriented Equation-based Modeling in ModelingToolkit.jl

Christian Gutsche[1, 2] Christoph Seidl[2] Volodymyr Prokopets[2]
Sebastian Götz[2] Zizhe Wang[1, 2] Uwe Aßmann[2]

[1]Boysen–TU Dresden–Research Training Group, Dresden, Germany
[2]Chair of Software Technology, Technische Universität Dresden, Dresden, Germany
{christian.gutsche, christoph.seidl, volodymyr.prokopets, sebastian.goetz1,
zizhe.wang, uwe.assmann}@tu-dresden.de

Abstract

Cyber-physical systems are self-adaptive, changing their behavior at run time to adapt to their context. Hence, their simulations must also handle variability at run time. The lack of support for variability in industrial equation-based modeling languages, such as Modelica, causes problems when simulating self-adaptive systems, e.g., they only limitedly support structural variability, and state transitions are based on if-then-else conditions that can cause conflicts, especially for complex control mechanisms.

We present a modeling technique for equation-based models containing variability by implementing concise and dedicated language constructs to express state space and transitions via contextual modeling. Contextual modeling abstracts the modeled world and allows the definition of constraints that reduce the risk of reaching conflicting states. We demonstrate the feasibility of our approach on a case study, presenting the advantages of our modeling technique regarding the definition of state control and the reduction of risk for reaching conflicting states.

Keywords: Cyber-physical systems, Self-Adaptive Systems, Contexts, VSS, Julia, Simulation

1 Introduction

The 5C architecture of cyber-physical systems (CPSs) (Lee, Bagheri, and Kao 2015) includes a configuration level that implies system behavior variability. Therefore, when simulating CPSs, their simulations must also handle variability. Variability at simulation time is supported only partially by the equation-based modeling (EBM) language Modelica (Fritzson and Engelson 1998), e.g., Modelica does not support conditional definitions of components during simulation (Zimmer 2010) and has limitations with switching equations at simulation time if the DAE index changes (Benveniste, Caillaud, and Malandain 2020). Even if these limitations are bypassed, Modelica supports only if-then-else or when conditions based on Boolean expressions. These expressions become complex and incomprehensible for complex state spaces and transitions. This lack of expressiveness results in an inconvenient definition of variability in simulation models.

In this paper, we present context-oriented equation-based modeling (COEBM), a modeling technique for EBM containing variability by implementing concise and dedicated language constructs to express state space and transitions via contextual modeling. Contexts abstract the simulated world and possible control actions. Constraints between contexts are checked during runtime, preventing reaching conflicting states, e.g., simultaneously charging and discharging a battery. During simulation, context changes imply transitions between models to achieve structural variability. We further develop *Contexts.jl*, presented in our previous work (Gutsche et al. 2024), to achieve better applicability of context-oriented programming (COP) and context modeling in EBM.

For our implementation, we use the EBM language *ModelingToolkit.jl* (MTK) (Ma et al. 2021). MTK is implemented as a library-based embedded domain-specific language (DSL) for the general-purpose programming language Julia (Bezanson et al. 2015; The Julia Project 2023). This enables utilizing Julia's features for extending the modeling language (Rackauckas and Nie 2019) and using the expressiveness of a programming language in callbacks during simulation time. We present how to extend MTK with syntax suitable for COEBM.

The main contributions of this paper are:

1. Integrating COP in EBM for concise syntax for variable reinitialization
2. Integrating *Contexts.jl* language constructs in MTK to define structural variability
3. Using context modeling with constraints to prevent conflicting states in EBM and implementing state control mechanism

We present a case study to evaluate the contribution's feasibility for simulating self-adaptive systems. Therefore, we model an energy park for green hydrogen production with COEBM. We port Modelica models to MTK and implement the control logic with contextual modeling. Simulation speed is crucial when using simulations to support CPS operation. Therefore, we also provide performance measurements for our COEBM implementation.

In section 2, we present background information on Julia and context modeling, followed by an overview of related work about variable structure system (VSS) simulations in EBM in section 3. In section 4, we present the concept of COEBM and how to simulate systems containing variability with COEBM. The evaluation, consisting

of the mentioned energy park case study and performance measurements, is presented in section 5. To conclude, we provide a discussion in section 6 and a conclusion and outlook for future work in section 7.

2 Background

2.1 Julia and ModelingToolkit.jl

Julia (Bezanson et al. 2015; The Julia Project 2023) is a just-in-time compiled general-purpose programming language. A major difference from other commonly used programming languages is the multiple dispatch paradigm. Instead of defining functions within classes and dispatching over the function's object, the dispatch relies on the types of the function's arguments. Some of the advantages of this design choice are discussed later. Besides this, Julia has powerful metaprogramming capabilities.

In addition, Julia provides the rich ecosystem of scientific machine learning (SciML) containing libraries for mathematical computation and optimization (Lubin et al. 2023), numerical equation solvers (Rackauckas and Nie 2017), and EBM (Ma et al. 2021). The implementation in Julia enables utilizing multiple dispatch, which allows developers to, thanks to design choices of *DifferentialEquations.jl*, extend the functionality of the SciML ecosystem (Rackauckas and Nie 2019).

MTK is an embedded DSL implementing Modelica-like object-oriented EBM in Julia. MTK components consist of variables, parameters, equations, sub-components, callbacks, and Julia code. Models defined with the macro `@mtkmodel` are translated to equation systems. Depending on the mathematical structure, MTK provides, among others, `ODESystem` and `PDESystem`. These systems are used to define `ODEProblem`s containing initial values and a time span. `ODEProblem`s are solved with an implemented `solve` function where different solver algorithms are available. MTK benefits from support for extensibility of the SciML ecosystem (Rackauckas 2021). We implement our modeling technique in MTK because of this support for extensibility and the possibilities for combining Julia language features with EBM.

2.2 Context Modeling

Contexts describe the situation and environment in which a program is executed as discrete categories. Context-awareness (Abowd et al. 1999) enables software to achieve dynamic adaptation of its behavior to the current situation by specifying behavior for different contexts. Contexts are activated and deactivated based on the global state of the program or its environment. We call this state activeness. A programming paradigm based on this principle is called COP (Keays and Rakotonirainy 2003; Hirschfeld, Costanza, and Nierstrasz 2008). In COP, developers can specify contextual variants of a function. During the program's run time, the current activeness of contexts is evaluated, calling the function defined for this context. Therefore, dynamic adaptivity is achieved.

In our previous work (Gutsche et al. 2024), we presented a COP implementation in Julia. We introduced the library *Contexts.jl*, which provides concise syntax based on metaprogramming to define contexts and achieve context-dependent behavior for functions by utilizing Julia's multiple dispatch. In (Gutsche et al. 2024), we also already showed that combining *Contexts.jl* and MTK to implement variability in equation-based models with contexts is possible.

As presented in (Gutsche et al. 2024), context modeling in *Contexts.jl* is based on singleton objects representing contexts. The activeness of contexts can be constrained by an approach based on context Petri nets (Cardozo et al. 2012). The Petri nets defined in *Contexts.jl* are Dynamic Feature Petri nets (Muschevici, Clarke, and Proença 2010) that represent the constraints from (Cardozo et al. 2012). For example, an exclusion constraint prevents two specific contexts from being active simultaneously, and weak inclusion constraints imply that the activation of one context causes the activation of another context. This reduces the risk of reaching conflicting states.

3 Related Work

Several approaches aim to increase functionalities for specifying and simulating variability in EBM. As (Wang et al. 2025) show, these approaches focus on enabling EBM to simulate models containing structural variability, called VSSs (Utkin 1977).

Several new languages were designed, heavily inspired by Modelica, including Sol (Zimmer 2010), Hydra (Giorgidze 2012), and Modelyze (Broman and Siek 2012). While they support structural variability, advanced modeling techniques to specify state spaces and control mechanisms are not implemented. The already mentioned MTK and *Modia.jl* (Elmqvist, Neumayr, and Otter 2018) are new EBM implementations, implemented as embedded DSLs in Julia. In contrast to MTK, *Modia.jl* has dedicated support for VSS simulation (Neumayr and Otter 2023) but also lacks advanced modeling techniques.

DySMo (Mehlhase 2014), MoVaSe (Esperon, Mehlhase, and Karbe 2015), and PyVSM (Stüber 2017) rely on scripts that orchestrate the simulation. DySMo and PyVSM use Python to simulate different models, defining variants, e.g., modeled with Modelica. While they enable VSS simulation, they lack modeling techniques for constructing state space and transitions. MoVaSe, later called MoVaSim, is implemented in the Eclipse ecosystem and includes modeling techniques to specify model variants. However, the only available references to MoVaSim are a Work-in-progress paper (Esperon, Mehlhase, and Karbe 2015) and a dissertation (Gómez Esperón 2017). The editor itself is not available. In these approaches, variability modeling and state control are performed in non-EBM technical spaces, namely Python and Eclipse, creating a hurdle for users.

OM.jl (Tinnerholm, Pop, and Sjölund 2022) is a Mod-

DOI 10.3384/ecp218155

elica compiler written in Julia. *OM.jl* adds syntax for VSS allowing specifying different modes and their transitions. However, also *OM.jl* lacks advanced modeling to specify mode transitions. An approach that aims to improve this modeling is presented in (Wang et al. 2025), where a preprocessor is used to implement syntax for specifying contexts that control the modes. The preprocessed code is then compiled by *OM.jl*. However, only one context can be active at a time, and context activeness is controlled by Boolean conditions based on the time value. Therefore, the advantage of using contexts is lost.

We identify a gap: none of them supports both VSS simulation and advanced modeling of the state space and control mechanisms with dedicated syntax.

4 Context-Oriented Equation-based Modeling

We use contexts in COEBM to achieve behavior variability. Therefore, a well-defined context model is the base of COEBM. This section first describes the modeling of such context models. Afterwards, we present two ways to use contexts and COP for variability: variable reinitialization and structure change.

4.1 Contexts in Equation-based Modeling

We classify contexts into measurable and control contexts. Measurable contexts change their activeness based on measured values of observables, e.g., a temperature. Those observables cannot be directly influenced. Control contexts describe states or variables controlled by operators, e.g., a variable voltage. By changing the context, the operational mode changes.

In (Gutsche et al. 2024), we presented how *Contexts.jl* enables the definition of contexts with concise syntax based on metaprogramming and how constraints based on (Cardozo et al. 2012) are used to describe the state space. Using primarily measurable contexts to imply the activation and deactivation of control contexts via constraints enables the definition of control mechanisms while minimizing the risk of reaching conflicting states.

We introduce a new constraint for contexts, the `alternative`. This constraint implies that out of N contexts, exactly 1 context must be active at any time. This constraint occurs frequently in modeling but was not described by (Cardozo et al. 2012) or (Gutsche et al. 2024). We implemented the constraint in *Contexts.jl* using a Petri net like the library's other constraints. The constraint checks two things when a context's activeness is changed. If the currently only active context gets deactivated, this deactivation will be undone, and if a context $C1$ gets activated, but an alternative context $C2$ was active before, $C2$ becomes deactivated.

Another concept that we introduce is context groups. A context group implies an `alternative` constraint between the contexts that compose the group. Calling the group object will return the currently active context of this group. Context groups should describe orthogonal context dimensions to prevent conflicts. We also implemented context groups in *Contexts.jl* to improve the applicability for COEBM. An example of how to define contexts within a context group is shown in Listing 1.

```
1 @newContext Cold, Medium, Warm
2 activateContext(Medium)
3 Temp = ContextGroup(Cold, Medium, Warm)
4 Temp() # returns Medium
```

Listing 1. Example for defining contexts `Cold`, `Medium`, and `Warm` as well as a context group `Temp`. These contexts describe the Temperature as a discrete classification.

Modeling the context space should consist of four steps. First, identify relevant contexts. Second, cluster contexts to context groups. Third, classify contexts as measurable or control contexts. Fourth, define constraints between contexts of different groups. The activation and deactivation of measurable contexts may imply the activation and deactivation of control contexts, but not vice versa.

An example is a temperature control system. For the sake of simplicity, we assume that the system either activates heating or cools the system by activating the air conditioning (AC). These two possible operations create four contexts: `HeatingOn` and `HeatingOff`, composing the group `Heating`, and `ACOn` and `ACOff`, composing the group `AirConditioning`. For a basic control mechanism, the activeness of heating and AC might only automatically depend on the temperature. We define a context group `Temperature` consisting of the contexts `Cold`, `Medium`, and `Warm`.

While the temperature contexts are measurable, the contexts for heating and AC are control contexts. We model the control contexts' activeness based on the activeness of the temperature contexts. Therefore, we specify the following constraints:

- `Cold` weakly includes `HeatingOn`
- `Medium` weakly includes both, `HeatingOff` and `ACOff`
- `Warm` weakly includes `ACOn`

Note that because `Cold` and `Warm` exclude each other, `ACOn` and `HeatingOn` will never be activated automatically at the same time. To also prevent manual simultaneous activation of `ACOn` and `HeatingOn`, we define an exclusion rule between those two contexts. A visualization of the specified context model is shown in Figure 1.

We use context models within EBM in MTK to control simulation variability. MTK allows the definition of discrete and continuous events. We use events to call callback functions that include the functionalities of *Contexts.jl* to activate or deactivate contexts. Based on the defined constraints, other contexts change their activeness, and the simulation model adapts based on these changes. Hereby, adaptation can be implemented by variable (or parameter) reinitialization or structural changes.

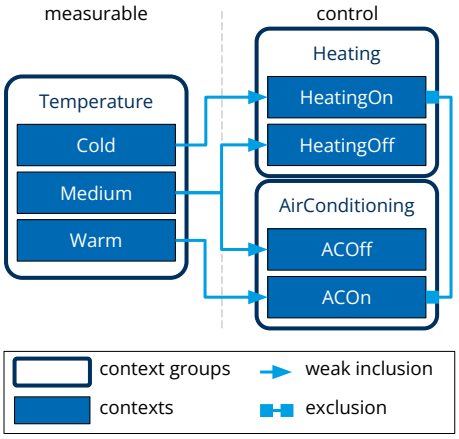


Figure 1. Context model for a heating system consisting of three contextual groups. The implicit alternative constraints between contexts of one group are not separately visualized.

4.2 Context-Oriented Variable Reinitialization

MTK supports variable reinitialization to achieve discrete changes of variable values. We extend this by using the COP features of *Contexts.jl* to control the reinitialization based on changes in the context activeness. Again, we explain this mechanism with the example from above. Therefore, we assume heating can be simulated as on or off by reinitializing a voltage value. We model possible context transitions within one context group as a state machine. For example, the temperature context transitions from `Cold` to `Medium` if $T \geq 15\,°\text{C}$ and from `Medium` to `Cold` if $T \leq 14\,°\text{C}$. The difference between temperature thresholds prevents cyclic event calls. Accordingly, transitions between `Warm` and `Medium` are defined, but transitions between `Warm` and `Cold` are not foreseen. Callback functions in MTK are Julia functions, i.e., Julia's complete expressiveness is available. In Listing 2, a callback function is defined on lines 9 to 26, and computational logic representing the state machine is shown on lines 10 to 24. In line 25, a context-oriented function is called to reinitialize the voltage value if the previously performed context changes influenced the activeness of the heating context. The two context-specific definitions for the function are shown in lines 1 to 7.

In this explanatory example, the voltage values could be reinitialized directly based on temperature values instead of activating the temperature contexts. However, for spaces containing multiple measurable context groups or when activating control contexts also depends on the activeness of other control contexts, separating the state machines for measurable contexts from the activation of control contexts is advantageous. Such an example will be presented in section 5.

4.3 Contextual Views for Structural Changes

A case not natively supported in MTK (or Modelica) is variability due to changing, adding, or deleting variables,

```
1  @context HeatingOn function setV(integ, u)
2    integ.u[u.V] = 12
3  end
4
5  @context HeatingOff function setV(integ, u)
6    integ.u[u.V] = 0
7  end
8
9  function tempChange!(integ, u, p, ctx)
10   if isActive(Warm)
11     if integ.u[u.T] <= 24
12       activateContext(Medium)
13     end
14   elseif isActive(Medium)
15     if integ.u[u.T] <= 14
16       activateContext(Cold)
17     elseif integ.u[u.T] >= 25
18       activateContext(Warm)
19     end
20   elseif isActive(Cold)
21     if integ.u[u.T] >= 15
22       activateContext(Medium)
23     end
24   end
25   @context Heating() setV(integ, u)
26 end
```

Listing 2. Example for a callback function. The computational logic from lines 10 to 24 represents a state machine changing the temperature context based on the current temperature value. The function call on line 25 is a context-oriented call of the function defined on lines 1 to 7, performing a variable reinitialization.

equations, and whole components. We realize the simulation of such VSS by defining distinct differential equation systems that are represented by `ODESystem` objects in MTK. When simulating the whole system, the solving algorithm must switch between those `ODESystem` objects and perform state migration. The algorithm must transfer the end values of all variables to the initial values for the respective variables in the new `ODESystem`. This workflow, realized in Python and not for MTK, was already applied in DySMo (Mehlhase 2014) to simulate VSS. We implement an adapted version of this workflow where contexts decide which model is simulated next.

In our modeling technique, when to switch the model and which model should be simulated next is decided based on the context model, called contextual VSS (CVSS). Therefore, every model variant is assigned to a context or multiple contexts linked with logical operators. Every context must be unambiguously mapped to a model to ensure systems have a well-defined behavior.

To simulate CVSS in MTK without performing the state migration manually, we implemented multiple features. As described in (Rackauckas and Nie 2019), we implement this extension based on Julia's type system and multiple dispatch paradigm.

1. We defined a new problem type for contextual VSS, `CVSSProblem`, as a Julia `struct`. Instead of a single `ODESystem` object, the `struct` contains a dictionary, here called `f`, for the perviously men-

DOI
10.3384/ecp218155

tioned mapping of contexts to `ODESystems`.

2. We specify a new implementation of the `solve` function of *DifferentialEquations.jl* for `CVSSProblem` types with multiple dispatch. This `solve` function checks the keys of `f`, which are the contexts `ODESystems` are mapped to. If a context is active, the respective `ODESystem` will be used for simulation. Therefore, an `ODEProblem` and a respective `Integrator` object are created. The initial values `u0` and simulation start time `t0` are set to either the original start values or the end values of the previous partial solving process. Then, the `ODEProblem` is solved, and its `ODESolution` is saved. If the simulation is restarted, the process is repeated. The simulation ends if the specified ending time is reached, the function `terminate!` is called, or an error occurs. The workflow is visualized in Figure 2
3. We define a new function `restart!`. It calls the `terminate!` function to stop the solving process. If a termination with `restart!` is detected by the `solve` function, the activeness of contexts is checked again, followed by solving the respective `ODESystem`.
4. We define `VSSSolution` as a subtype of `SciMLBase.AbstractODESolution`. The `VSSSolution` contains all the attributes of ODE solution `structs` but also contains a vector of the `ODESolution` objects obtained during the solving. We implement the `getindex` function for the `VSSSolution` objects so that the solutions are obtained as usually done in MTK.

```
1  function tempChange!(integ, u, p, ctx)
2    ACContext = AirConditioning()
3    ## State machine as shown in
4    ## Listing 1, lines 10 to 25
5    if ACContext != AirConditioning()
6      restart!(integ)
7    end
8  end
9
10 f = Dict(ACOn => system_acOn,
11           ACOFF => system_acOff)
12 prob = CVSSProblem(f, [], (0, 60))
13 sol = solve(prob, RadauIIA5())
```

Listing 3. Example for a callback function (lines 1 to 8) and the code to simulate the system (lines 10 to 13). The function `restart!` (line 6) is used to stop the simulation and restart it with a new mode based on the context activeness of the AC context. In lines 10 and 12, the dictionary `f` is defined that maps context to `ODESystem` objects that are used for simulation. With `f`, we define a `CVSSProblem` (line 12) and simulated by solving the `CVSSProblem` (line 13).

We assume the heating example from above is modeled as a VSS system where there is one model for active AC and one model for inactive AC. Accordingly, we create two `ODESystems`, that we call `system_acOn` and `system_acOff`. Both `ODESystems` contain events that call the callback function `tempChange!` as we defined in Listing 2, but we extend it with the model switching logic as shown in Listing 3 on lines 1 to 8. In line 2 in Listing 3, we save the AC context before changing the temperature context. If the AC context was changed due to changes in the temperature context, the `restart!` function is called to perform the model switch. The definition of the context to feature map `f` is shown on lines 10 and 11. Line 12 shows `CVSS` definition and line 13 the solve call, recreating MTK syntax.

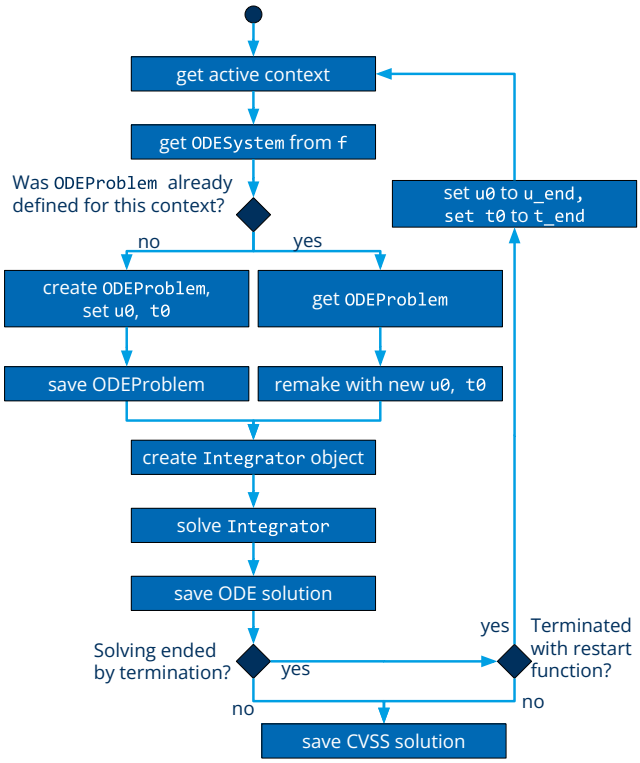


Figure 2. Workflow for solving a CVSS problem. The algorithm checks which contexts mapped to `ODESystems` are active. If this `ODESystem` is solved for the first time, a respective `ODEProblem` is created with the initial values `u0` and start time `t0` and saved. If the `ODESystem` is used another time during solving the `CVSSProblem`, this `ODEProblem` is obtained again and newly initialized with `remake` using the new `u0` and `t0`. An `Integrator` object is created from `ODEProblem` and the `ODESystem` is solved, and the solution is saved. If the simulation was stopped with the `restart!` function, the end values of variables are used to restart the simulation within another context. In the end, the solution is returned as a `VSSSolution` object.

5 Evaluation

We show the feasibility of our modeling process by presenting a simulation of an energy park where hydrogen is produced from renewable energy sources as a case study. After presenting the modeling process and simulation results, performance measurements are provided.

5.1 Case Study: Modeling of an Energy Park

The energy park model consists of wind turbines and photovoltaic (PV) modules for producing electricity, a battery, and an electrolyzer to produce green hydrogen. Also, the park is connected to the electricity grid. Because only green hydrogen should be produced, hydrogen production must be stopped when renewable electricity is unavailable. Also, excess electricity should be stored in a battery to power the electrolyzer with green electricity, even if the weather conditions would not allow it. The energy park should adapt its behavior based on its state, Hence, it is self-adaptive, and context awareness is advantageous.

One of the disadvantages of MTK compared to Modelica is the relatively low availability of libraries. Therefore, most of the needed components have been modeled by us. The models are inspired by Modelica libraries like the Modelica Standard Library, PhotoVoltaics (Brkic et al. 2019), and Wind Power Plants (Eberhart et al. 2015). The electrolyzer model and system design are inspired by (Migoni et al. 2016). A scheme of the system is shown in Figure 3.

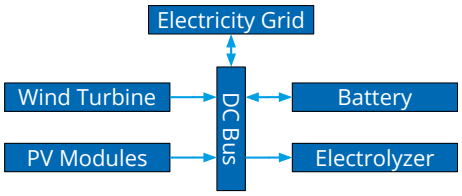


Figure 3. Scheme of the simulation model. Wind turbines and PV modules supply electricity, and an electrolyzer consumes electricity. A battery is used for storing excess electricity. The system can also exchange electricity with the grid.

As this simulation does not aim to give detailed insight into the technical properties of an energy park, the model contains numerous simplifications. We simplified the components themselves compared to the implementations in Modelica. We modeled the system completely in DC with a constant voltage, and converters are neglected. The hydrogen has no physical properties besides mass. Nevertheless, the electricity and hydrogen output are in the correct order of magnitude, and their behavior is modeled reasonably. Because these properties are significant for the presented control via COEBM, the model is suited to show the feasibility of COEBM.

To only produce green hydrogen, two variation points can be controlled: First, the battery can be charged and discharged, and second, the electrolyzer can be put into active and standby mode. Battery charging and discharging is controlled by variable reinitialization, while the electrolyzer operation mode is controlled with a structural change. The active electrolyzer is modeled inspired by (Migoni et al. 2016), while we model standby mode as a resistor representing a constant electrical load. When to switch modes depends on different variable values and the system's state itself.

The variation points are directly translated to contexts. The control contexts `BatteryCharging`, `BatteryDischarging`, and `BatteryIdle` form the context group `BatteryChargingMode`. The contexts `ElectrolyzerActive` and `ElectrolyzerStandby` form the context group `ElectrolyzerOperation`. The measurable contexts represent three properties. First, the power that is fed into or drawn from the grid. We define 4 contexts, `GridOutput` when power is drawn from the grid and `GridInputLow`, `GridInputHigh`, and `GridInputHighWithElectrolyzer`, representing different quantities of power supply to the grid. `GridInputHighWithElectrolyzer` got its name because it can only be reached if the electrolyzer is operated with maximal power. The second context group represents the state of charge (SoC) of the battery containing the contexts `BatterySoCMin`, `BatterySoCMedium`, and `BatterySoCMax`. And third, we define two contexts `ElectrolyzerMaxPower` and `ElectrolyzerMinPower` that are activated based on the voltage with which the electrolyzer is operated. Note that those contexts are not part of a context group because they can be deactivated simultaneously. The context model is visualized in Figure 4.

The lower part of Figure 4 represents the weak inclusion constraints, which implement the control logic. We define seven constraints. Every constraint realizes a specific control step. If the electrolyzer is not operated with maximal power and the battery is charging, charging is stopped (C1). If the electrolyzer is not operated with maximal power and the battery is in idle mode and not empty, the battery gets discharged (C2). Those constraints increase hydrogen production. If the battery discharges but becomes empty, it is set to idle mode (C3). Respectively, if the battery charges but becomes fully charged, it is set to idle mode (C4). If the electrolyzer is active and at minimum power, but grid input is low or we have grid output, the electrolyzer is set to standby mode (C5). This constraint prevents producing hydrogen from grid electricity. If the electrolyzer is in standby mode and the grid input is high, the electrolyzer is set to active mode (C6). If the electrolyzer is operated with maximal power and a certain amount of power is supplied to the grid, the battery is charged if it is not already fully charged (C7).

We define events and callback functions to activate measurable contexts during simulation time. For each of the two context groups `GridState` and `BatterySoC`, we define one callback similarly designed as the one shown in Listing 2. Again, we implement a state machine to control the activation of measurable contexts, and constraints imply the activation of control contexts. We use the `restart!` function and context-oriented functions for variability control. We also define two callback functions for each context: `ElectrolyzerMaxPower` and `ElectrolyzerMinPower`. One function activates the context, and one deactivates it. One of those callback

DOI
10.3384/ecp218155

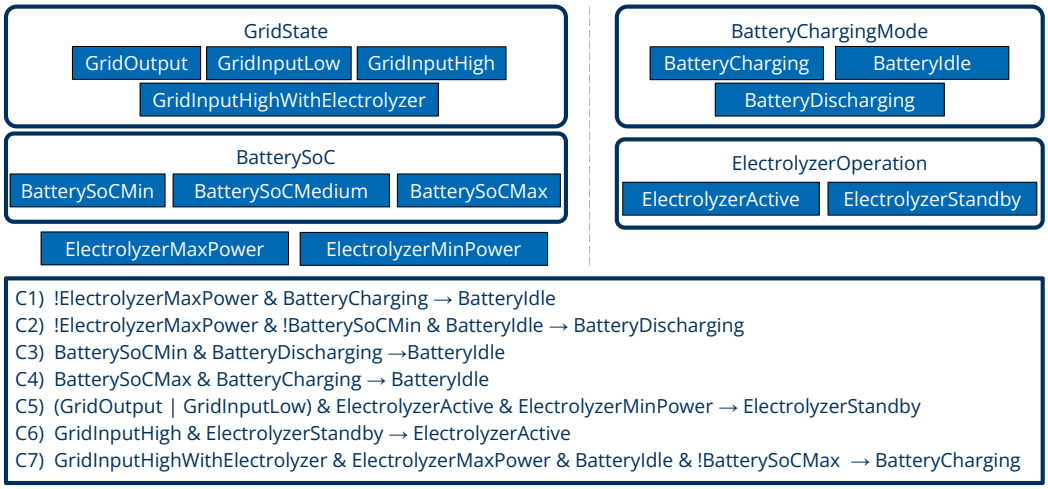


Figure 4. Context model of the energy park simulation. It defines a total of 14 contexts and 4 context groups. The grid state, battery SoC, and electrolyzer power are used for measurable contexts. The battery's charging mode and electrolyzer activeness are implemented as control contexts. The seven constraints that control the system are shown in the lower part of the scheme.

functions is shown in Listing 4, and the rest can be viewed in the code example in the Git repository[1]. The events that call these functions are also shown in the repository. The call on lines 5 and 6 in Listing 4 changes the battery's activation mode. Here, the advantage of using COP is noticeable. Without COP, if-then-else conditions would be needed to check the wanted operation mode. This would result in 7 lines instead of the one presented. In line 8, the restart function is called. The `solve` function automatically chooses the correct next simulation model. It is not needed to specify explicitly to which model to switch.

```
1  function electrolyzerMin!(integ, u, p, ctx)
2    oldContext = ElectrolyzerOperation()
3    if !(isActive(ElectrolyzerMinPower))
4      activateContext(ElectrolyzerMinPower)
5      @context BatteryChargingMode() (
6              batteryMode(integ, u))
7      if oldContext != ElectrolyzerOperation()
8        restart!(integ)
9      end
10   end
11 end
```

Listing 4. Example of a callback function that checks if the context `ElectrolyzerMinPower` must be activated. If the control contexts change after that, variable reinitialization and model switches are performed.

We use solar irradiation data from (Krähenmann et al. 2016b) and wind speed data from (Krähenmann et al. 2016a) as input data. We simulate the system for 1000 h. Figure 5a shows the power of the wind turbines, PV, electrolyzer, battery, and the power exchange with the grid. Positive power values indicate power consumption, while negative values indicate power supply. For the grid, positive values indicate power fed into, and negative values indicate power drawn from the grid.

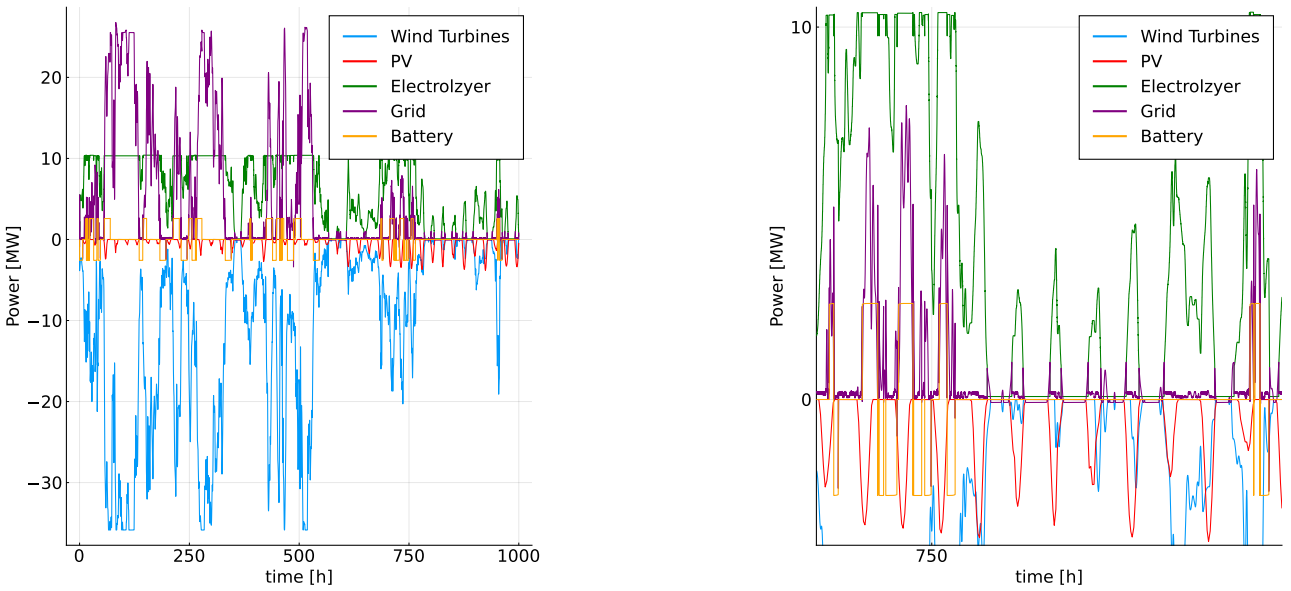
[1] github.com/cgutsche/ContextualEBM.jl

The effects of context control are better visible in Figure 5b. We achieved control of the electrolyzer and battery by activating measurable contexts. The positive values of the battery power curve show that the battery is only charged if the electrolyzer is operated with maximal power. The negative values show that the battery is discharged if the electrolyzer can not be operated with maximum power because the power generated by renewable energy systems is insufficient. The results show that for multiple days, electricity is barely produced by wind power plants. Then, only the PV provides electricity during the day to operate the electrolyzer, while the electrolyzer is set to standby at night. Therefore, by properly defining the context model, we achieved the goals of producing hydrogen only with energy supplied by renewables and increasing the hydrogen output with battery control. In total, the model switched 32 times during the simulation. The electrolyzer was active during 826.3 h from a total 1000 h simulated.

5.2 Performance

One advantage of simulating VSS is to switch between more and less detailed models depending on the needed accuracy under different input conditions. Also, one possible use case for CVSS simulation is to support the operation of CPS where simulation time becomes crucial. However, computing the context control logic, callback functions in events, and the system switches are expected to have a computational overhead. Therefore, we provide performance measurements to evaluate its impact.

We compare 7 different scenarios. First, the CVSS simulation of the energy park as it was presented (*CVSS*). Second, the `restart!` functions are removed from the callbacks, and the system is simulated as an `ODEProblem` (*Active* and *Standby*). Hence, the events are still triggered, and contexts are activated, but the simulation model is not switched. Third, the events are kept, but the callback functions contain no action, i.e., the context activeness is un-



(a) Complete Simulation results.

(b) Detailed view.

Figure 5. Simulation results of the energy park model.

changed (*ActiveEmptyCB* and *StandbyEmptyCB*). Fourth, the events are entirely removed from the simulation models (*ActiveNoCB* and *StandbyNoCB*). The measurements are performed 10 times, and the simulations not based on `CVSSProblem` are performed for the model with the electrolyzer in active and standby mode. The results are shown in Figure 6

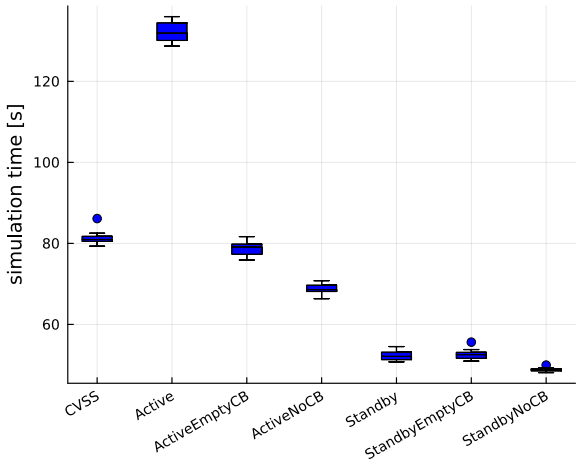


Figure 6. Boxplot that shows the simulation time for seven different simulations. Every model was simulated 10 times.

The comparison of the models that do not contain any events shows that the active electrolyzer model increases the simulation time, which is explainable by the model's increased complexity. Adding empty callbacks also increases simulation time. This effect is stronger for the model with an active electrolyzer, which is explainable by a larger number of events. Adding the context control logic to the callbacks significantly increases the simulation time for the model with an active electrolyzer. Using profiling showed that computing the constraints between contexts is the main reason for this increase.

To estimate the additional computation time of the COEBM simulation, we compare the mean simulation time of the CVSS simulation with a weighted sum of the mean simulation time for the active and standby models with empty callbacks that do not activate contexts. The weights are calculated as $826.3\,\mathrm{h}/1000\,\mathrm{h}$ for the active model and $1 - 826.3\,\mathrm{h}/1000\,\mathrm{h}$ for the standby model, so that they represent the share of the respective models in the total simulation time. The mean CVSS simulation took 81.4 s, and the weighted sum of the active and standby simulation results in 74.2 s. Hence, the increase in simulation time is approximately 10 %.

6 Discussion

The presented approach of COEBM allows simulating VSS that can not be simulated in standard Modelica or MTK without external tools or manually concatenating simulations and their results. COEBM enables specifying control mechanisms based on contextual modeling with concise syntax provided by *Contexts.jl*. Instead of manually (de-)activating components or reinitializing variables based only on if-then-else conditions, constraints between contexts can be used to control the system, and context-adaptive behavior is achieved. Defining constraints within the context model prevents reaching conflicting states. As discussed in (Gutsche et al. 2024), context control in *Contexts.jl* provides the basis for model-checking.

The case study shows that using the context model sim-

DOI 10.3384/ecp218155

ulation control is feasible and advantageous. COP reduces the code complexity. Instead of if-then-else-cascades, only a context-oriented function call is used to check how to operate the battery. The remaining if-then-else conditions are comprehensible as they represent state machines and contain Boolean expressions that depend on one or a few variables. The goals of the variability were achieved.

Contexts discretize continuous variables and, therefore, are activated based on threshold values. Therefore, the choice of threshold values influences the control. In the case study, the threshold values were defined meaningfully. However, changing them can, e.g., change when the electrolyzer gets activated and influences the amount of produced hydrogen. Future research could investigate obtaining optimal values via optimization.

The performance measurements show that computing context control is time-consuming. However, *Contexts.jl* was not optimized for performance. Julia gives developers many tools to improve performance. The context control computation can be optimized. For example, the library uses normal Julia arrays for computing constraints, but static arrays can improve the performance [2].

Context activeness must be initialized before the simulation. If initial activeness does not fit the initial variable values, the control mechanisms could work faulty. MTK allows the calculation of initial variable values, which can be used to initialize the context activeness correctly.

The presented case study has several simplifications. Some simplifications are only relevant for getting technical insights and could be resolved by adding or replacing components in the model. However, other simplifications would also influence the control logic. The model does not include an intermediate state between active electrolyzer and standby, resulting in discrete jumps within the simulation. Also, shutdown is not implemented. Adding models for deactivated and ramp-up states would create additional models and contexts and, therefore, more complexity, but the modeling technique is still applicable.

While we focused on modeling variable behavior, another interesting use case for variability in simulation is the use of surrogate models. If the needed accuracy allows surrogates, contexts can switch between more and less detailed models or data-driven components. Using neural networks within EBM models is supported by MTK.

7 Conclusion and Outlook

We presented how COEBM enables simulating equation-based models containing variability. Therefore, we described a modeling technique based on contexts. Contextual modeling is used to abstract the simulated world and specify control mechanisms. Constraints in the contextual model prevent reaching conflicting states. This is advantageous when modeling self-adaptive systems, e.g., cyber-physical systems, with complex state spaces.

We presented a case study of an energy park for hydrogen production, showing the feasibility of the modeling process in more complex scenarios. We achieve VSS simulation by switching between different simulation models and performing state migration. Performance measurements show that the context control negatively impacts simulation time but can be improved by changing the *Contexts.jl* implementation of computing context control.

We are currently working on implementing additional modeling tools in Julia to simplify the generation of model variants using a single underlying model (Atkinson, Stoll, and Bostan 2010; Seifert 2011). This will improve code reuse and maintainability. Additionally, concise syntax for defining variability not located in the simulation model's top-level would improve modeling.

Another interesting topic for future research is investigating the combination of the presented approach and *OM.jl* to achieve COEBM in Modelica. Additionally, MTK provides an experimental feature to use Functional Mock-up Units (FMUs) within MTK models. This could pave the way for coupling FMUs and Modelica models to COEBM.

Acknowledgements

The authors would like to thank the Boysen–TU Dresden–Research Training Group for the financial support that has made this contribution possible. The Research Training Group is co-financed by the Friedrich and Elisabeth Boysen Foundation and the TU Dresden. The authors would also like to thank the German Research Foundation (DFG) for financially supporting the presented research via SFB/TRR 339 (Project ID 453596084).

Disclaimer

To improve the use of the English language, the authors used the AI-based tools DeepL and Grammarly.

References

Abowd, Gregory D. et al. (1999). "Towards a Better Understanding of Context and Context-Awareness". In: *Handheld and Ubiquitous Computing*. Ed. by Hans-W. Gellersen. Berlin, Heidelberg: Springer. ISBN: 978-3-540-48157-7.

Atkinson, Colin, Dietmar Stoll, and Philipp Bostan (2010). "Orthographic Software Modeling: A Practical Approach to View-Based Development". In: *Evaluation of Novel Approaches to Software Engineering*. Ed. by Leszek A. Maciaszek, César González-Pérez, and Stefan Jablonski. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 206–219. ISBN: 978-3-642-14819-4.

Benveniste, Albert, Benoit Caillaud, and Mathias Malandain (2020). "The mathematical foundations of physical systems modeling languages". In: *Annual Reviews in Control* 50, pp. 72–118. ISSN: 1367-5788. DOI: https://doi.org/10.1016/j.arcontrol.2020.08.001. URL: https://www.sciencedirect.com/science/article/pii/S1367578820300547.

Bezanson, Jeff et al. (2015). *Julia: A Fresh Approach to Numerical Computing*. arXiv: 1411.1607 [cs.MS].

[2] github.com/JuliaArrays/StaticArrays.jl

Brkic, Jovan et al. (2019-03). "Open Source PhotoVoltaics Library for Systemic Investigations". In: *Proceedings of the 13th International Modelica Conference, Regensburg, Germany, March 4–6, 2019*. Linköping Electronic Conference Proceedings. DOI: 10.3384/ecp1915741.

Broman, David and Jeremy G Siek (2012). "Modelyze: a gradually typed host language for embedding equation-based modeling languages". In: *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2012-173*.

Cardozo, Nicolas et al. (2012). *Context Petri Nets: Definition and Manipulation*. English. Vrije Universiteit Brussel.

Eberhart, Philip et al. (2015-09). "Open Source Library for the Simulation ofWind Power Plants". In: *Proceedings of the 11th International Modelica Conference, Versailles, France, September 21-23, 2015*. Linköping Electronic Conference Proceedings, pp. 929–936. DOI: 10.3384/ecp15118929.

Elmqvist, Hilding, Andrea Neumayr, and Martin Otter (2018). "Modia-dynamic modeling and simulation with julia". In: Juliacon, University College London, UK.

Esperon, Daniel Gomez, Alexandra Mehlhase, and Thomas Karbe (2015). "Appending variable-structure to modelica models (WIP)". In: *Proceedings of the Conference on Summer Computer Simulation*, pp. 1–6.

Fritzson, Peter and Vadim Engelson (1998). "Modelica—A unified object-oriented language for system modeling and simulation". In: *ECOOP'98—Object-Oriented Programming: 12th European Conference Brussels, Belgium, July 20–24, 1998 Proceedings 12*. Springer, pp. 67–90.

Giorgidze, George (2012). "First-class models: On a noncausal language for higher-order and structurally dynamic modelling and simulation". PhD thesis. University of Nottingham.

Gómez Esperón, Daniel (2017). "Strukturvariabilität in der objektorientierten Modellierung und Simulation durch Generierung von Varianten". PhD thesis. Technische Universitaet Berlin (Germany). DOI: 10.14279/depositonce-5805.

Gutsche, Christian et al. (2024). "Context-Oriented Programming and Modeling in Julia with Context Petri Nets". In: *2024 50th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pp. 1–9. DOI: 10.1109/SEAA64295.2024.00011.

Hirschfeld, Robert, Pascal Costanza, and Oscar Nierstrasz (2008). "Context-oriented Programming". In: *Journal of Object Technology*.

Keays, Roger and Andry Rakotonirainy (2003). "Context-oriented programming". In: *Proceedings of the 3rd ACM International Workshop on Data Engineering for Wireless and Mobile Access*. MobiDe '03. San Diego, CA, USA: Association for Computing Machinery, pp. 9–16. ISBN: 1581137672. DOI: 10.1145/940923.940926. URL: https://doi.org/10.1145/940923.940926.

Krähenmann, S. et al. (2016a). *Stündliche Raster der Globalstrahlung für Deutschland (Projekt TRY-Weiterentwicklung)*. DWD Climate Data Center (CDC). DOI: DOI : 10 . 5676 / DWD_CDC/TRY_Basis_v001.

Krähenmann, S. et al. (2016b). *Stündliche Raster der Windgeschwindigkeit für Deutschland (Projekt TRY-Weiterentwicklung)*. DWD Climate Data Center (CDC). DOI: DOI:10.5676/DWD_CDC/TRY_Basis_v001.

Lee, Jay, Behrad Bagheri, and Hung-An Kao (2015). "A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems". In: *Manufacturing Letters* 3, pp. 18–23. ISSN: 2213-8463. DOI: https://doi.org/10.1016/j.mfglet.2014.12.001.

Lubin, Miles et al. (2023). "JuMP 1.0: Recent improvements to a modeling language for mathematical optimization". In: *Mathematical Programming Computation*. DOI: 10 . 1007 / s12532-023-00239-3.

Ma, Yingbo et al. (2021). *ModelingToolkit: A Composable Graph Transformation System For Equation-Based Modeling*. arXiv: 2103.05244 [cs.MS].

Mehlhase, Alexandra (2014). "A Python framework to create and simulate models with variable structure in common simulation environments". In: *Mathematical and Computer Modelling of Dynamical Systems* 20.6, pp. 566–583. DOI: 10 . 1080/13873954.2013.861854.

Migoni, G. et al. (2016). "Efficient simulation of Hybrid Renewable Energy Systems". In: *International Journal of Hydrogen Energy* 41.32, pp. 13934–13949. ISSN: 0360-3199. DOI: https://doi.org/10.1016/j.ijhydene.2016.06.019.

Muschevici, Radu, Dave Clarke, and José Proença (2010-01). "Feature Petri Nets." In: *Proceedings of the 14th International Software Product Line Conference (SPLC 2010)* 2, pp. 99–106.

Neumayr, Andrea and Martin Otter (2023). "Modelling and Simulation of Physical Systems with Dynamically Changing Degrees of Freedom". In: *Electronics* 12.3. ISSN: 2079-9292. DOI: 10.3390/electronics12030500. URL: https://www.mdpi.com/2079-9292/12/3/500.

Rackauckas, Christopher (2021). *Modelingtoolkit, modelica, and modia: The composable modeling future in julia*. The Winnower. DOI: 10.15200/winn.162133.39054.

Rackauckas, Christopher and Qing Nie (2017). "DifferentialEquations.jl–a performant and feature-rich ecosystem for solving differential equations in Julia". In: *Journal of Open Research Software* 5.1.

Rackauckas, Christopher and Qing Nie (2019). "Confederated modular differential equation APIs for accelerated algorithm development and benchmarking". In: *Advances in Engineering Software* 132, pp. 1–6. ISSN: 0965-9978. DOI: https://doi.org/10.1016/j.advengsoft.2019.03.009. URL: https://www.sciencedirect.com/science/article/pii/S0965997818310251.

Seifert, Mirko (2011). *Designing Round-Trip Systems by Change Propagation and Model Partitioning*. Technische Universität Dresden. URL: https://nbn-resolving.org/urn:nbn:de:bsz:14-qucosa-71098.

Stüber, Moritz (2017). "Simulating a Variable-structure Model of an Electric Vehicle for Battery Life Estimation Using Modelica/Dymola and Python." In: *Proceedings of the 12th international Modelica conference*, pp. 132–031.

The Julia Project (2023). *Julia 1.10 Documentation*. https://docs.julialang.org/en/v1/. Accessed: 2024-01-25.

Tinnerholm, John, Adrian Pop, and Martin Sjölund (2022). "A Modular, Extensible, and Modelica-Standard-Compliant OpenModelica Compiler Framework in Julia Supporting Structural Variability". In: *Electronics* 11.11. ISSN: 2079-9292. DOI: 10.3390/electronics11111772.

Utkin, Vadim (1977). "Variable structure systems with sliding modes". In: *IEEE Transactions on Automatic control* 22.2, pp. 212–222.

Wang, Zizhe et al. (2025-01). "Proposal for A Context-oriented Modelica Contributing to Variable Structure Systems". In: *Modelica Conferences*, pp. 53–62. DOI: 10.3384/ecp20753.

Zimmer, Dirk (2010). *Equation-based modeling of variable-structure systems*. Swiss Federal Institute of Technology, Zürich. URL: https://people.inf.ethz.ch/fcellier/PhD/zimmer_phd.pdf.

Context-Oriented Modelica for Advanced Variability Management

Zizhe Wang[1,2] Christian Gutsche[1,2] Uwe Aßmann[2]

[1]Boysen-TU Dresden-Research Training Group, Dresden, Germany
[2]Software Technology Group, Technische Universität Dresden, Germany
zizhe.wang, christian.gutsche, uwe.assmann@tu-dresden.de

Abstract

Context-aware systems are crucial in modern cyber-physical systems (CPS), enabling dynamic adaptation to changing conditions. Many such systems involve complex variability. Modelica, a leading equation-based language for modeling and simulating physical systems, struggles to manage this complexity. As variability management becomes more complex, traditional Modelica constructs such as conditional statements, state machines, and state graphs become increasingly inadequate and difficult to maintain. This work introduces a **context-oriented modeling paradigm for Modelica** by integrating Petri Net-based context control with FMI-based Variable Structure Systems (VSS). Specifically, we present Context Petri Nets (CoPN) as a formal mechanism for representing and managing contextual dependencies. By combining CoPN with VSS, our approach enables advanced variability management, supporting the modeling and simulation of complex, context-aware CPS.

Keywords: Modelica, variable structure systems, FMI, Context Petri Nets, context-aware systems, variability management, context-oriented modeling and simulation

1 Introduction

Context-aware systems exist everywhere in daily life. According to Dey and Abowd 2000, the context refers to conditions that influence a system's behavior or state. A typical example is the smartphone, where different operation modes (e.g., `do not disturb`, `power saving`, `gaming`) dynamically adjust software and hardware configurations based on active contexts (e.g., battery level). More importantly, these modes often have contextual relationships. For example, when the battery level drops below 20%, `power saving` mode activates, reducing the screen refresh rate and limiting CPU/GPU performance. If `gaming` mode attempts to activate at that moment, it fails, which demonstrates an **exclusion** relationship between the two modes. Another example is the iPhone's crash detection feature. When a car crash occurs, the iPhone detects the event and activates the `emergency` context, automatically dialing emergency services. In this case, `crash` and `emergency` have an **inclusion** relationship, as one triggers the other. More specifically, this represents a **weak inclusion** relationship, where `crash` triggers `emergency`, but not vice versa.

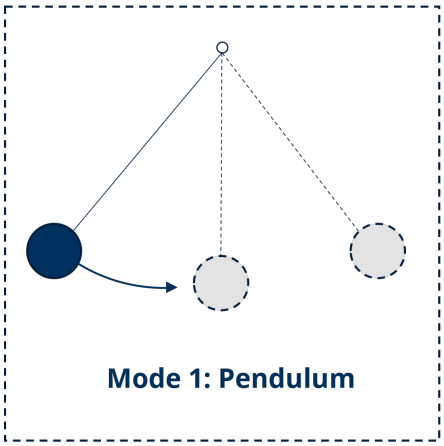
Modelica, a state-of-the-art equation-based modeling language, along with its various tools and libraries, is widely used in industry for modeling and simulating multi-domain and multi-physical systems. For simple context expression and variability management, conditional statements (e.g., `if-else` and `when` clauses) are sufficient. However, for more advanced context modeling and management, especially when dealing with complex contextual relationships, traditional conditional statements become inadequate. They not only lack expressiveness but also lead to unreadable and hard-to-maintain code. Addressing advanced contextual relationships in Modelica remains an open research challenge. This work addresses this challenge by introducing **context-oriented Modelica**, an approach inspired by context-oriented programming (Hirschfeld, Costanza, and Nierstrasz 2008), tailored to fit the Modelica modeling paradigm.

The structure of this work is as follows: Section 2 introduces the foundational concepts. Section 3 explores a novel and efficient solution for enabling VSS through FMI. Section 4 focuses on Context Petri Nets (CoPN) and their integration into Modelica for advanced variability management. Section 5 brings together CoPN and VSS into a more generalized approach. A proof of concept demonstrates how this enables the development of context-aware systems. Finally, Section 6 concludes the work and highlights future directions.

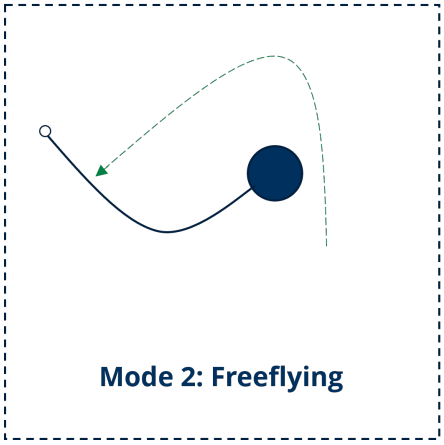
2 Background

2.1 Variable Structure Systems in Modelica

One significant challenge for realizing advanced context relationships is Variable Structure Systems (VSS), which was first introduced by Utkin 1977. VSS consist of various subsystems with proper switching logic for switching between these subsystems. In the context of Modelica, VSS mean to switch between different submodels and equation systems based on conditions, as Figure 1 shows. The challenge is that, because of Modelica's static nature (everything must be defined before simulation), the switching of submodels and equation systems is limited. Wang et al. 2024 details the conditions under which VSS works and where it fails. In a summary, when equation systems have different variables and/or differential indexes, the structural transition will fail.




```
model Pendulum
  Real phi(start=0); // Pendulum angle
  Real dphi(start=-2); // Angular velocity
  parameter Real m = 1; // Bob mass
  parameter Real g = 9.81; // Gravity
  parameter Real L = 2; // Pendulum length
  Real x, y, dx, dy, F;
equation
  x = L * sin(phi);
  y = -L * cos(phi);
  dx = der(x); dy = der(y);
  dphi = der(phi);
  der(dphi) = -g/L * sin(phi);
  F = m * (g * cos(phi) + L * dphi^2);
end Pendulum;
```




```
model Freeflying
  Real x;
  Real y;
  Real vx;
  Real vy;
  parameter Real m = 1;
  parameter Real g = 9.81;
  parameter Real L = 2;
equation
  vx = der(x);
  vy = der(y);
  m * der(vx) = 0;
  m * der(vy) = -m * g;
  r = sqrt(x^2 + y^2);
end Freeflying;
```

Figure 1. Two modes with their corresponding equation systems. Modelica cannot handle the transition, thus, at the moment of switching from mode 1 to mode 2, the simulation fails.

2.2 Variability Management in Modelica

Modelica supports several constructs for managing variability, the most basic being conditional statements like **`if-else`** and **`when`**. While suitable for simple systems, these constructs quickly become unwieldy and error-prone in complex systems, making the logic hard to maintain and debug. Modelica also offers **state machines** (as defined in the *Modelica Language Specification*) and **state graphs** (`Modelica.StateGraph`, as defined in the *Modelica Standard Library*). These introduce hierarchy and parallelism, improving clarity compared to raw conditionals. However, their reliance on directed transitions limits flexibility, since state activation typically follows predefined, sequential paths, making it difficult to handle highly dynamic, condition-driven context changes. For more advanced variability management in complex systems, a more flexible and expressive mechanism is needed. Petri Nets offer a powerful alternative. Their token-based, condition-driven structure allows non-directional context activation, supporting complex, concurrent, and non-sequential behaviors without rigid state progression. This enables precise control over context interdependencies while maintaining system reliability through strong formal analysis, which is essential for robust orchestration of complex, dynamic behaviors.

2.3 Petri Nets

Petri 1962 introduced Petri Nets as a formal modeling approach in his PhD dissertation *Kommunikation mit Automaten* (English: Communication with Automata). A Petri Net consists of four fundamental components: **places** (P), which denote system states and are typically represented as circles; **transitions** (T), which represent events or actions that drive state changes and are depicted as rectangles or bars; **tokens**, shown as dots inside places, which indicate the current system state and move between places; and **arcs**, drawn as arrows, which connect places and transitions, directing token flow. The token distribution across all places is called a marking and represents the current state of the Petri Net. The system's behavior is determined by firing rules, where a transition is enabled if all its input places contain the necessary number of tokens. Upon firing, the transition consumes tokens from its input places and generates tokens in its output places, updating the system state accordingly. Their later-introduced graphical nature also makes them intuitive for visualizing complex dynamic behaviors in distributed or reactive systems. Figure 2 shows how a Petri Net works. This Petri Net consists of three places, P_1, P_2, and P_3, and one transition T. The diagram shows the system both **before** and **after** firing.

DOI
10.3384/ecp218165

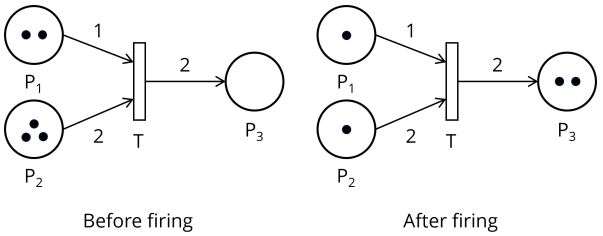


Figure 2. Example of a Petri Net

2.4 Functional Mock-Up Interface

Functional Mock-up Interface (FMI) (Blochwitz et al. 2011) is an open standard for the exchange and co-simulation of dynamic models across different modeling languages and platforms. It defines a container and a set of APIs that allow models, packaged as Functional Mock-up Units (FMUs), to be shared and executed consistently in various simulation environments. FMI is widely adopted in cyber-physical systems engineering for enabling interoperability. FMI 1.0, 2.0, and 3.0 were released in 2010, 2014, and 2022, respectively. FMI supports three FMU types: Model Exchange, Co-Simulation, and the newly introduced Scheduled Execution in FMI 3.0.

3 Advancing VSS with FMI

3.1 State of the Art

Many approaches have been developed to realize VSS in equation-based modeling environments. Based on timeline and design concepts, these can be broadly categorized into four groups. The **first group** includes early works based on new languages/frameworks, such as Mosilab (Nytsch-Geusen et al. 2005), Sol (Zimmer 2010), Hydra (Giorgidze 2012), and Modelyze (Broman and Siek 2012). These support true structural changes and enable VSS modeling but are incompatible with the Modelica ecosystem and cannot reuse existing Modelica libraries. The **second group** focuses on integrating VSS support directly into Dymola, as proposed by Elmqvist, Mattsson, and Otter 2014 and Mattsson, Otter, and Elmquvist 2015. These extensions were only validated with simple examples and have not been included in stable Dymola releases due to limited robustness and generality. The **third group** involves frameworks that orchestrate VSS simulations externally, such as DySMo (Mehlhase 2015), MoVaSim (Gómez Esperón 2017), and PyVSM (Stüber 2017). These do not support true structural changes, and thus only simulate VSS. However, since the models remain in Modelica, modelers benefit from the Modelica ecosystem. The **fourth group** features recent Julia-based approaches, such as Modia (Elmqvist, Neumayr, and Otter 2018), OM.jl (Tinnerholm et al. 2020), and ModelingToolkit (Ma et al. 2021). Julia's dynamic type system and multiple dispatch enhance VSS support. However, like the first group, they require porting Modelica libraries and generally lack the graphical interfaces.

The goal is to remain within the Modelica ecosystem. In DySMo, conditions must be defined within each model. If conditions are managed externally, DySMo can only evaluate them after each time step. For example, if the time step is 1 second and a condition is met at 0.5 seconds, it will still be evaluated at 1 second, not immediately at 0.5 seconds when the condition occurs. This limitation makes DySMo unsuitable for our work. MoVaSim, while more flexible, is implemented on the JModelica compiler. Since JModelica has not been maintained since late 2019, MoVaSim is effectively obsolete unless it is ported to another Modelica compiler. PyVSM lacks sufficient public documentation, with only a single mention available, making detailed evaluation infeasible. To overcome these limitations, we propose an approach for realizing VSS using FMI. This approach resolves DySMo's conceptual issues, improves performance, and leverages FMI's standardized APIs for simulation and management of VSS.

3.2 Concept

The concept, inspired by DySMo, coordinates different simulations by detecting transition points, recording necessary variable values, and using them to initialize the subsequent modes. Simulation results from different submodels are then combined. Using FMI streamlines this process, with Python libraries like FMPy (Sommer 2017) and PyFMI (Andersson, Åkesson, and Führer 2016) providing efficient APIs that avoid costly data serialization and external tool calls, significantly speeding up execution.

This concept was implemented with examples such as the *Pendulum-Freeflying* example mentioned earlier in Figure 1. The simulation result is shown in Figure 3. The first mode (*Pendulum*) will stop when the string loses force on the bob. The system detects this, records the stop time and variable values, and then starts the second mode (*Freeflying*) at the stop time of the first mode, initializing the model with the recorded values. The second mode stops when the string holds the bob again. Other examples can be found at GitHub | FMUVSS.

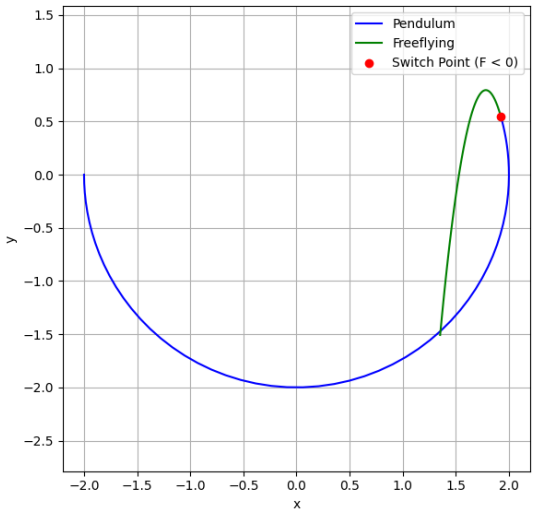


Figure 3. Trajectory of the *Pendulum-Freeflying* example

3.3 The Framework FMUVSS

Based on this concept, the FMUVSS framework was developed to automate the process and provide a user-friendly interface, eliminating the need for manual prototyping. It streamlines the simulation of VSS and supports both FMI 2.0 and 3.0 by leveraging their low-level APIs for finer control over the simulation process (e.g., retrieving specific variable values during simulation). Stop conditions can be defined directly without the need to modify the original models, ensuring a clean separation of concerns: model design remains intact while variability management and switching logic are handled externally. Compared to FMI 2.0, FMI 3.0 offers significant advantages by introducing **event mode** and **early return** in Co-Simulation FMUs, enabling the master algorithm to respond to zero-crossings or mode switches precisely when they occur. In contrast, FMI 2.0 only evaluates events after each simulation step, preventing events from being triggered at their exact time of occurrence. In critical systems, such timing inaccuracies can lead to unstable or incorrect simulation outcomes.

Listing 1. Example of mode configuration in FMUVSS

```
'SlidingBall': {
    'fmu_path': './Ball.fmu',
    'monitored_vars': ['y'],
    'outputs': ['x', 'y', 'vx', 'vy'],
    'stop_condition': lambda vars:
        vars['y'] < 10,
    'transition_mapping': {
        'FlyingBall': {
            'x' : 'x', 'y' : 'h',
            'vx' : 'vx', 'vy' : 'vy'}
            },
    'next_mode': 'FlyingBall'},

'FlyingBall': {
    'fmu_path': './FlyingBall.fmu',
    'monitored_vars': ['h', 'r'],
    'outputs': ['x', 'h', 'vx', 'vy'],
    'stop_condition': lambda vars:
        vars['h'] < vars['r'],
    'transition_mapping': {
        'BouncingBall': {
            'x' : 'x', 'h' : 'h',
            'vx' : 'vx', 'vy' : 'v'}
            },
    'next_mode': 'BouncingBall'},

'BouncingBall': {
    'fmu_path': './ContactBall.fmu',
    'monitored_vars': ['mass.s', 'r'],
    'outputs': ['x', 'h', 'vx', 'v'],
    'stop_condition': lambda vars:
        vars['mass.s'] > vars['r'],
    'transition_mapping': {
        'FlyingBall': {
            'x' : 'x', 'h' : 'h',
            'vx' : 'vx', 'v' : 'vy'}
            },
    'next_mode': 'FlyingBall'}
```

To illustrate the framework in practice, Listing 1 shows part of the configuration interface used to define modes for a *SlidingBall–FlyingBall–BouncingBall* system: The ball begins by sliding down an inclined surface in the `SlidingBall` mode. After reaching a certain height, it leaves the surface and transitions into the `FlyingBall` mode. When it hits the ground, it switches to the `BouncingBall` mode, where damping reduces its bounce height. The system continues alternating between flying and bouncing until the simulation ends.

Compared to DySMo, FMUVSS eliminates the need for modifications to the Modelica models. In DySMo, stop conditions and a `transitionID` must be inserted into the original models to trigger the transition logic. The `transitionID` is initialized to 0, and when the transition condition is met, it changes to 1, signaling the framework to stop the current simulation and switch to the next mode. In contrast, FMUVSS allows developers to export unmodified Modelica models as FMUs. Transition conditions are defined externally and managed centrally, enabling advanced variability handling such as nested conditions and inter-condition relationships.

Experiments demonstrate that FMUVSS runs significantly faster than DySMo, as shown in Table 1. This comparison was conducted using the same models, the DASSL solver proposed by Petzold 1982, a tolerance of `1e-4`, and identical hardware for three VSS models: the Pendulum and SlidingBall models, as well as a clutch model from Benveniste et al. 2022, where two clutches engage and disengage. Compilation time for compiling each model with DySMo was excluded.

Table 1. Performance comparison: FMUVSS vs DySMo

VSS Model	Framework	Time (s)
Pendulum	FMUVSS	0.53
	DySMo	18.69
SlidingBall	FMUVSS	3.34
	DySMo	81.08
Clutch	FMUVSS	2.43
	DySMo	339.4

Profiling with Python tools, such as `cProfile`, reveals that DySMo's slow performance is due to its heavy reliance on Python interpretation, text-based file parsing, and high-overhead interprocess communication (IPC) for executing Modelica simulations. Additionally, DySMo uses dynamic list growth, leading to memory allocation inefficiencies. In contrast, FMUVSS achieves superior performance by using direct binary access through the FMI standard, which avoids the overhead of text parsing and IPC. By leveraging preallocated arrays and zero-copy buffers, FMUVSS minimizes both Python overhead and memory allocation delays, resulting in faster and more efficient execution. More details are available at: GitHub | FMUVSS | Performance Evaluation.

DOI
10.3384/ecp218165

4 Context-Oriented Petri Nets

4.1 Petri Nets for Context-Aware Systems

Gutsche, Wang, et al. 2023 presents using Petri Nets as a control mechanism for state transitions in Modelica. However, if the contexts become more complicated, e.g., if some of the contexts are dependent, such as activating one context (de)activates the other, or activating one requires the other to be active. Using original Petri Nets is not easy for describing these dependencies. Thus a more powerful form of Petri Nets is needed. Petri Nets have many extensions and they have also been extended in different forms to model context-aware systems. Gutsche, Prokopets, et al. 2024 summarized four main extensions in this field: Contextual Petri Nets from Baldan, Corradini, and Montanari 2001; Feature Petri Nets from Muschevici, Clarke, and Proenca 2010; Context Petri Nets from Cardozo et al. 2012; Adaptive Petri Nets from Mai et al. 2018. Context Petri Nets are explicitly designed for context-aware systems, where different contexts (de)activate parts of the model. For this reason, CoPN were chosen for this work.

4.2 Context Petri Nets in Modelica

In CoPN, contexts are represented as places, each guarded by its activation and deactivation conditions. When a condition is met, the corresponding transition fires, adding/removing a token to/from the place. A token count of **one** indicates an active context; zero means inactive. This token-based representation makes dependencies clear and supports modular control logic and formal verification.

If no VSS is involved, CoPN can be directly applied to existing Modelica models without modifying the original structure. The separation of concerns ensures that the control layer and system layer remain separate, allowing for better maintenance. To achieve it, we can utilize the Petri Nets library *PNlib* (Proß and Bachmann 2012)[1]. Through careful design with the use of the inhibitor arc (which inhibits a transition from firing if the connected place has tokens), CoPN and their four types of context relationships can be effectively represented in Modelica. As shown in Figure 4, different dependencies can also be composed together. However, due to the nature of PNlib, the number of input and output ports for each place and transition must be explicitly defined, and connections between them must be unique: each port can be connected only once, and all ports must be used. Otherwise, the model becomes unbalanced. As the number of context places and their relationships increases, modeling and debugging CoPN becomes increasingly complex. Even with only a few contexts and relationships, debugging can be highly challenging, as demonstrated in Figure 4.

The original CoPN work proposed four relationships between contexts: **weak/strong inclusion**, **exclusion**, and **requirement**. These are explained in detail in Figure 5.

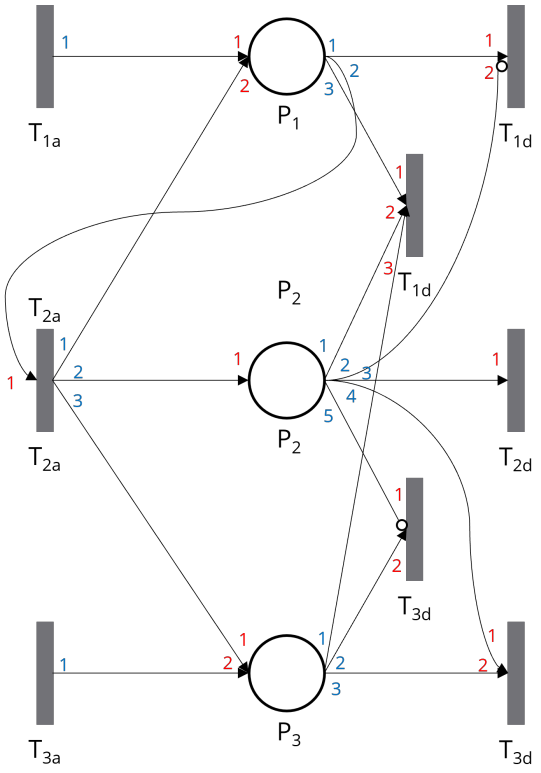


Figure 4. A CoPN containing both requirement and strong inclusion relations. P_2 requires P_1, meaning P_2 can only be active if P_1 is active. P_2 and P_3 are connected by a strong inclusion relation, so they are always activated and deactivated together.

4.3 Composition with CoPNCompositor

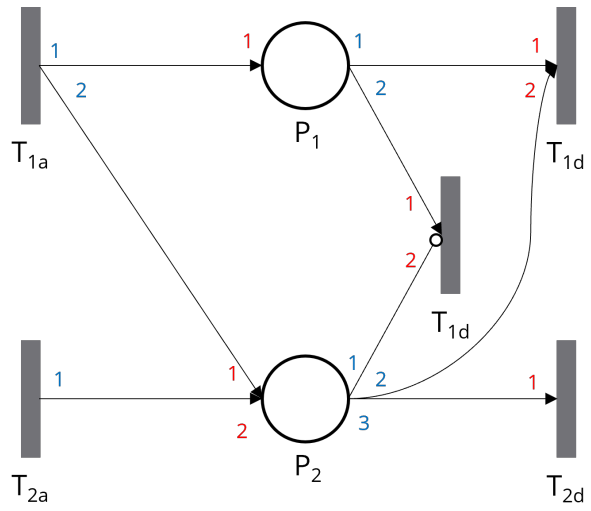
To address this growing complexity with context relationships, the Python-based tool `CoPNCompositor`[2] was developed. It utilizes the Python template engine `Jinja`, where basic components such as places, transitions, and arcs (normal or inhibitor arcs) are defined as templates. Based on specific goals, components are automatically created and connected. Developers can then define contexts, their (de)activation conditions (time-based or event-based), and any context relationships in natural language, as shown in Listing 2. The tool automatically composes the CoPN and generates the corresponding Modelica code.

Listing 2. Configuration interface of CoPNCompositor

```
contexts = ["A", "B", "C", "D", "E"]
weak_inclusions = [("A", "B")]
strong_inclusions = [("B", "C")]
exclusions = [["A", "E"]]
requirements = [("D", "E")]
event_definitions = {
    "A":{"activation": "",
         "deactivation": ""}
    "B":{"activation": "",
         "deactivation": ""}
    ...}
```

[1] https://github.com/AMIT-HSBI/PNlib

[2] https://github.com/wangzizhe/CoPN



Weak Inclusion

P: Discrete Place

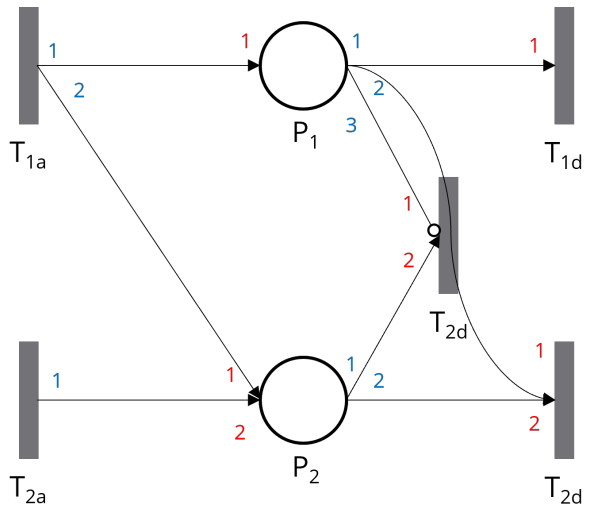
T_a: Activation Transition

T_d: Deactivation Transition

Normal / Inhibitor Arc

1, 2... / 1, 2... Input / Output Ports

(a) Contexts P_1 and P_2 are weakly included (from P_1 to P_2), meaning the (de)activation of P_1 also (de)activates P_2.



Strong Inclusion

P: Discrete Place

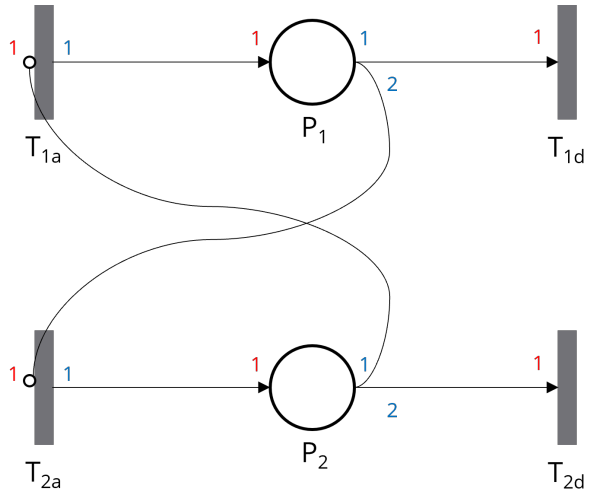
T_a: Activation Transition

T_d: Deactivation Transition

Normal / Inhibitor Arc

1, 2... / 1, 2... Input / Output Ports

(b) Contexts P_1 and P_2 are strongly included (bidirectionally), meaning the (de)activation of one also (de)activates the other.



Exclusion

P: Discrete Place

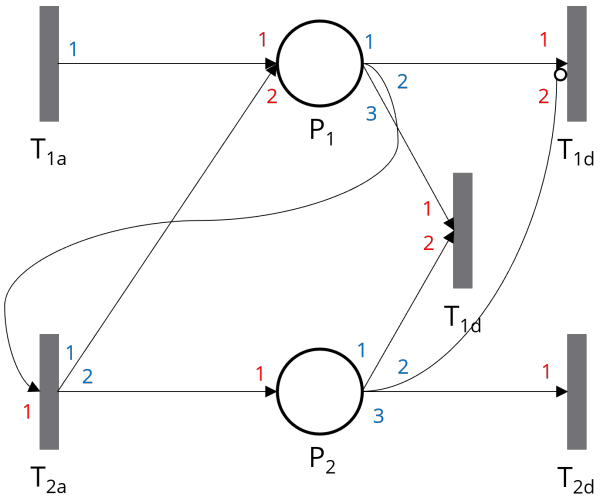
T_a: Activation Transition

T_d: Deactivation Transition

Normal / Inhibitor Arc

1, 2... / 1, 2... Input / Output Ports

(c) Contexts P_1 and P_2 are mutually exclusive, meaning one can only be activated if the other is not active.



Requirement

P: Discrete Place

T_a: Activation Transition

T_d: Deactivation Transition

Normal / Inhibitor Arc

1, 2... / 1, 2... Input / Output Ports

(d) Contexts P_1 and P_2, where P_1 is a prerequisite for P_2; activating P_2 requires P_1 to be active, and deactivating P_1 also deactivates P_2.

Figure 5. Four context relationships of CoPN modeled with PNlib in Modelica

DOI 10.3384/ecp218165

5 Context-Oriented Modelica

In Section 3, we introduced how to realize VSS with FMI. In Section 4, we discussed how CoPN can be applied in Modelica. By combining these two approaches, we can achieve advanced variability management in a more generalized manner and realize context-oriented Modelica.

5.1 ContextModelica

In systems without structural change, CoPN can be directly integrated into the model, as described in the previous section. For systems involving structural change, CoPN act as the central context manager, while FMUVSS handles VSS. As shown in Figure 6, each mode or context (i.e., submodel) is represented as a place in the CoPN. The token count of each place indicates whether the mode/context is active (1) or inactive (0). Each place is guarded by activation and deactivation transitions, and may include context dependencies spanning the entire net. These (de)activation conditions are typically linked to variable values during simulation. CoPN observe and evaluate these values at runtime. When all conditions and dependencies are satisfied, the corresponding transition fires, thereby (de)activating the relevant mode/context.

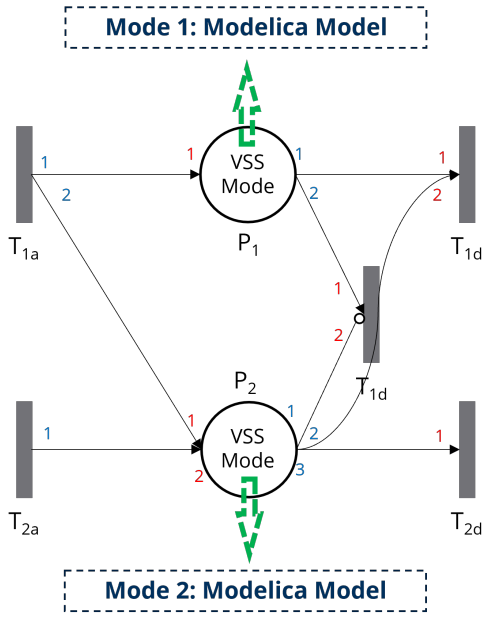


Figure 6. The corresponding Modelica submodels are represented as places in the CoPN. Activating a place triggers FMUVSS to coordinate the simulation of the associated submodel. As a result, the overall model dynamically activates specific submodels based on the currently active contexts. The final output reflects the complete simulation result of the entire system.

Building CoPN outside of Modelica is more flexible, as it avoids the need to manually define the number of ports and their exact connections. Any Python-based Petri Net library that supports inhibitor arcs can be utilized for building the CoPN. In this work, the SNAKES library (Pommereau 2015) has been used to construct the CoPN, as it offers a compact syntax and support for inhibitor arcs.

The framework `ContextModelica` was developed based on this concept to unify the functionality and configuration interfaces of the CoPN compositor and FMUVSS described earlier. Through this framework, developers can define configurations such as various VSS modes, context (de)activation conditions, and their interdependencies. In essence, ContextModelica enables context-oriented modeling in Modelica, allowing for flexible and modular simulation of systems with structural variability.

5.2 Proof of Concept

A simplified IT system model has been applied. To focus on demonstrating the underlying mechanism, the complexity of each submodel has been intentionally reduced. The goal is to validate the proposed concept through a minimal working example.

The system includes two energy supply modes: `greenSupply`, powered solely by green hydrogen, and `hybridSupply`, which combines hydrogen and grid electricity. "H_2ProducedPower" and "loadDemand" are input variables that change over time. Depending on load demand and energy availability, the system switches between three operation modes: `energySaving`, `normal`, and `highPerformance`. Each mode adjusts CPU core count and frequency to reflect the desired performance. (De)activation conditions are based on variable thresholds during simulation, as summarized in Table 2.

Table 2. (De)activation conditions of different modes

Mode	(De)activation Condition
greenSupply	H_2ProducedPower $\geq$ loadDemand
hybridSupply	H_2ProducedPower $<$ loadDemand
energySaving	loadDemand $<$ 150 kW
normal	150 kW $\leq$ loadDemand $<$ 200 kW
highPerformance	loadDemand $\geq$ 200 kW

In addition to conditions, the system enforces **contextual relationships** between modes, as illustrated in Figure 7. Both energy supply modes and operation modes are *mutually exclusive*, ensuring that only one mode from each category is active at any given time. Moreover, some modes depend on others: for instance, `energySaving` requires `greenSupply` to be active, while `highPerformance` depends on `hybridSupply`. Therefore, even if the activation condition is satisfied, these modes will not activate unless their required contexts are also active.

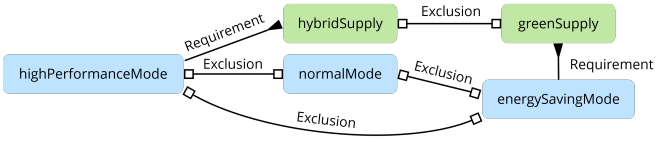


Figure 7. Context relationships of the IT system model

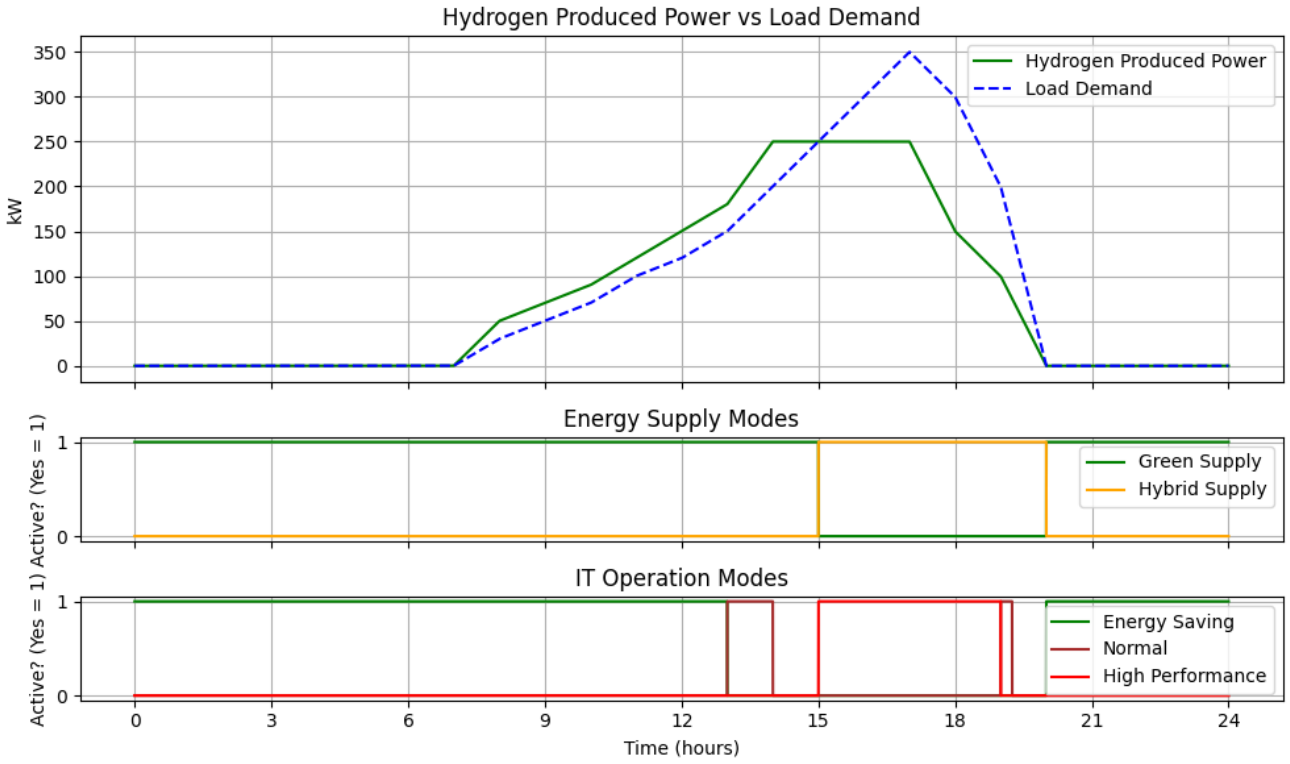


Figure 8. Simulation result of the IT system example

Each mode maps to a specific Modelica submodel. During simulation, the CoPN monitors relevant variables and manages transitions based on defined relationships and guard conditions. The IT system model was simulated over a 24-hour period. Figure 8 illustrates the results. At hour 13, `loadDemand` reached 200 kW, satisfying the activation condition for `highPerformance`. However, since it depends on `hybridSupply`, it only became active at hour 15 when `hybridSupply` was also activated. Likewise, `energySaving` met its condition at hour 19.25 but could not activate until `greenSupply` was active. The delayed activation of `energySaving` and `highPerformance` highlights the importance of context-aware coordination, showcasing the net's ability to synchronize mode interactions safely.

5.3 Managing Submodels

Each mode in a VSS system corresponds to a distinct submodel. Therefore, effectively managing these submodels is essential. The most conventional approach is to export a separate FMU for each VSS mode. While straightforward, this method can become inefficient in systems with tightly coupled components or a large number of modes. In such cases, developers often have to duplicate parameters, variables, and equations across multiple models, which increases maintenance overhead. To address these challenges, we turn to software composition techniques for developing more scalable and maintainable methodologies for managing submodels.

One approach is **model composition**, where individual FMU components are combined dynamically during the VSS simulation based on the active context. FMI supports **hierarchical FMUs**, which allows communication between FMU components via defined input and output ports. However, a major limitation of this approach is the requirement to modify each submodel to expose appropriate ports, which increases maintenance overhead.

Another approach is to encapsulate all submodels within a single unified model and activate only relevant parts based on the active context. This is known as the **Single Underlying Model (SUM)** methodology, as illustrated in Figure 9.

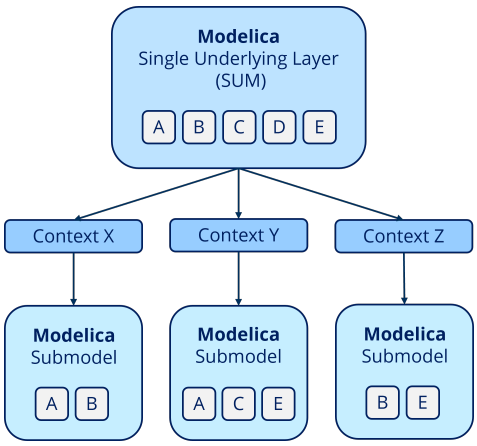


Figure 9. Modelica SUM to submodels based on contexts

DOI
10.3384/ecp218165

In this case, a single model or FMU contains all potential subcomponents, and the active ones are selected at runtime according to the simulation context. The new **Scheduled Execution** FMU type from Gomes et al. 2021 introduced in FMI 3.0 aligns well with this concept. It allows different partitions (representing different modes) within a single FMU to be activated or deactivated based on events. However, as of August 2025, no existing Modelica tools support exporting FMUs using the Scheduled Execution type.

To bridge this gap, the module **ModeGen** was developed as an intermediate solution. ModeGen embraces the core idea of SUM by encapsulating the full system structure in a single Modelica model. It then dynamically generates mode-specific submodels by selectively including or excluding components based on the active context, and exports them as corresponding FMUs. To remain consistent with standard Modelica syntax, a lightweight labeling system/syntax has been introduced to annotate the elements of the Modelica SUM. ModeGen includes a **parser** that reads these labels, interprets them in relation to the active mode or context, and automatically composes the corresponding submodels. A minimal example of a Modelica SUM based on the *Pendulum–Freeflying* scenario is shown in Listing 3. After generation, ModeGen invokes a Modelica compiler to perform model checking for all generated submodels and return the visualized results.

Listing 3. SUM of the *Pendulum-Freeflying* example

```
/*#
  MODEL-METADATA:
    Modes: [pendulum, freeflying]
    Shared: [m, g, y]
#*/

model PendulumFreeflyingSUM
  //# [all]
  parameter Real m = 1;
  parameter Real g = 9.81;
  Real y;

  //# [pendulum]
  parameter Real L = 2;
  Real phi(start=0), dphi(start=2);
  Real x(start=-2), dx, dy, F;

  //# [freeflying]
  Real x, vx, vy;

@#equation
  //# [pendulum]
  x = L * sin(phi); y = -L * cos(phi);
  dx = der(x); dy = der(y);
  dphi = der(phi);
  der(dphi) = -g/L * sin(phi);
  F = m * (g * cos(phi) + L * dphi^2);

  //# [freeflying]
  vx = der(x); vy = der(y);
  m * der(vx) = 0; m * der(vy) = -m * g
```

6 Conclusion and Outlook

6.1 Conclusion

Managing variability in Modelica is challenging due to its static nature and lack of native VSS support, which limits switching between modes and submodels. Constructs like `if-else` and `when` handle only simple cases, while state machines and state graphs offer greater expressiveness but depend on sequential paths, making them less suitable for highly dynamic context-driven changes.

To address these challenges, we introduced CoPN as an advanced variability management pattern for Modelica. CoPN allow for the efficient representation of complex contextual dependencies and dynamic transitions between system states. Additionally, we presented FMUVSS as a framework for realizing VSS within the Modelica ecosystem, offering a comparison with DySMo and identifying DySMo's bottlenecks. With its user-friendly configuration interface, FMUVSS provides a seamless approach to managing variability and dynamic transitions.

By combining CoPN with FMUVSS, we proposed a generalized approach for context-oriented Modelica, where simulations are composed of submodels activated by current contexts. This forms the foundation of a **context-oriented modeling paradigm for Modelica**. The proof of concept demonstrates the potential for advanced variability management in dynamic and adaptive systems. Specifically, we showed that CoPN can be implemented directly in Modelica for systems without structural change and introduced the CoPNCompositor for automated CoPN construction using PNlib. For systems with structural change (VSS), we introduced ContextModelica, which integrates CoPN and FMUVSS.

We further explored submodel management, discussed how hierarchical FMUs can assist in managing submodels, and highlighted the potential of the new Scheduled Execution FMU type introduced in FMI 3.0. This FMU type aligns well with the SUM concept by enabling event-driven mode transitions within a unified FMU. However, since this functionality is not yet widely supported, we proposed an intermediate solution: managing VSS submodels through the SUM, which encapsulates all model information and can be dynamically composed into submodels based on active contexts and modes.

Overall, we demonstrated how context-oriented modeling and programming can be integrated into Modelica, presenting a scalable framework with a unified configuration interface. Its key modules are summarized in Table 3.

Table 3. Key modules for context-oriented Modelica

Module 1: FMUVSS
Module 2: ContextModelica
Module 3: ModeGen
Supporting Tool: CoPNCompositor

6.2 Outlook

Future work aims to further refine and expand the framework, focusing particularly on improving scalability and usability. We plan to integrate the Scheduled Execution FMUs into the framework, once they become widely supported in Modelica tools. Additionally, we plan to explore further collaborations to apply our framework to real-world models, particularly in industries where variability management and dynamic system behavior are critical. These collaborations will provide valuable feedback to help improve the framework.

Acknowledgements

The authors would like to thank the Boysen–TU Dresden–Research Training Group for the financial and general support that has made this contribution possible. The Research Training Group is co-financed by the Friedrich and Elisabeth Boysen Foundation and TU Dresden.

References

Andersson, Christian, Johan Åkesson, and Claus Führer (2016). "PyFMI: A Python Package for Simulation of Coupled Dynamic Models with the Functional Mock-up Interface". In: *Technical Report in Mathematical Sciences* 2.

Baldan, Paolo, Andrea Corradini, and Ugo Montanari (2001). "Contextual Petri Nets, Asymmetric Event Structures, and Processes". In: *Information and Computation* 171.1, pp. 1–49.

Benveniste, Albert et al. (2022). "Algorithms for the Structural Analysis of Multimode Modelica Models". In: *Electronics* 11.17, p. 2755.

Blochwitz, Torsten et al. (2011). "The Functional Mockup Interface for Tool independent Exchange of Simulation Models". In: *8th international Modelica Conference*, pp. 105–114.

Broman, David and Jeremy G Siek (2012). *Modelyze: a Gradually Typed Host Language for Embedding Equation-Based Modeling Languages*. Tech. rep. UCB/EECS-2012-173.

Cardozo, Nicolas et al. (2012). *Context Petri Nets: Definition and Manipulation*. Vrije Universiteit Brussel.

Dey, Anind K. and Gregory D. Abowd (2000). "The Context Toolkit: Aiding the Development of Context-Aware Applications". In: *Workshop on Software Engineering for Wearable and Pervasive Computing*, pp. 431–441.

Elmqvist, Hilding, Sven Erik Mattsson, and Martin Otter (2014). "Modelica extensions for multi-mode DAE systems". In: *10th international Modelica conference*, pp. 183–193.

Elmqvist, Hilding, Andrea Neumayr, and Martin Otter (2018). "Modia - Dynamic Modeling and Simulation with Julia". In: JuliaCon.

Giorgidze, George (2012). "First-Class Models: On a Noncausal Language for Higher-order and Structurally Dynamic Modelling and Simulation". PhD thesis. University of Nottingham.

Gomes, Cláudio et al. (2021). "The FMI 3.0 Standard Interface for Clocked and Scheduled Simulations". In: *14th International Modelica Conference*, pp. 27–36.

Gómez Esperón, Daniel (2017). "Strukturvariabilität in der objektorientierten Modellierung und Simulation durch Generierung von Varianten". PhD thesis. TU Berlin.

Gutsche, Christian, Volodymyr Prokopets, et al. (2024). "Context-Oriented Programming and Modeling in Julia with Context Petri Nets". In: *50th Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2024)*. IEEE, pp. 1–9.

Gutsche, Christian, Zizhe Wang, et al. (2023). "PNRG - A Library for Modeling Variable Structure Energy Grids in Modelica using Energetic Petri Nets". In: *15th International Modelica Conference*, pp. 727–736.

Hirschfeld, Robert, Pascal Costanza, and Oscar Nierstrasz (2008). "Context-Oriented Programming". In: *Journal of Object technology* 7.3, pp. 125–151.

Ma, Yingbo et al. (2021). "Modelingtoolkit: A Composable Graph Transformation System For Equation-Based Modeling". In: *arXiv preprint: 2103.05244*.

Mai, Carl et al. (2018). "Adaptive Petri Nets - A Petri Net Extension for Reconfigurable Structures". In: *10th International Conference on Adaptive and Self-Adaptive Systems and Applications*. Vol. 2.

Mattsson, Sven Erik, Martin Otter, and Hilding Elmquvist (2015). "Multi-mode DAE systems with varying index". In: *11th International Modelica Conference*, pp. 89–98.

Mehlhase, Alexandra (2015). "Konzepte für die Modellierung und Simulation strukturvariabler Modelle". PhD thesis. TU Berlin.

Muschevici, Radu, Dave Clarke, and Jose Proenca (2010). "Feature Petri Nets". In: *14th International Software Product Line Conference (SPLC 2010)*. Vol. 2. Lancaster University, United Kingdom, pp. 99–106.

Nytsch-Geusen, Christoph et al. (2005). "MOSILAB: Development of a Modelica based generic simulation tool supporting model structural dynamics". In: *4th International Modelica Conference*. Vol. 2. Citeseer.

Petri, Carl Adam (1962). "Kommunikation mit Automaten". PhD thesis. Universität Hamburg.

Petzold, Linda R (1982). *Description of DASSL: A Differential/Algebraic System Solver*. Tech. rep. Sandia National Labs., Livermore, CA (USA).

Pommereau, Franck (2015). "SNAKES: A Flexible High-Level Petri Nets Library (Tool Paper)". In: *Application and Theory of Petri Nets and Concurrency: 36th International Conference, PETRI NETS 2015*. Springer, pp. 254–265.

Proß, Sabrina and Bernhard Bachmann (2012). "PNlib - An Advanced Petri Net Library for Hybrid Process Modeling". In: *9th International Modelica Conference*, pp. 3–5.

Sommer, Torsten (2017). *FMPy: A Python Library to simulate Functional Mock-up Units (FMUs)*.

Stüber, Moritz (2017). "Simulating a Variable-structure Model of an Electric Vehicle for Battery Life Estimation Using Modelica/Dymola and Python". In: *12th international Modelica Conference*, pp. 132–031.

Tinnerholm, John et al. (2020). "Towards an Open-Source Modelica Compiler in Julia". In: *Asian Modelica Conference 2020*, pp. 143–151.

Utkin, Vadim (1977). "Variable Structure Systems with Sliding Modes". In: *IEEE Transactions on Automatic control* 22.2, pp. 212–222.

Wang, Zizhe et al. (2024). "Proposal for A Context-Oriented Modelica Contributing to Variable Structure Systems". In: *American Modelica Conference 2024*, pp. 53–62.

Zimmer, Dirk (2010). "Equation-Based Modeling of Variable-Structure Systems". PhD thesis. ETH Zurich.

Benchmarking the Modular Structural Analysis Algorithm

Benoît Caillaud[1] Albert Benveniste[1] Mathias Malandain[1]

[1]Inria Centre at Rennes University, Campus universitaire de Beaulieu, Avenue du Général Leclerc, 35042 Rennes Cedex, France; firstname.lastname@inria.fr

Abstract

In a 2023 Modelica Conference paper, we proposed a novel method for the *modular structural analysis* of DAE systems, in which the structural analysis is not performed on flattened models, but rather at the class level. A new notion of *structural interface* was proposed, in which classes are enriched with context information. That paper developed our approach based on a few illustrative examples.

In this paper, we provide the details of our algorithm. Its performance depends on the system architecture: the analysis of models having a small number of classes (possibly instantiated many times), with a low treewidth system architecture, scales up very efficiently with this approach. We then present additional benchmarks, among which a urban heating network, a representative real-life example on which a near-logarithmic scaling up is shown.

Keywords: DAE, Modelica, object-oriented modeling, structural analysis, index reduction, block-triangular decomposition, interface theory, Difference Bound Matrices

1 Introduction

Modelica advocates an *object-oriented style of programming* (Fritzson 2015), in which subsystem or component models are encapsulated as classes, and instantiated to form large system models. Unfortunately, state of the art structural analysis proceeds by, first, flattening the structured model, and then performing the analysis (Pop et al. 2019). This impedes the storage of model components as executable code or in a format similar to it, which is an crucial feature of compiled programming languages. Additionally, it poses a challenge when attempting to extend DAE-based modeling to higher-order modeling or dynamically changing systems (Broman and Fritzson 2008; Broman 2021). Moreover, as pointed out in (Höger 2015), there are situations in which modeling capabilities are limited by the structural analysis itself, as performance issues arise because of model flattening. Finally, an index-reduction method for DAE systems defined as array equations is proposed in (Otter and Elmqvist 2017). In the same direction, (Fioravanti et al. 2023) proposes an array-aware matching algorithm that scales up to large systems of equations with arrays and loops.

In our previous work (Benveniste et al. 2023) a *structural analysis-aware interface* for components was introduced, thanks to which:

1. structural analysis is performed at the class level;
2. the results of this structural analysis are then instantiated for each component, knowing its context;
3. these component-level structural analyses are composed to derive system-level structural information.

Our approach was presented with the help of a few illustrative examples. In this paper, we provide a formal description of our algorithm. Its performance relates to the following two features of the (otherwise very large) system model: a small number of classes (possibly instantiated many times), and a close-to-tree-shaped system architecture. The urban heating network we present as a benchmark exhibits these features.

The paper is organized as follows. Section 2 provides a background on structural analysis. In Section 4, we introduce Σ-systems, the abstraction of DAE components that this work builds upon, and every function needed to perform the modular structural analysis of models built from these systems. Section 5 presents the core result of this paper, namely, the algorithms used for modular structural analysis. Finally, in Section 6, after addressing implementation details that are key to the performance of the algorithm, we present experimental results obtained from several benchmark models of variable sizes, including an industrial-strength model of the heating network mentioned above.

The full structural analysis also includes the construction of a block-triangular form for the index-reduced system of equations. To simplify our presentation, we ignore this aspect.

2 Background on DAE systems and their structural analysis

We consider square DAE systems, of the form:

$$S \quad : \quad f_j(\text{the } x_i\text{'s and their derivatives}) = 0 \qquad (1)$$

where $X =_{\text{def}} \{x_1, \ldots, x_m\}$ is the set of dependent variables and $F =_{\text{def}} \{f_1, \ldots, f_n\}$ is the set of functions. Throughout this section, we only consider square systems, meaning that $m = n$. Instead of the Pantelides algorithm (Pantelides 1988) for structural index reduction, we will use the Σ-method developed by John Pryce (Pryce 2001). This method is recalled next.

To S as in (1), we associate its (bipartite) *incidence graph* $\mathscr{G}_S = (F \cup X, E)$ having $F \cup X$ as set of vertices, and

having an edge $(f,x) \in E$ if and only if x occurs in function f, regardless of its differentiation degree. To account for differentiations, we augment $\mathscr{G}_S$ to

$$\mathscr{G}_S \quad = \quad (F \cup X, E, \sigma) \tag{2}$$

by adding $\sigma =_{\text{def}} \{\sigma_{f,x} \mid (f,x) \in E\}$, the set of nonnegative integer-valued *weights*, where $\sigma_{f,x}$ is equal to the maximal differentiation degree of variable x in function f. This yields a *weight* for any perfect matching $\mathscr{M}$ of $\mathscr{G}_S$[1] through $\sigma(\mathscr{M}) = \sum_{(f,x)\in\mathscr{M}} \sigma_{f,x}$. The Σ-method consists in solving the following pair of dual linear programming problems:

$$\begin{array}{rll} \text{maximize} & J = \sum_{(f,x)\in E} \sigma_{f,x}\, \xi_{f,x} & \\ \text{subject to} & \sum_{f:(f,x)\in E} \xi_{f,x} = 1 & \forall x \in X \\ \text{and} & \sum_{x:(f,x)\in E} \xi_{f,x} = 1 & \forall f \in F \\ \text{and} & \xi_{f,x} \geq 0 & \forall (f,x) \in E \end{array} \tag{3}$$

$$\begin{array}{rll} \text{minimize} & K = \sum_x d_x - \sum_f c_f & \\ \text{subject to} & d_x - c_f \geq \sigma_{f,x} & \forall (f,x) \in E \\ \text{and} & c_f \geq 0 & \forall f \in F \end{array} \tag{4}$$

Problem (3) is the *primal* and Problem (4) is the *dual*. The former encodes the search for a *maximum weight perfect matching* $\mathscr{M}$ of $\mathscr{G}$, also called *optimal matching* in the sequel. It is proved in (Pryce 2001, Theorem 3.6) that, if a solution exists to the index problem, then a unique element-wise minimal solution for Problem (4) exists. Furthermore, this solution does not depend on the choice of the optimal matching solving the primal problem. The following procedure is proposed in (Pryce 2001) to solve (3,4):

Procedure 1 (Σ-method)

1. *Solve LP* (3)*, which gives an optimal matching* $\mathscr{M}$ *of* $\mathscr{G}$*, described as* $(f,x_f)_{f\in F}$ *or* $(f_x,x)_{x\in X}$*;*

2. *Using* $\mathscr{M}$ *found in step 1, solve for* $(d_x)_{x\in X}$ *the following constraint system:*

$$\begin{array}{rrcl} \forall x \in X: & d_x & \geq & \sigma_{f_x,x} \\ \forall (f,x) \in E: & d_x - d_{x_f} & \geq & \sigma_{f,x} - \sigma_{f,x_f} \end{array} \tag{5}$$

   whereas $c_f = d_{x_f} - \sigma_{f,x_f}$ *yields the equation offsets. Index reduction then consists in differentiating* c_f *times equation* f=0*.*

Notations 1 *We denote by* $\mathscr{L}^{\mathsf{p}}$ *the primal problem* (3) *associated to S, by* $\mathscr{L}^{\mathsf{d}}$ *the problem* (5) *in which matching* $\mathscr{M}=(\xi_{f,x})_{(f,x)\in E}$ *is a solution of primal problem* (3)*, and we write* $\mathscr{L} = (\mathscr{L}^{\mathsf{p}}, \mathscr{L}^{\mathsf{d}})$. □

[1] A perfect matching is a one-to-one assignment of variables to equations, obtained by selecting edges of the incidence graph $\mathscr{G}_S$. The existence of a perfect matching is a criterion for structural nonsingularity.

Using DBMs for the dual problem: In (Pryce 2001), an iterative procedure is proposed to compute the element-wise minimal solution of the dual problem, which is unique. However, handling Problem (5) in a modular way requires additional algebraic manipulations, which are beyond the applicability of the Σ-method. However, we found that Difference Bound Matrices (DBM) (Dill 1989; Miné 2001) turn out to be a subclass of linear programs to which problem (5) belongs.

DBMs are matrices of integer or real coefficients encoding constraint systems like $\mathscr{L}^{\mathsf{d}}$. We generically denote them by A in the sequel. The composition of dual problems $A_i, i = 1,2$ is then encoded by the element-wise minimum of the DBMs: $\mathsf{A}_1 \wedge \mathsf{A}_2$. Eliminating variables from the dual problem is encoded by a matrix denoted by $\Pi_Z(\mathsf{A})$, and the complementary operation (reconstructing after the elimination) is denoted by $\amalg_Z(\mathsf{A})$. Both matrices are easily computed from A.

3 A coupled pendulums model

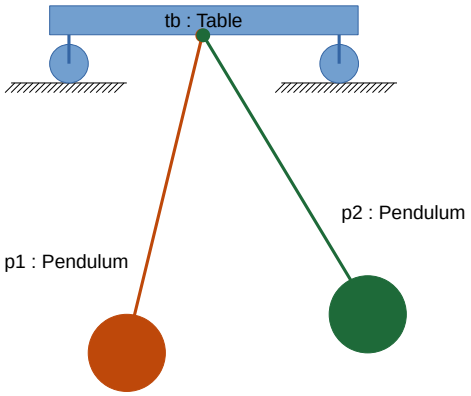


Figure 1. Coupled pendulums, attached to a sliding mass.

Concepts and algorithms introduced presented in the sequel of the paper are illustrated on a model of two 2D coupled pendulums, attached to a mass that can slide on the ground with zero friction, in one dimesion only. The mechanical assembly is depicted Fig. 1, and the Modelica model is defined Fig. 2.

To demonstrate the modularity of the structural analysis method, the model has been as modular as it can be: model `CoupledPendulums` consists of an instance of model `Table`, two instances of model `Pendulum`, and three connecting equations. Two of these connecting equations are trivial, that amount to a variable renaming. Among the three equations, only the third, line 37, a force balance equation, needs to be considered as an equation in the structural analysis. Equations line 9, 26, and 27 are also considered as defining synonyms for the first order derivatives of some state variables. They are not to be considered as equations during the structural analysis.

An important feature of this model is that most variables of models `Table` and `Pendulum` are protected, meaning that they can not be referenced from outside of the model

DOI 10.3384/ecp218175

they belong to. The only variables that remain public are those that are necessary for assembling together the components of the mechanical system, namely: the position of the table `x` and `f`, the projection along the x axis of the forces exerted on the pivot linking the table to each pendulum. As this is explained in the sequel of the paper, this is crucial to the performance of the modular structural analysis.

```
model CoupledPendulums
  model Table
    parameter Real m = 100.;
    public Real x(start=0.,fixed=true);
    protected Real u(start=0.,fixed=true);
    public Real f;
  equation
    m*der(u) - f = 0;
    der(x) = u;
  end Table;
  model Pendulum
    parameter Real m = 1.;
    parameter Real l = 1.;
    parameter Real g = 9.81;
    parameter Real theta0 = 0.;
    public Real x;
    public Real f;
    protected Real a(start=l*sin(theta0),fixed=true);
    protected Real b(start=-l*cos(theta0),fixed=false);
    protected Real u(start=0.,fixed=true);
    protected Real v(start=0.,fixed=true);
    protected Real lambda;
  equation
    m*der(u) - lambda*(a-x);
    m*der(v) - lambda*b + m*g;
    der(a) = u;
    der(b) = v;
    (a-x)^2 + b^2 - l^2 = 0;
    f - lambda*(a-x)/l = 0;
  end Pendulum;
  protected Table tb;
  protected Pendulum p1(theta0=0.1);
  protected Pendulum p2(theta0=0.);
equation
  p1.x = tb.x;
  p2.x = tb.x;
  tb.f + p1.f + p2.f = 0;
end CoupledPendulums;
```

Figure 2. Modelica model of the coupled pendulums.

4 Open DAE systems, Σ-systems, and Σ-interfaces

So far, we focused on closed systems, i.e., systems that operate in isolation. In contrast, open systems are designed to operate in different, yet unspecified environments or contexts. They are the appropriate notion when dealing with modularity. In this section, we introduce open DAE systems and their abstraction for structural analysis, called Σ-systems.

4.1 Open DAE systems and Σ-systems

An *open DAE system* (*oDAE* for short) is a pair $S=(F,X)$, where F is a set of equations $f=0$ and X is a set of variables and their derivatives $x'^k, x \in X$. We require that the cardinalities of these sets satisfy $|X| \geq |F|$. The excess of variables with respect to equations allows capturing interactions with yet unspecified contexts, as shown by the following notion of composition.

Given S_1 and S_2 two oDAEs such that $F_1 \cap F_2 = \emptyset$, their *composition* $S_1 \,\|\, S_2$ is the oDAE $S=(F,X)$ defined by $F = F_1 \uplus F_2$ and $X = X_1 \cup X_2$. Let $S_i, i=1,2$ be two oDAEs. Two solutions $(\mathscr{M}^i, c^i, d^i), i=1,2$ for their respective oDAE index problems are called *complementary* if the union $\mathscr{M}^1 \uplus \mathscr{M}^2$ is a matching, and if variable offsets agree on common variables: $\forall x \in X_1 \cap X_2, d^1_x = d^2_x$. When this holds, matching $\mathscr{M} =_{\text{def}} \mathscr{M}^1 \uplus \mathscr{M}^2$ is equation-complete.

Lemma 1 *Let $S_i, i=1,2$ be two oDAEs, with composition $S = S_1 \,\|\, S_2$. The following properties hold:*

1. *Every solution $(\mathscr{M}, c, d)$ to the oDAE index problem for S projects to a pair of complementary solutions $(\mathscr{M}^i, c^i, d^i), i=1,2$ for the oDAE index problems of $S_i, i=1,2$, where*

$$\mathscr{M}^i = \mathscr{M} \cap (F_i \times X_i),\ c^i = \Pi_{F_i}(c),\ d^i = \Pi_{X_i}(d)\,.$$

2. *Conversely, every pair of complementary solutions $(\mathscr{M}^i, c^i, d^i), i=1,2$ for the oDAE index problems of $S_i, i=1,2$ yields a solution to the oDAE index problem for S by setting*

$$\mathscr{M} = \mathscr{M}^1 \uplus \mathscr{M}^2;\ c_f = c^i_f \text{ if } f \in F_i;\ d_x = d^i_x \text{ if } x \in X_i\,.$$

To account for the fact that the structural analysis of DAEs and oDAEs works on weighted incidence graphs only, we introduce the adequate abstraction for this, called Σ-system. To this end, we distinguish between shared and local variables,[2] by enforcing the following assumption:

Assumption 1 *For $S=(F,X)$ an oDAE, we assume a decomposition $X = X^s \uplus X^\ell$ of its set X of variables into its subsets of* shared *and* local *variables. By convention, local variables of* **S** *are never shared with any other oDAE.*[3]

Local variables are determined locally, by the oDAE equations, whereas shared variables can be determined, depending on the cases, either locally, by the oDAE system, or externally, by the equations of its environment. Interaction with the environment occurs through shared variables only.

Since the structural analysis of oDAE S is built on top of its incidence graph $\mathscr{G}_S$ only, we consider an abstraction of S as $\mathscr{G}_S$, now called a Σ-system:

1. A Σ-*system* is a tuple $\mathbf{S} = (F, X^\ell, X^s, E, \sigma)$, where:

[2] For instance, local variables are declared using the `protected` modifier in Modelica, while shared variables are declared using the default `public` modifier.

[3] We ensure that local variable identifiers are chosen unique, e.g., by performing appropriate renaming to avoid local variable identifier clashes.

- F is a set of functions; X^ℓ and X^s are two disjoint sets of *local* and *shared* variables; we set $X{=}X^s{\uplus}X^\ell$ (the cardinality of F is greater than, or equal to, that of X^ℓ);
- $E \subseteq F \times X$ is the set of *edges*; and
- $\sigma : E \to \mathbb{N}$ is the *weight function*.

2. Given two Σ-systems $\mathbf{S}_1$ and $\mathbf{S}_2$, their *composition* $\mathbf{S}_1 \,\|\, \mathbf{S}_2$ is only defined if $F_1 \cap F_2 = \emptyset$ and yields the Σ-system $\mathbf{S} = (F_1 \uplus F_2, X_1^\ell \uplus X_2^\ell, X_1^s \cup X_2^s, E_1 \uplus E_2, \sigma^1 \uplus \sigma^2)$.

3. Given $x \in X^s$, the *hiding* of x in $\mathbf{S}$ is the Σ-system denoted by $\mathbf{S}{\downarrow_x}$ and obtained by replacing, in $\mathbf{S}$, the pair (X^ℓ, X^s) with $(X^\ell \cup \{x\}, X^s \setminus \{x\})$.

4. Given $x{\in}X^s$ and $y{\notin}X$, the *renaming* of x by y in $\mathbf{S}$ is the Σ-system denoted by $\mathbf{S}[y/x]$ and obtained by replacing x by y everywhere in $\mathbf{S}$.

The combined use of these operators allows to define all Modelica constructs as macros (with the exception of *inner/outer* variable declarations, which imply complex name scoping rules).

Using Section 2, we can associate to a Σ-system $\mathbf{S}$ its oDAE index problem, which, reusing Notations 1, we denote by $\mathscr{L}_\mathbf{S} = (\mathscr{L}_\mathbf{S}^\mathsf{p}, \mathscr{L}_\mathbf{S}^\mathsf{d})$, or simply $\mathscr{L} = (\mathscr{L}^\mathsf{p}, \mathscr{L}^\mathsf{d})$. Its solutions take the form $(\mathscr{M}, c, d)$, where matching $\mathscr{M}$ is equation-complete. The variables matched in $\mathscr{M}$ are determined by the system itself, therefore we call them *dependent variables* and denote the resulting set by X^dep. There are generally several possible choices for $X^\mathsf{dep} \subseteq X$, depending on which variables from X^s are chosen to be part of it.

This choice is parametrized using the notion of selector. Let $\mathbf{S} = (F, X^\ell, X^s, E, \sigma)$ be a Σ-system. Call *selector* any subset $Y \subseteq X^s$, and consider:

- $X^\mathsf{dep} = X^\ell \uplus Y$ and $E^Y = E \cap (F \times X^\mathsf{dep})$;
- σ^Y, the restriction, to E^Y, of the weight function σ;
- $\mathscr{G}^Y = (F \cup X^\mathsf{dep}, E^Y, \sigma^Y)$, the incidence graph; and
- any optimal matching $\mathscr{M}^Y$ for $\mathscr{G}^Y$, if it exists.

Call $\mathscr{M}^Y$ the *selected optimal matching* (SOM) for $\mathbf{S}$ with Y, and *selected optimal weight* its associated weight.

The following result will be instrumental in identifying the appropriate notion of interface required for structural analysis:

Theorem 1 *Let $\mathbf{S}_1$ and $\mathbf{S}_2$ be two Σ-systems, and let $\mathbf{S} = \mathbf{S}_1 \,\|\, \mathbf{S}_2$ be their composition. The following formula holds, where Y is any selector for $\mathbf{S}$ and Y_i is any selector for $\mathbf{S}_i, i{=}1,2$:*

$$\mathsf{J}^Y = \max \left\{ \mathsf{J}_1^{Y_1} + \mathsf{J}_2^{Y_2} \;\middle|\; Y_1 \cap Y_2 = \emptyset \text{ and } Y_1 \cup Y_2 = Y \right\}. \quad (6)$$

We now move to considering the dual problem. For $\mathbf{S}$ a Σ-system and $Y \subseteq X^s$ a selector for it, we denote by $\mathscr{L}^{Y\mathsf{d}}$ the dual problem (5) with $X^\mathsf{dep} = X^\ell \uplus Y$ and $X^\mathsf{free} = X^s \setminus Y$. The following result relates the dual problem of a composition to the dual problems of each of the composed open systems:

Theorem 2 *Let $\mathbf{S}_1$ and $\mathbf{S}_2$ be two open systems and $\mathbf{S} = \mathbf{S}_1 \,\|\, \mathbf{S}_2$ their composition. Denoting by $\mathscr{L}_i^{Y_i\mathsf{d}}, i{=}1,2$ the dual problem of $\mathbf{S}_i$ with selector Y_i, then, for* any *choice of (Y_1, Y_2) achieving the max in (6):*

$$\mathscr{L}^{Y\mathsf{d}} \equiv \mathscr{L}_1^{Y_1\mathsf{d}} \cup \mathscr{L}_2^{Y_2\mathsf{d}}, \quad (7)$$

$$\mathsf{A}^Y = \mathsf{A}_1^Y \wedge \mathsf{A}_2^Y. \quad (8)$$

In (7), the $\cup$ on the right hand side indicates the union of sets of constraints, whereas the symbol $\equiv$ means that the two sides possess identical sets of solutions.

4.2 Σ-interfaces

The Σ-interface is *precisely what is required for interactions with the outside world* — no more, no less. We complement it with the notions of local decomposition and extension, which summarize the missing information in the Σ-interfaceneeded to reconstruct solutions to the structural analysis of the Σ-system.

Notations 2 *In the sequel, given a relation $R \subseteq W \times W$ or a function $f : W \to Z$, and $V \subseteq W$, we denote by*

$$\Pi_V(R) \text{ and } \Pi_V(f)$$

the restriction *of R to $V \times V$ and of f to V.*[4] *We will use this notation with some liberty. For $Y \subseteq X^s$ a selector, we will write $\Pi_Y(\mathsf{J})$ instead of $\Pi_{\mathscr{P}(Y)}(\mathsf{J})$ to mean the restriction of function J to $\mathscr{P}(Y)$, the set of all subsets of Y.*

For the following definition, we reuse the DBM associated to the dual problem introduced in Section 2.

Definition 1 (interface) *To Σ-system $\mathbf{S} = (F, X^\ell, X^s, E, \sigma)$, we associate its* Σ-interface

$$\mathbf{I}_\mathbf{S} = \left(X^s, \mathsf{J}, \Pi_{X^s}(\mathsf{A})\right), \quad (9)$$

where J and $\Pi_{X^s}(\mathsf{A})$ are functions that respectively map any selector $Y \subseteq X^s$ to:

- J^Y*, the optimal weight of $\mathbf{S}$ for the primal problem with selector Y;*
- $\Pi_{X^s}(\mathsf{A}^Y)$*, the projection, over X^s, of the DBM A^Y associated to the dual problem with selector Y.*

In our modular algorithm, we use the following function:

$$\text{INTERFACECOINTERFACE}(\mathbf{S}) = \left(\mathbf{I}_\mathbf{S}, \amalg_{X^s}(\mathsf{A})\right) \quad (10)$$

[4] Our notation is reminiscent of a projection, on purpose.

DOI 10.3384/ecp218175

where $\amalg_{X^s}(\mathsf{A})$ is a function that maps each selector $Y \subseteq X^s$ to $\amalg_{X^s}(\mathsf{A}^Y)$, the extension of A^Y from X^s. Function (10) splits $\mathbf{S}$ as its Σ-interface $\mathbf{I_S}$ and an object $\amalg_{X^s}(\mathsf{A}^Y)$ that makes up for the loss of information in $\mathbf{I_S}$. The latter is stored for subsequent reuse when "reverting the Σ-interface", as we shall see.

To illustrate the concept of Σ-interface, let us consider model `Table` (Fig. 2). It has two public variable, `x` and `f`. Protected variable `u` (line 5) should be seen as a synomym for the first order derivative of `x` (equation line 9). This means that for the structural analysis, this model has two variables and one equation (line 8). As a consequence, the interface $I_t = (\{x,f\}, J_t, A_t)$ of this model admits two valid selectors: $\{x\}$ and $\{f\}$. The optimal weight J_t and the DBM A_t are given in the table below, for each of the two valid selectors:

Y	J_t	A_t
$\{f\}$	0	$A_t^{\{f\}}$
$\{x\}$	2	$A_t^{\{x\}}$

where:

$$A_t^{\{f\}} = \begin{array}{c} \\ 0 \\ x \\ f \end{array} \begin{array}{c} \begin{array}{ccc} 0 & x & f \end{array} \\ \begin{bmatrix} & -2 & 0 \\ & & \\ & -2 & 0 \end{bmatrix} \end{array} \qquad A_t^{\{x\}} = \begin{array}{c} \\ 0 \\ x \\ f \end{array} \begin{array}{c} \begin{array}{ccc} 0 & x & f \end{array} \\ \begin{bmatrix} & -2 & 0 \\ & 0 & 2 \\ & & \end{bmatrix} \end{array}$$

Matrix elements that are left blank have the value $+\infty$, and impose no constraint on the corresponding offset variables. Therefore, DBM $A_{\{f\}}$ is equivalent to the system of inequalities $d_x \geq 2, d_f \geq 0, d_x \geq 2 + d_f$. The second DBM $A_{\{x\}}$ encodes $d_x \geq 2, d_f \geq 0, d_f \geq d_x - 2$.

Theorem 3 (hiding) *Let* $\mathbf{S}$ *be a* Σ*-system, with* Σ*-interface* (9). *Let* $\mathbf{T} =_{\text{def}} \mathbf{S}{\downarrow_Z}$*, where* $Z \subseteq X^s$*, and define* $Z^s = X^s \backslash Z$*. Then, the* Σ*-interface of* $\mathbf{T}$ *is*

$$\mathbf{I_T} = \left(Z^s, \Pi_{Z^s}(\mathsf{J}), \Pi_{Z^s}(\mathsf{A})\right),$$

where $\Pi_{Z^s}(\mathsf{J})$ *and* $\Pi_{Z^s}(\mathsf{A})$ *are functions mapping any selector* $Y \subseteq Z^s$ *to* $\Pi_{Z^s}(\mathsf{J}^Y)$ *and* $\Pi_{Z^s}(\Pi_{X^s}(\mathsf{A}^Y))$.

As a direct consequence of the properties of the projection, note that, for every selector Y,

$$\Pi_{Z^s}(\mathsf{A}^Y) \quad = \quad \Pi_{Z^s}(\Pi_{X^s}(\mathsf{A}^Y)) \tag{11}$$

In our modular algorithm, we will use the following function, where $\mathbf{I_S}$ is as in (9):

$$\text{HIDEEXPOSE}(\mathbf{I_S}, Z) = \left(\mathbf{I_T}, \amalg_Z(\mathsf{A})\right) \tag{12}$$

In (12), $\amalg_Z(\mathsf{A})$ is the function mapping each selector $Y \subseteq X^s$, to $\amalg_{Z^s}(\mathsf{A}^Y)$, the extension of A^Y from Z^s. Function (12) replaces the pair $(\mathbf{I_S}, Z)$ by its Σ-interface $\mathbf{I_T}$, and object $\amalg_{Z^s}(\mathsf{A}^Y)$ that makes up for the loss of information in $\mathbf{I_T}$. The latter is stored for subsequent reuse when "reverting the hiding".

Theorem 3 and (11) show that HIDEEXPOSE can be evaluated through the Σ-interfaces $\mathbf{I_S}$ and $\mathbf{I_T}$ only.

Theorem 4 (composition) *Let* $\mathbf{S}_1$ *and* $\mathbf{S}_2$ *be two* Σ*-systems, with* Σ*-interfaces* $\mathbf{I}_{\mathbf{S}_i} = (X_i^s, \mathsf{J}_i, \mathsf{A}_i), i = 1,2$.

1. *We have* $\mathbf{I}_{\mathbf{S}_1 \| \mathbf{S}_2} = (X^s, \mathsf{J}, \mathsf{A})$*, where:*

$$\begin{aligned} X^s &= X_1^s \cup X_2^s, \text{ and, for } Y \subseteq X^s, \\ \mathsf{J}^Y &= \max \left\{ \mathsf{J}_1^{Y_1} + \mathsf{J}_2^{Y_2} \;\middle|\; \begin{array}{l} Y_i \subseteq X_i^s, i{=}1,2 \\ Y_1 \cap Y_2 = \emptyset \\ Y_1 \cup Y_2 = Y \end{array} \right\} \qquad (13) \\ \mathsf{A}^Y &= \mathsf{A}_1^{Y_1} \wedge \mathsf{A}_2^{Y_2} \qquad (14) \end{aligned}$$

2. *The resulting* A^Y *in* (14) *is independent from the chosen optimizing pair.*

The right-hand sides of (13, 14) depend only on the Σ-interfaces of $\mathbf{S}_1$ and $\mathbf{S}_2$. Hence, we will denote by $\mathbf{I_{S_1}} \| \mathbf{I_{S_2}}$ the so defined operation on Σ-interfaces.

Call *optimizing pair* the function $Y \mapsto (Y_1, Y_2)$, mapping any Y to a pair (Y_1, Y_2) achieving the maximum in (13). In our modular algorithm, we use the following function:

$$\text{COMPOSEDECOMPOSE}(\mathbf{I_{S_1}}, \mathbf{I_{S_2}}) = \begin{pmatrix} \mathbf{I_{S_1}} \| \mathbf{I_{S_2}} \\ Y \mapsto (Y_1, Y_2) \end{pmatrix} \tag{15}$$

computing the optimizing pair as its second component, for subsequent reuse when "reverting the composition".

To illustrate the composition operator, let us consider the composition of model `Table` with the equation "`der(f) + x = 0`". The interface of the former is given above, while the latter has for interface $I_e = (\{x,f\}, J_e, A_e)$, where J_e and A_e are defined as follows:

Y	J_e	A_e
$\{f\}$	1	$A_e^{\{f\}}$
$\{x\}$	0	$A_e^{\{x\}}$

where:

$$A_e^{\{f\}} = \begin{array}{c} \\ 0 \\ x \\ f \end{array} \begin{array}{c} \begin{array}{ccc} 0 & x & f \end{array} \\ \begin{bmatrix} & 0 & -1 \\ & & \\ & 1 & 0 \end{bmatrix} \end{array} \qquad A_e^{\{x\}} = \begin{array}{c} \\ 0 \\ x \\ f \end{array} \begin{array}{c} \begin{array}{ccc} 0 & x & f \end{array} \\ \begin{bmatrix} & 0 & -1 \\ & 0 & -1 \\ & & \end{bmatrix} \end{array}$$

The composition $(I_t \parallel I_e) = (\{x,f\}, J, A)$ is as follows: it admits only one valid selector $Y = \{x,f\}$, which optimal partitioning is $Y_t = \{x\}$ and $Y_e = \{f\}$. This gives a weight $J^Y = J_t^{Y_t} + J_e^{Y_e} = 3$. DBM A^Y is the pointwise minimum of $A_t^{Y_t}$ and $A_e^{Y_e}$:

$$A^Y = \begin{array}{c} \\ 0 \\ x \\ f \end{array} \begin{array}{c} \begin{array}{ccc} 0 & x & f \end{array} \\ \begin{bmatrix} & -2 & -1 \\ & 0 & 2 \\ & 1 & 0 \end{bmatrix} \end{array}$$

4.3 Reverting the Σ-interface

In this section, we formalize what was meant above by "reverting" interfaces, hiding, and composition.

Consider a Σ-system $\mathbf{S} = (F, X^\ell, X^s, E, \sigma)$. Call Σ-*solution of* $\mathbf{S}$ any tuple

$$\nu \quad = \quad \big(Y, (d_x)_{x\in X}, (c_f)_{f\in F}\big)\,, \tag{16}$$

where $Y \subseteq X^s$ is a chosen selector such that $\mathsf{J}^Y > -\infty$, and variable offsets $(d_x)_{x\in X}$ are solution of A^Y, whereas equation offsets $(c_f)_{f\in F}$ are computed as in Procedure 1. We denote by $\mathsf{sol}(\mathbf{S})$ the set of all Σ-solutions of $\mathbf{S}$. We equip $\mathsf{sol}(\mathbf{S})$ with the partial order $\leq$ defined by

$$\nu \leq \nu' \quad \text{iff} \quad Y = Y' \text{ and } \forall x \in X : d_x^Y \leq d_x'^Y\,. \tag{17}$$

The term "minimal" will be referring to this order.

Let $\mathbf{I}$ be a Σ-interface following (9). Call *local solution* of $\mathbf{I}$ any tuple

$$\mu \quad = \quad \big(Y, (d_x)_{x\in X^s}\big)\,, \tag{18}$$

where $Y \subseteq X^s$ is a chosen selector such that $\mathsf{J}^Y > -\infty$, and variable offsets $(d_x)_{x\in X^s}$ are solution of $\Pi_{X^s}(\mathsf{A}^Y)$. Minimality is defined as in (17).

Nota: Σ-solutions are the wanted solutions of the index reduction problem. They are large objects since their cardinality relates to the size of X. In contrast, local solutions are small objects, whose cardinality relates to the size of X^s. Local solutions are the minimal objects from which Σ-solutions for the considered Σ-system can be rebuilt. Our overall objective is to compute Σ-solutions in a modular way, by manipulating only local solutions.

When composing Σ-interfaces, their local solutions fuse into local solutions of the composed interface. Local decomposition is the reverse operation.

Theorem 5 (local decomposition) *Let $\mathbf{I}_i, i{=}1,2$ be two Σ-interfaces and let $\mathbf{I}{=}\mathbf{I}_1 \,\|\, \mathbf{I}_2$ be their composition.*

1. *Let $\mu_i, i{=}1,2$ be a local solution of $\mathbf{I}_i, i{=}1,2$. Say that μ_1 and μ_2 are* compatible *if, with reference to Theorem 4:*

   (a) (Y_1, Y_2) is the optimizing pair for $Y{=}Y_1{\cup}Y_2$, and

   (b) for every $x, y \in X_1^s{\cap}X_2^s$, we have: $d_{1,x} = d_{2,x}$.

   If μ_1 and μ_2 are compatible, we can define their join

$$\begin{array}{rcl} \mu_1 \sqcup \mu_2 & =_{\text{def}} & (Y, (d_x)_{x\in X^s})\,, \text{ where:} \\ Y & = & Y_1 \cup Y_2 \\ d_x & = & \text{if } x \in X_1^s \text{ then } d_{1,x} \text{ else } d_{2,x} \end{array} \tag{19}$$

2. *Conversely, any Σ-solution μ of $\mathbf{I}{=}\mathbf{I}_1 \,\|\, \mathbf{I}_2$* decomposes *as $\mu = \mu_1 \sqcup \mu_2$, where μ_i is a Σ-solution of $\mathbf{I}_i$ and*

   (a) Y decomposes as $Y = Y_1{\cup}Y_2$, where (Y_1, Y_2) is the pair returned by the function COMPOSEDECOMPOSE *defined in (15), and*

   (b) $(d_{i,x})_{x\in X_i^s}$ is the restriction of $(d_x)_{x\in X^s}$ to X_i^s.

The important step in this decomposition is 2a. Step 2b is just a restriction and does not need any side information.

Nota: The decomposition $\mu = \mu_1 \sqcup \mu_2$ may not be unique, since optimizing pairs of selectors can be several. However, step 2b always yields the same result.

We now focus on how to revert the effect of hiding on Σ-interfaces. We use macro (12) together with objects and notations of Lemma 3.

Theorem 6 *Let $\mathbf{I}$ be a Σ-interface as in (9), and let $Z \subseteq X^s$ and $Z^s = X^s \setminus Z$. Consider the interface $\mathbf{J}$ of the hiding $\mathbf{I}{\downarrow_Z}$.*

1. *Let $\mu = (Y, (d_x)_{x\in X^s}, \preceq)$ be a local solution of $\mathbf{I}$ such that $Y \cap Z = \emptyset$. The* hiding of Z in μ, *defined by*

$$\mu{\downarrow_Z} \quad = \quad (Y, (d_x)_{x\in Z^s})\,, \tag{20}$$

   is a local solution of $\mathbf{J}$ and every local solution of $\mathbf{J}$ is obtained in this way. Say that $\mu{\downarrow_Z}$ extends to *μ.*

2. *Conversely, every local solution μ of $\mathbf{J}$ extends to a unique local solution $\mu^\star$ of $\mathbf{I}$, which is minimal among the set of all local solutions of $\mathbf{I}$ extending μ. This minimal extension is computed by using only $\amalg_Z(\mathsf{A})$, stored as the second component of the function* HIDEEXPOSE *introduced in* (12).

Theorem 7 *Let $\mathbf{S}$ be a basic Σ-system with Σ-interface $\mathbf{I}_\mathbf{S}$, and let $\nu = \big(Y, (d_x)_{x\in X}, (c_f)_{f\in F}\big)$ be a Σ-solution of $\mathbf{S}$.*

1. *The* hiding of X^ℓ in ν, *defined by*

$$\nu{\downarrow_{X^\ell}} \quad = \quad (Y, (d_x)_{x\in X^s})\,,$$

   is a local solution of $\mathbf{I}_\mathbf{S}$ and every local solution of $\mathbf{I}_\mathbf{S}$ is obtained in this way. Say that $\nu{\downarrow_{X^\ell}}$ extends to *ν.*

2. *Conversely, every local solution μ of $\mathbf{I}_\mathbf{S}$ extends to a unique Σ-solution $\nu = \mu^\star$ of $\mathbf{S}$, which is minimal among the set of all Σ-solutions of $\mathbf{S}$ extending μ. This minimal extension is computed by using only the pair $\amalg_{X^s}(\mathsf{A})$, which was stored as the second component of the function* INTERFACECOINTERFACE *in* (10).

This completes the toolbox for computing a Σ-solution of the Σ-system in a modular way. With the exception of basic Σ-systems, only shared variables are involved. Hence, these concepts remain as small as the set of shared variables of the associated Σ-systems.[5]

[5] Referring to the Modelica taxonomy, we expect all programs to have a set of *public* variables of small cardinality. In addition, we expect basic classes to possess a small set of variables.

DOI 10.3384/ecp218175

5 Modular algorithm

This section contains the most significant and practical contribution of this work, namely: a term algebra for the modular definition of Σ-systems, and a modular algorithm performing their structural analysis.

While the structural interface theory developed in Section 4.2 provides the needed pillars, it is not sufficient to construct an efficient and scalable algorithm, taking advantage of certain assumptions regarding the DAE systems we target.

Indeed, the large DAE systems that are typically seen in the digital twins of industrial systems are sparse. In addition, they generally result from a composition of a large number of instances of basic models in much smaller number, e.g., taken from libraries. Said differently, our targeted DAE systems contain a large number of structurally identical components, in which only parameter values may differ.

Our modular algorithm ensures that structural analyses are computed only once for each representative component, and then properly fused to get the global analysis. Achieving this relies on the interface theory developed in Section 4.2, complemented with adequate algorithmic techniques and data structures that we explain in this section.

Algorithm 1 Upward sweep: computing interfaces and cointerfaces with memoization

global $\mathscr{I}C : \mathscr{A} \to \mathscr{I} \times \mathscr{C}$ ▷ memoization table
▷ initially empty
function INTERFACE(a)
  if $\mathscr{I}C(a)$ is defined **then** ▷ if already memoized
    $(i,\cdot) \leftarrow \mathscr{I}C(a)$ ▷ then retrieve interface
  else
    if $a = (a_1 \parallel a_2)$ **then** ▷ composition
      $i_1 \leftarrow$ INTERFACE(a_1) ▷ compute interfaces
      $i_2 \leftarrow$ INTERFACE(a_2) ▷ of operands
      $(i,c) \leftarrow$ COMPOSEDECOMPOSE(i_1,i_2)
    else if $a = a'\downarrow_x$ **then** ▷ hiding
      $i' \leftarrow$ INTERFACE(a') ▷ interface of operand
      $(i,c) \leftarrow$ HIDEEXPOSE(i',x) ▷ compute hiding
    else if $a = [y/x]a'$ **then** ▷ renaming
      $i' \leftarrow$ INTERFACE(a') ▷ interface of operand
      $(i,c) \leftarrow$ RENAMEREVERSE(i',y,x)
    else if $a = b$ **then** ▷ basic Σ-system
      $(i,c) \leftarrow$ INTERFACECOINTERFACE(S_b)
    end if
    $\mathscr{I}C(a) \leftarrow (i,c)$ ▷ memoize (co-)interface
  end if
  return i ▷ return interface
end function

Algorithm 2 Downward sweep: computing local solutions

global $\mathscr{I}C$ ▷ initialized by function INTERFACE
  $\mathscr{S}ol \subseteq \mathscr{A} \times \mathscr{S}$ ▷ set of local solutions
  ▷ initially empty
procedure SOLUTION(a,μ)
  if $(a,\mu) \notin \mathscr{S}ol$ **then** ▷ visited yet ?
    add (a,μ) to $\mathscr{S}ol$ ▷ mark visited
    $(\cdot,c) \leftarrow \mathscr{I}C(a)$ ▷ retrieve cointerface
    if $a = (a_1 \parallel a_2)$ **then** ▷ composition
      $(\mu_1,\mu_2) \leftarrow c(\mu)$ ▷ compute local solutions
      ▷ of both operands
      SOLUTION(a_1,μ_1) ▷ propagate local solutions
      SOLUTION(a_2,μ_2) ▷ to the operands
    else if $a = a'\downarrow_x$ **then** ▷ hiding
      $\mu' \leftarrow c(\mu)$ ▷ compute local solution
      ▷ of the operand
      SOLUTION(a',μ') ▷ propagate to operand
    else if $a = [y/x]a'$ **then** ▷ renaming
      $\mu' \leftarrow c(\mu)$ ▷ compute renamed solution
      SOLUTION(a',μ') ▷ propagate to operand
    else if $a = b$ **then** ▷ basic Σ-system
      $\nu \leftarrow c(\mu)$ ▷ compute Σ-solution
      generate reduced index system of
      S_b for Σ-solution ν
    end if
  end if
end procedure

5.1 A term algebra $\mathscr{A}$ of Σ-systems

In this section, we consider a term algebra of Σ-systems, defined by the following grammar:

$$\mathscr{A} \quad ::= \quad \mathscr{B} \mid (\mathscr{A} \parallel \mathscr{A}) \mid \mathscr{A}\downarrow_{\mathscr{X}} \mid \mathscr{A}[\mathscr{X}/\mathscr{X}] \tag{21}$$

where $\mathscr{B}$ is a predefined set of symbols b denoting basic Σ-system S_b, and $\mathscr{X}$ is a set of variable names x, y, or z. The semantics of $\mathscr{A}$ is a mapping that associates a Σ-system $[\![a]\!]$ to term a. It is defined inductively on the term:

$$\begin{aligned} [\![b]\!] &= S_b \\ [\![(a_1 \parallel a_2)]\!] &= \rho_1[\![a_1]\!] \parallel \rho_2[\![a_2]\!] \\ [\![a\downarrow_x]\!] &= [\![a]\!]\downarrow_x \\ [\![a[y/x]]\!] &= (\rho[\![a]\!])[y/x] \end{aligned} \tag{22}$$

where ρ_k, $k = 1,2$ are any injective renamings of the vertices of $[\![a_k]\!] = (F_k, X_k^\ell \uplus X_k^s, E_k, \sigma_k)$, $k = 1,2$, such that $F_1 \cap F_2 = \emptyset$, $X_1^\ell \cap X_2^\ell = \emptyset$, and ρ_k, $k = 1,2$ are identity functions on X_k^s. Renaming ρ is any mapping such that it is the identity on the shared variables of $[\![a]\!]$ and y is not a variable of $\rho[\![a]\!]$. Our central problem is the following:

Problem 1 *How can one perform the structural analysis of $[\![a]\!]$ without computing $[\![a]\!]$, that is, by considering only term a and the Σ-systems S_b of the basic terms b appearing in a?*

A solution to this problem is developed in this section. The following notations are used in the sequel:

Notations 3 *In the sequel, symbols a, a', a_1, a_2 are use to represent terms; i, i', i_1, i_2 denote Σ-interfaces; c denotes cointerfaces and μ, μ', μ_1, μ_2 represent local solutions, while symbol ν is used to represent Σ-solutions. Also, we will use the generic term of* Cointerface *to denote the complementary components computed by the functions* INTERFACECOINTERFACE, HIDEEXPOSE, *and* COMPOSEDECOMPOSE, *introduced in (10,12,15).* □

5.2 A modular algorithm for the structural analysis of $\mathscr{A}$

In this section, we propose an algorithm solving Problem 1. This algorithm consists in traversing the sub-term graph, by first performing an *upward sweep* using Algorithm 1, followed by a *downward sweep* using Algorithm 2, which we comment next.

Upward sweep Algorithm 1 Σ-interfaces are computed inductively, starting from basic Σ-systems. A *memoization* table $\mathscr{I}C: \mathscr{A} \to \mathscr{I} \times \mathscr{C}$ is used to store the local interface and cointerface for each sub-term. Memoization (Cormen et al. 2022, Section 14.3) is an algorithmic technique that consists in ensuring that a function $f(x)$ is evaluated at most once for each parameter value x. This relies on a data structure that stores $f(x)$ for each value x encountered so far during program execution. Table $\mathscr{I}C$ has a double purpose: it is both used to prevent the evaluation of function INTERFACE(a) whenever this has already been done, and to store the local cointerface associated to a term. This cointerface is used in the downward sweep.

Downward sweep Algorithm 2 Local solutions are computed for each sub-term. In procedure SOLUTION(a, μ), a is a term and μ a local solution, seen as its context. This procedure computes the local solutions of the sub-terms of a, in the context of μ. To do this, the local cointerface c of term a, computed during the first phase, is retrieved from table $\mathscr{I}C$.

In the sequel, we call *modular structural analysis* the successive execution of Algorithms 1 and 2.

5.3 Back to the coupled pendulums example

The coupled pendulums model, introduced Section 3, can be expressed as the following component algebra term:

$$
\begin{aligned}
\text{CoupledPendulums} \;=\; & (\,\text{Table}[tb.f/f] \parallel \\
& \quad (\,\text{ConnectingEquation} \parallel \\
& \qquad \text{Pendulum}[p1.f/f] \parallel \\
& \qquad \text{Pendulum}[p2.f/f] \\
& \quad)\downarrow_{p1.f,p2.f} \\
&)\downarrow_{x,tb.f}
\end{aligned}
$$

where Table and Pendulum are the Σ-systems obtained from the classes `Table` and `Pendulum` defined Fig. 2, and ConnectingEquation is the Σ-system obtained from the connecting equation, line 37.

The execution of Algorithm 1 is detailed Fig. 9. The figure displays the term as an acyclic directed graph where each node represents a subterm of term CoupledPendulums, and each edge represents the containment relation. The algorithm starts with basic Σ-systems, and propagates Σ-interfaces in the opposite direction of the edges. Each time the Σ-interface of a term is computed, its corresponding cointerface is stored in the memoization table.

Figure 10 details the downward sweep of Algorithm 2: partial Σ-solutions are propagated down the subterm graph. The end result consists of Σ-solutions for each basic Σ-system instance, that can be used to perform the index reduction of each of these oDAE systems.

6 Implementation and benchmarking

In this section, we address an important implementation aspect of our modular method, then we benchmark our prototype implementation on several examples: a vibrating string model, an industrial-strength thermofluid model of a urban heat network, and a Heat Equation model defined with arrays and a *for* loop.

6.1 Avoiding the combinatorial explosion due to selectors

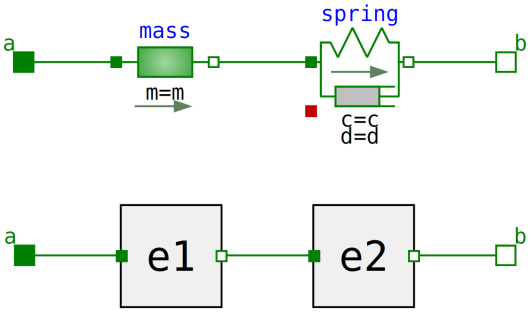
A difficulty in our interface theory is the consideration of selector-dependent structural analysis problems. The primal and dual problems are functions of the selector, that can be numerous, except for systems with only a few public variables. With a mere enumeration of selectors, composition operators can result in an exponential growth of the number of valid selectors.

Indeed, the number of valid selectors usually remains rather small, as the maximum number of shared variables for all the subsystems of a given system is small (below 20 or so). In such cases, enumerating valid selectors is not costly. It is however possible that, for some subsystems, the set of shared variables is larger, and enumerating selectors may become too costly. In Modelica, this typically happens whenever a class has public variables of array types.

In such cases, the set of valid selectors can be represented by its Boolean *characteristic function*. The primal and dual interfaces can also be represented as tuples of Boolean functions In this approach, efficient representations exist, namely Binary Decision Diagrams and their extensions; we use the Reduced-Ordered variant (ROBDD) introduced by (Bryant 1986). In our implementation, we adopted a hybrid approach, where selectors are enumerated for systems with a number of public variables below a given threshold, while the characteristic function approach is used when this threshold is exceeded (Benveniste et al. 2022).

The experimental comparison of the enumerative and symbolic approaches, presented in Section 6.2, turns in

DOI 10.3384/ecp218175




```
import ...;
model HarmonicString
  parameter Integer n = 1 "Number of elements";
  parameter Mass m = 1e-3 "Mass";
  parameter Distance l = 1e-2 "Length";
  parameter TranslationalSpringConstant c = 1;
  parameter TranslationalDampingConstant d = 1e-3;
  parameter Distance s0 = 0 "Initial position";
  Flange_a a;
  Flange_b b;
  static if n > 1 then
  protected
    parameter Integer n1 = n / 2;
    parameter Integer n2 = n - n1;
    HarmonicString s1(n=n1, m=n1*m/n, ...);
    HarmonicString s2(n=n2, m=n2*m/n, ...);
  else
  protected
    Element e(m=m, l=l, c=c, d=d, s0=s0);
  end if;
equation
  ...
end HarmonicString;
```

Figure 3. A mass-spring-damper system; the element (top left) is assembled from the 1-D translational mechanical components of the Modelica Standard Library. An assembly of two elements is shown at the bottom left. A chain of mass-spring-damper elements of length n is defined by the recursive Modelica-like class shown on the right. Although Modelica does not allow for recursive classes, our software prototype allows recursion, provided conditional statements can be evaluated at compile time.

favor of the enumerative approach. Nevertheless, the symbolic approach should not be ruled out, as subsystems with a large number of shared variables may occur, and the enumerative approach would fail on these particular instances.

6.2 Benchmarking

Our implementation has been benchmarked on two academic examples and an industrial strength model. The former are a vibrating string model, and a 1-D finite element model of the Heat Equation; the latter is an industrial-strength model of a urban heat network system. All three models have been made parametric so that one can adjust their size, in terms of numbers of equations and variables. For these examples, the performance of the modular structural analysis is compared with a global reference implementation of the Σ-method.

6.2.1 The Vibrating String example

This example is a lumped model of a vibrating string, consisting in a chain of mass-spring-damper elements. The Modelica diagrams and code of this model are given by Figure 3. The mass-spring-damper element is built using the translational mechanics package of the Modelica Standard Library (MSL). The chain is defined by dichotomy, using a recursive class. , exactly like the chain circuit example.

The performance of the modular method was compared with a classical implementation of the Σ-method, where the primal problem is solved using an implementation of the Hungarian method (Munkres 1957), the dual problem is solved by a fixed point iteration, as advocated in (Pryce 2001). This reference implementation also puts the equations in Block Triangular Form (BTF), based on the computation Strongly Connected Components, using Tarjan's

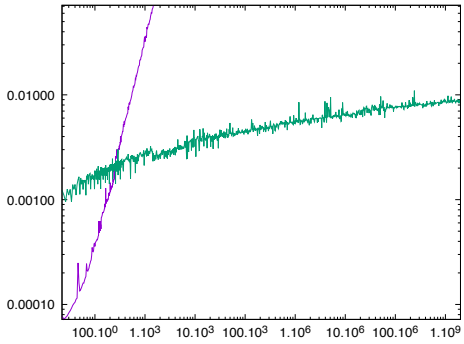


Figure 4. CPU times (seconds) of the modular (in green) and global (in purple) methods for the Vibrating String, as a function of the number of equations.

algorithm (Tarjan 1972). Figure 4 gives the computation times of both methods, for a range of instances of Vibrating String model, up to about 10^9 equations. The computation times for the modular method are logarithmic in the number of equations, because the number of interfaces to be computed is logarithmic in the model parameter value. The modular method runs faster than the global one, except for small models with less than a few hundred equations.

6.2.2 The Urban Heat Network model

This is an industrial-strength thermofluidic model of a urban heat network system, inspired by (Mans et al. 2022). The complete network comprises hot and cold water pipes, heat exchangers and circulation pumps, expansion tanks, and heat sources and sinks. Pipes and heat exchangers are lumped models, consisting in several elements connected in series. Figure 5 shows the variables-to-components incidence graph of the model for parameter value $n = 10$, and

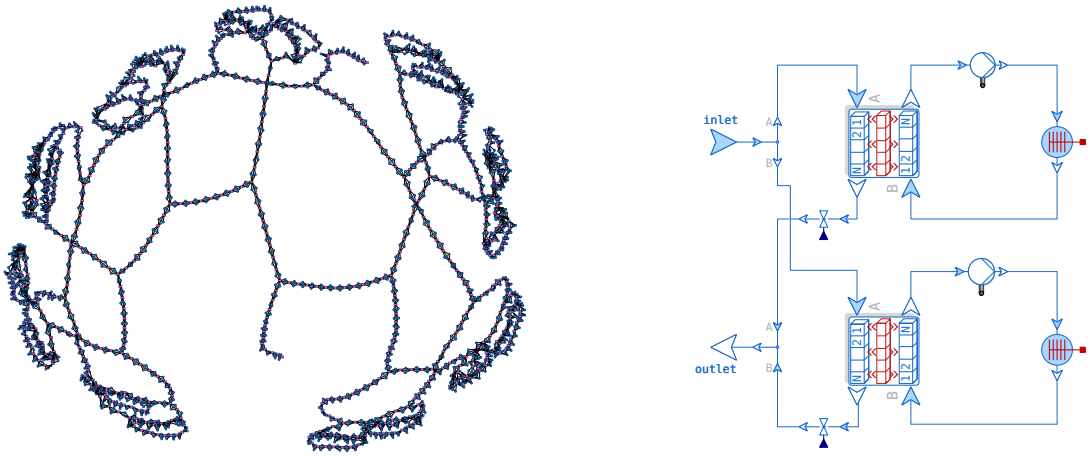


Figure 5. (a) Variables-to-components graph of the Urban Heat Network model for $n = 10$. Blue circles represent variables that are shared between two components; purple boxes represent instances of the basic thermofluidic components. (b) Modelica model of a substation, instantiated n times in the model and interconnected with a network of pipes.

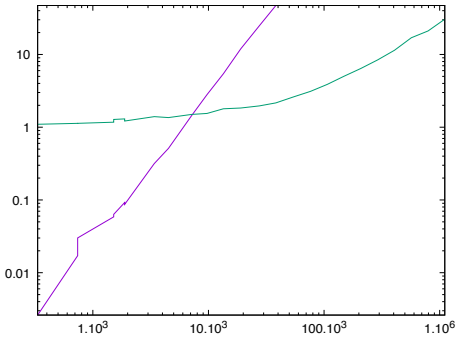


Figure 6. CPU times (seconds) of the modular (in green) and global (in purple) methods, as a function of the number of equations.

```
model Thermal1D
  parameter Integer N = 5;
  parameter Real g = 0.00314785; // W/K
  parameter Real c = 0.2707936; // J/K
  parameter Real Tleft = 400 + 273.15; // K
  parameter Real Tright = 20 + 273.15; // K
  Real T[N](each start=Tright);
equation
  c*der(T[1]) = g * (2*Tleft - 3*T[1] + T[2]);
  for i in 2:N-1 loop
    c*der(T[i]) = g*(T[i-1] - 2*T[i] + T[i+1]);
  end for;
  c*der(T)[N] = g*(T[N-1] - 3*T(N) + 2*Tright);
end Thermal1D;
```

Figure 7. Modelica code of the 1-D Heat Equation, with an array and a loop of equations.

the Modelica model of a substation, that is instantiated n times and connected to a tree-shaped network of hot and cold water pipes. Figure 6 gives the computation times of the global and modular methods for instances of this model up to about 10^6 equations.

Although its measured time complexity is not logarithmic in the size of the model, the modular method runs faster than the global method for instances of the model with more than about $7,000$ equations. This is due to the fact that the model has a low treewidth (Bodlaender 1988), as evidenced by Figure 5a.

6.2.3 The Heat Equation model

Figure 7 shows the Modelica code of a finite element model of the 1D Heat Equation. This model, taken from (Fioravanti et al. 2023), consists of an array of real variables and a set of equations defined thanks to a *for* loop. This is a challenging benchmark for two reasons. First, it is a system of ordinary differential equations, on which each of the three steps of the global structural analysis method (primal and dual problems, then BTF decomposition) has a computation time that is linear in the size of the model. Second, this model does not provide a class instantiation tree decomposition of the system of equations; this decomposition has to be defined manually, or computed automatically.

As a reference, we performed the structural analysis using a manually-defined balanced binary tree decomposition of the system of equations, very similar to that of the Vibrating String example. Unsurprisingly, the performances (Figure 8, top graph) show that, above a certain threshold in the size of the system, the modular structural analysis takes only a fraction of the computation time required for the global (classical) structural analysis. As a matter of fact, it should be noticed that the empirical time complexity is logarithmic in the number of equations.

To our knowledge, computing an almost-balanced tree decomposition, that minimizes the number of shared variables between subtrees, is an open problem, in the general case of (possibly nested) loops of equations. It is nevertheless possible to compute tree decompositions, not necessarily balanced, using the heuristics implemented in Snowflake (Thibault 2024; Thibault 2022). Because

DOI
10.3384/ecp218175

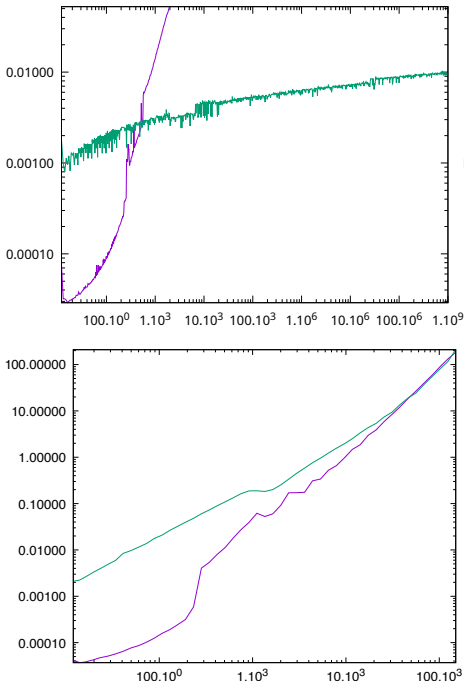


Figure 8. CPU times (seconds) of the modular (in green) and global (in purple) methods for the Heat Equation model, as a function of the number of equations, with balanced (top graph) and unbalanced (bottom graph) tree decompositions.

Snowflake implements a greedy decomposition algorithm, it results in an unbalanced, comb-shaped tree. The performances obtained with this tree decomposition are shown in the bottom graph of Figure 8. Although the modular structural analysis also has a linear time complexity for this model, it can be observed that, for large instances of the model, the computation times of the global and modular methods are almost the same. This examplifies the importance of a balanced tree decomposition for the performance of the modular method.

7 Conclusion and perspectives

Only square and structurally nonsingular DAE models are suitable for simulation. Such models correspond to *closed* systems, considered in isolation. In contrast, *open component models*, possessing more variables than equations, constitute the adequate framework to deal with model composition. Variables in open models are either local (*protected* variables in Modelica, i.e., private to the class), or public (possibly shared with other component models). Open models can be made aware of assumptions on their (unspecified yet) context. Overall, open models address the system view of physical modeling.

In this paper, we propose a notion of Σ-*interface* for open component models. Σ-interfaces involve only shared variables, thus abstracting away equations and local variables. The Σ-interface of an open model carries the information needed to perform modular structural analysis. We propose a hierarchical and modular algorithm that performs the interface-based structural analysis for DAE systems composed of possibly many open component models. Our algorithm exploits the sparsity of the underlying system as reflected in its incidence graph. Furthermore, very much like message passing, its efficiency relates to the model treewidth: the lower, the better.

The modular structural analysis algorithm was benchmarked on two academinc examples and an industrial-strength model of a urban district heating network. We succeeded in performing the structural analysis of DAE systems comprizing from 10^6 to 10^9 equations, depending on the model. Except for small instances of the models, computation times of the modular algorithm were a fraction of the computation times of state-of-the-art global structural analysis algorithms. As expected, sublinear execution times have been measured for several of the use cases.

This Σ-interface theory breaks one of the acknowledged obstacles to the scalability of DAE-based modeling languages, such as Modelica. Our contribution thus provides an important step towards the modular compilation of Modelica.

Of course, some issues remain open. In particular, models with higher treewidth are not handled well by our approach. Such unstructured models typically involve arrays of variables and components, with square regular grids being an example. While our notion of Σ-interface remains a useful candidate abstraction in this case, alternative algorithms for solving the Σ-method in a modular way still have to be found.

References

Benveniste, Albert et al. (2022). “Algorithms for the Structural Analysis of Multimode Modelica Models”. In: *Electronics* 11.17. ISSN: 2079-9292. DOI: 10.3390/electronics11172755. URL: https://www.mdpi.com/2079-9292/11/17/2755.

Benveniste, Albert et al. (2023-10). “Towards the separate compilation of Modelica: modularity and interfaces for the index reduction of incomplete DAE systems”. In: *Linköping Electronic Conference Proceedings*. Vol. 204. Aachen, Germany, p. 10. DOI: 10.3384/ecp204. URL: https://inria.hal.science/hal-04295096.

Bodlaender, Hans L. (1988). “Dynamic programming on graphs with bounded treewidth”. In: *Automata, Languages and Programming*. Ed. by Timo Lepistö and Arto Salomaa. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 105–118. ISBN: 978-3-540-39291-0.

Broman, David (2021). “Interactive Programmatic Modeling”. In: *ACM Trans. Embed. Comput. Syst.* 20.4, 33:1–33:26. DOI: 10.1145/3431387. URL: https://doi.org/10.1145/3431387.

Broman, David and Peter Fritzson (2008). “Higher-Order Acausal Models”. In: *Proceedings of the 2nd International Workshop on Equation-Based Object-Oriented Languages and Tools, EOOLT 2008, Paphos, Cyprus, July 8, 2008*. Ed. by Peter Fritzson, François E. Cellier, and David Broman. Vol. 29. Linköping Electronic Conference Proceedings. Linköping University Electronic Press, pp. 59–69. URL: http://www.ep.liu.se/ecp/article.asp?issue=029%5C&article=007%5C&volume=.

Bryant, R. E. (1986). “Graph-Based Algorithms for Boolean Function Manipulation”. In: *IEEE Transactions on Computers* C-35.8, pp. 677–691. DOI: 10.1109/tc.1986.1676819.

Ⓙ

Y	J	A	Y_1	Y_2
$\{x, tb.f\}$	6	A_{10}	$\{x\}$	$\{tb.f\}$

$$A_{10} = \begin{array}{c} \\ 0 \\ x \\ tb.f \end{array} \begin{array}{c} \begin{array}{ccc} 0 & x & tb.f \end{array} \\ \begin{bmatrix} -2 & 0 \\ 0 & 2 \\ -2 & 0 \end{bmatrix} \end{array}$$

Ⓘ is identical to Ⓗ, after renaming f by $tb.f$.

Ⓗ

Y	J	A
$\{f\}$	0	A_8
$\{x\}$	2	A_9

$$A_8 = \begin{array}{c} \\ 0 \\ x \\ f \end{array} \begin{array}{c} \begin{array}{ccc} 0 & x & f \end{array} \\ \begin{bmatrix} -2 & 0 \\ & \\ -2 & 0 \end{bmatrix} \end{array} \qquad A_9 = \begin{array}{c} \\ 0 \\ x \\ f \end{array} \begin{array}{c} \begin{array}{cc} 0 & x \end{array} \\ \begin{bmatrix} -2 & 0 \\ 0 & 2 \\ & \end{bmatrix} \end{array}$$

Ⓖ

Y	J	A
$\{tb.f\}$	4	A_6
$\{x\}$	6	A_7

$$A_6 = \begin{array}{c} \\ 0 \\ x \\ tb.f \end{array} \begin{array}{c} \begin{array}{ccc} 0 & x & tb.f \end{array} \\ \begin{bmatrix} -2 & 0 \\ -2 & 0 \\ & \end{bmatrix} \end{array} \qquad A_7 = \begin{array}{c} \\ 0 \\ x \\ tb.f \end{array} \begin{array}{c} \begin{array}{ccc} 0 & x & tb.f \end{array} \\ \begin{bmatrix} -2 & 0 \\ & \\ & 2 \end{bmatrix} \end{array}$$

Ⓔ and Ⓕ are given Fig. 10.

Ⓓ

Y	J	A	Y_1	Y_2
$\{p1.f, p2.f\}$	4	A_3	$\{p1.f\}$	$\{p2.f\}$
$\{x, p1.f\}$	6	A_4	$\{p1.f\}$	$\{x\}$
$\{x, p2.f\}$	6	A_5	$\{x\}$	$\{p2.f\}$

$$A_3 = \begin{array}{c} \\ 0 \\ x \\ p1.f \\ p2.f \end{array} \begin{array}{c} \begin{array}{cccc} 0 & x & p1.f & p2.f \end{array} \\ \begin{bmatrix} -2 & 0 & 0 \\ & & & \\ -2 & 0 & \\ -2 & & 0 \end{bmatrix} \end{array}$$

$$A_4 = \begin{array}{c} \\ 0 \\ x \\ p1.f \\ p2.f \end{array} \begin{array}{c} \begin{array}{cccc} 0 & x & p1.f & p2.f \end{array} \\ \begin{bmatrix} -2 & 0 & 0 \\ 0 & & 2 \\ -2 & 0 & \\ & & \end{bmatrix} \end{array}$$

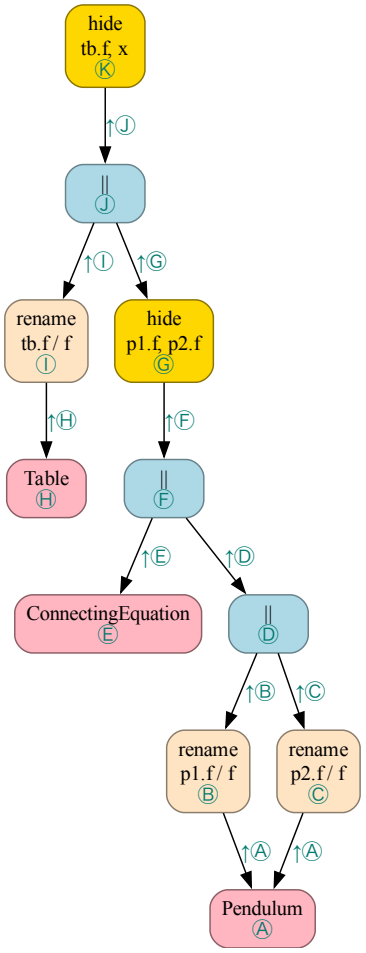
$$A_5 = \begin{array}{c} \\ 0 \\ x \\ p1.f \\ p2.f \end{array} \begin{array}{c} \begin{array}{cccc} 0 & x & p1.f & p2.f \end{array} \\ \begin{bmatrix} -2 & 0 & 0 \\ 0 & 2 & \\ & & \\ -2 & & 0 \end{bmatrix} \end{array}$$

Ⓚ

Y	J	A
$\emptyset$	6	A_{11}

$$A_{11} = \begin{array}{c} \\ 0 \end{array} \begin{array}{c} \begin{array}{c} 0 \end{array} \\ \begin{bmatrix} \end{bmatrix} \end{array}$$

Term Ⓚ has no public variable. It admits one valid selector: the empty set. This means that the Coupled-Pendulum model is structurally nonsingular.



Ⓑ (resp. Ⓒ) is identical to Ⓐ, up to a renaming of f by $p1.f$ (resp. $p2.f$).

Ⓐ

Y	J	A
$\{f\}$	2	A_1
$\{x\}$	4	A_2

$$A_1 = \begin{array}{c} \\ 0 \\ x \\ f \end{array} \begin{array}{c} \begin{array}{ccc} 0 & x & f \end{array} \\ \begin{bmatrix} -2 & 0 \\ & \\ -2 & 0 \end{bmatrix} \end{array} \qquad A_2 = \begin{array}{c} \\ 0 \\ x \\ f \end{array} \begin{array}{c} \begin{array}{ccc} 0 & x & f \end{array} \\ \begin{bmatrix} 0 & 0 \\ 0 & 2 \\ & \end{bmatrix} \end{array}$$

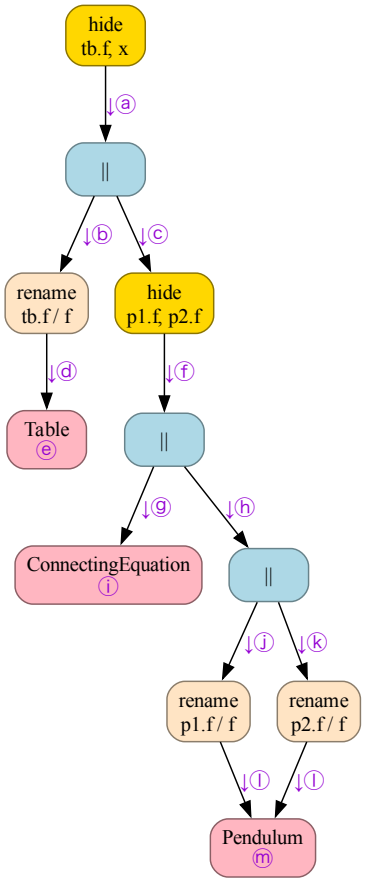
Figure 9. Primal/dual interfaces and primal cointerfaces computed during the upward sweep. Start reading the figure from the bottom. For the sake of clarity, the BTF components of the interfaces/cointerfaces have been omitted.

DOI 10.3384/ecp218175

$$A_{14} = \begin{array}{c} \\ 0 \\ tb.f \\ p1.f \\ p2.f \\ x \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & tb.f & p1.f & p2.f & x \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & -2 \\ 0 & 0 & 0 & \\ & 0 & & -2 \\ & & 0 & -2 \\ & & & \end{bmatrix} \end{array} \quad A_{15} = \begin{array}{c} \\ 0 \\ tb.f \\ p1.f \\ p2.f \\ x \end{array} \begin{array}{c} \begin{array}{cccc} 0 & tb.f & p1.f & p2.f & x \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & -2 \\ 0 & 0 & 0 & \\ & 0 & & -2 \\ & & & \\ & & 2 & 0 \end{bmatrix} \end{array}$$

Ⓕ

Y	J	A
$\{tb.f, p1.f, p2.f\}$	4	A_{14}
$\{tb.f, p1.f, x\}$	6	A_{15}
$\{tb.f, p2.f, x\}$	6	A_{16}
$\{p1.f, p2.f, x\}$	6	A_{17}



Ⓔ

Y	J	A
$\{tb.f\}$	0	A_{11}
$\{p1.f\}$	0	A_{12}
$\{p2.f\}$	0	A_{13}

$$A_{11} = \begin{array}{c} \\ 0 \\ tb.f \\ p1.f \\ p2.f \end{array} \begin{array}{c} \begin{array}{ccc} 0 & tb.f & p1.f & p2.f \end{array} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ & & \\ & & \end{bmatrix} \end{array}$$

A_{16} is obtained from A_{15} by exchanging rows and columns $p1.f$ and $p2.f$. A_{17} is obtained from A_{15} by exchanging rows $tb.f$ and $p1.f$, then columns $p1.f$ and $p2.f$.

ⓐ $= \left(\{x, tb.f\}, < d_x = 2, d_{tb.f} = 0 >\right)$ is computed using the cointerface of term ⑪, retrieved from the memoization table.

ⓑ $= \left(\{x\}, < d_x = 2, d_{tb.f} = 0 >\right)$ is computed from ⓐ, using the cointerface of term ⑩.

ⓒ $= \left(\{tb.f\}, < d_x = 2, d_{tb.f} = 0 >\right)$.

ⓓ $= \left(\{x\}, < d_x = 2, d_f = 0 >\right)$ is the reverse renaming of ⓑ.

ⓔ $= \left(\{x\}, < d_x = 2, d_f = 0 >, < c_8 = 0 >\right)$ is the Σ-solution for the instance of model `Table` (Fig. 2). c_8 is the offset of the equation line 8. This means that this equation is not differentiated.

ⓕ $= \left(\{tb.f, p1.f p2.f\}, < d_x = 2, d_{tb.f} = d_{p1.f} = d_{p2.f} = 0 >\right)$.

ⓖ $= \left(\{tb.f\}, < d_{tb.f} = d_{p1.f} = d_{p2.f} = 0 >\right)$.

ⓗ $= \left(\{p1.f, p2.f\}, < d_x = 2, d_{p1.f} = d_{p2.f} = 0 >\right)$.

ⓘ $= \left(\{tb.f\}, < d_{tb.f} = d_{p1.f} = d_{p2.f} = 0 >, < c_{37} = 0 >\right)$. Equation line 37 does not need to be differentiated.

ⓙ $= \left(\{p1.f\}, < d_x = 2, d_{p1.f} = 0 >\right)$.

ⓚ $= \left(\{p2.f\}, < d_x = 2, d_{p2.f} = 0 >\right)$.

ⓛ $= \left(\{f\}, < d_x = 2, d_f = 0 >\right)$.

ⓜ $= \left(\{f\}, < d_x = d_a = d_b = 2, d_f = d_{lambda} = 0, >, < c_{28} = 2, c_{24} = c_{25} = c_{29} = 0 >\right)$. Both instances of model `Pendulum` appear in the context of the same partial solution ①. This model has to undergo only one index reduction, where the only equation to be differentiated is line 28.

A_{12} (resp. A_{13}) is obtained from A_{11} by exchanging rows $tb.f$ and $p1.f$ (resp. $p2.f$).

Figure 10. Partial solutions computed during the downward sweep. Start reading the figure from the top. For the sake of clarity, the BTF components of the partial solutions have been omitted.

Cormen, Thomas H. et al. (2022). *Introduction to Algorithms, 4th Edition*. The MIT Press. ISBN: 9780262046305. URL: https://mitpress.mit.edu/9780262046305/introduction-to-algorithms/.

Dill, David L. (1989). "Timing Assumptions and Verification of Finite-State Concurrent Systems". In: *Automatic Verification Methods for Finite State Systems, International Workshop, Grenoble, France, June 12-14, 1989, Proceedings*. Ed. by Joseph Sifakis. Vol. 407. Lecture Notes in Computer Science. Springer, pp. 197–212. DOI: 10.1007/3-540-52148-8_17. URL: https://doi.org/10.1007/3-540-52148-8%5C_17.

Fioravanti, Massimo et al. (2023-09). "Array-Aware Matching: Taming the Complexity of Large-Scale Simulation Models". In: *ACM Trans. Math. Softw.* 49.3. ISSN: 0098-3500. DOI: 10.1145/3611661. URL: https://doi.org/10.1145/3611661.

Fritzson, Peter (2015). *Principles of Object-Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach*. 2nd ed. IEEE Press. Hoboken, NJ: Wiley. ISBN: 978-1-118-85912-4. DOI: 10.1002/9781118989166.

Höger, Christoph (2015). "Faster Structural Analysis of Differential-Algebraic Equations by Graph Compression". In: *IFAC-PapersOnLine* 48.1. 8th Vienna International Conference on Mathematical Modelling, pp. 135–140. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2015.05.100. URL: https://www.sciencedirect.com/science/article/pii/S2405896315001019.

Mans, Michael et al. (2022). "Development and Application of an Open-Source Framework for Automated Thermal Network Generation and Simulations in Modelica". In: *Energies* 15.12. ISSN: 1996-1073. DOI: 10.3390/en15124372. URL: https://www.mdpi.com/1996-1073/15/12/4372.

Miné, Antoine (2001). "A New Numerical Abstract Domain Based on Difference-Bound Matrices". In: *Programs as Data Objects, Second Symposium, PADO 2001, Aarhus, Denmark, May 21-23, 2001, Proceedings*. Ed. by Olivier Danvy and Andrzej Filinski. Vol. 2053. Lecture Notes in Computer Science. Springer, pp. 155–172. DOI: 10.1007/3-540-44978-7_10. URL: https://doi.org/10.1007/3-540-44978-7%5C_10.

Munkres, James (1957). "Algorithms for the Assignment and Transportation Problems". In: *Journal of the Society for Industrial and Applied Mathematics* 5.1, pp. 32–38. DOI: 10.1137/0105003.

Otter, Martin and Hilding Elmqvist (2017-05). "Transformation of Differential Algebraic Array Equations to Index One Form". In: *Proceedings of the 12th International Modelica Conference*. Prag, Tschechische Republik: Linköping University Electronic Press. DOI: 10.3384/ecp17132565.

Pantelides, C. C. (1988). "The Consistent Initialization of Differential-Algebraic Systems". In: *SIAM Journal on Scientific and Statistical Computing* 9.2, pp. 213–231. DOI: 10.1137/0909014.

Pop, Adrian et al. (2019). "A New OpenModelica Compiler High Performance Frontend". In: *Proceedings of the 13th International Modelica Conference, Regensburg, Germany, March 4-6, 2019*, 157:071. DOI: 10.3384/ecp19157689.

Pryce, J. D. (2001). "A Simple Structural Analysis Method for DAEs". In: *BIT Numerical Mathematics* 41.2, pp. 364–394. DOI: 10.1023/a:1021998624799.

Tarjan, Robert (1972). "Depth-First Search and Linear Graph Algorithms". In: *SIAM Journal on Computing* 1.2, pp. 146–160. DOI: 10.1137/0201010.

Thibault, Joan (2022-07). *Constraint System Decomposition*. Research Report RR-9478. Inria Rennes, pp. 1–68. DOI: 10.13140/RG.2.2.13004.49285. URL: https://hal.inria.fr/hal-03740562.

Thibault, Joan (2024). *Snowflake*. https://gitlab.com/boreal-ldd/snowflake [Accessed: 29-04-2025].

Resizable Arrays in Object-Oriented Modeling

Martin Otter[1] Hilding Elmqvist[2]

[1]DLR, Institute of Vehicle Concepts, Germany, martin.otter@dlr.de

[2]Mogram AB, Sweden, hilding.elmqvist@mogram.net

Abstract

The Modelica language (Modelica.org) makes it easy to build large, complex models by allowing the instantiation of reusable component models. Modelica tools typically expand arrays of variables, equations and components and perform symbolic transformations on the scalar elements. This reduces the efficiency of the translation process and makes it difficult to change array dimensions after translation.

This paper describes modest enhancements of standard algorithms to avoid scalarization. As a result, arrays can be resized both after translation and during simulation. The new technique does, however, impose certain restrictions on the way models are written. It is also sketched how to provide more meaningful diagnostics for erroneous models. Several examples demonstrate the new algorithms using the Web App Modiator.

Keywords: Modelica, array equations, compilation

1 Introduction

The Modelica language[1] makes it easy to build large, complex models by allowing the instantiation of reusable component models. Modelica tools typically expand arrays of variables, equations and components and perform symbolic transformations on the scalar elements. This reduces the efficiency of the translation process especially for models with fine discretization of partial differential equations or for handling arrays of parametric data. In many cases, this scalarization can be omitted, which will be explored in this paper.

Furthermore, avoiding scalarization of arrays allows changing array sizes without recompilation. This is important since it's essential to check the fidelity of the discretization for partial differential equations to ensure sufficient accuracy, i.e. to allow quick comparison of simulation results without recompilation with different numbers of segments. The technique also enables changing array dimensions during simulation, i.e., a model modifies its discretization depending on the model's behavior, for example increasing the number of segments when a transient occurs.

This paper outlines the modest enhancements to standard algorithms required for this new approach. A tool can verify whether the model complies with the required restrictions. If so, the new technique is employed, otherwise the current scalarization approach is used. The new approach imposes certain restrictions on models. These restrictions and how to rewrite the model equations to fulfill them are described. The alternative is to scalarize the equations internally, and then try to combine them into array equations and for-loops after the symbolic algorithms have been applied. Such an approach is much more involved.

The new approach and all the examples in this article have been tested with the Web App *Modiator* which supports a subset of Modelica. The current features, restrictions and future plans of Modiator are discussed in the companion paper (Elmqvist and Otter 2025).

Previous work on avoiding scalarization includes the following articles: (Otter and Elmqvist 2017) show that scalarization of array equations is needed for index reduction when using the standard algorithms, e.g., for multibody systems. They scalarize array equations internally and reconstruct arrays after sorting. (Pop et al. 2019) describe a new high-performance frontend to OpenModelica[2] (Fritzson et al. 2020) to convert arrays of component models to array equations and to keep arrays during flattening. (Zimmermann et al. 2020) propose set-based graph algorithms to avoid unrolling of for-loop equations of Modelica models. (Abdelhak et al. 2023,2025) map indices of array variables and equations, as well as iterator values of for-equations to unique scalar indices, apply standard algorithms for matching and sorting on these internally scalarized systems and reconstruct array equations and for-loops afterwards. They tested this approach with an extension to OpenModelica. (Marzorali et al. 2024) propose a matching algorithm for repetitive structures with for-loops over vectors of unknowns and test the implementation as part of the ModelicaCC compiler[3].

2 Model Restrictions

This section explores the restrictions that are needed to be imposed on Modelica authoring to avoid scalarization with the new method and achieve compile times that are independent of array sizes (*O(1) rather than O(n) or worse*).

Restriction 1: *The model must be balanced with regards to non-scalarized variables and equations. The equation*

[1] https://specification.modelica.org/maint/3.6/MLS.html

[2] https://openmodelica.org/

[3] https://github.com/CIFASIS/modelicacc

number of a for loop is the number of equations in the loop.

This restriction is illustrated using the CascadedFirstOrder model from the ScalableTestSuite[4] (Casella 2015):

```
tau*der(x[1]) = u - x[1];
for i in 2:N loop
  tau*der(x[i]) = x[i-1] - x[i];
end for;
```

Equations for tau and u are provided elsewhere. This equation set violates **Restriction 1** and needs to be modified since the number of unknowns is one (der(x)) but there are two equations. It can, for example, be rewritten as:

```
for i in 1:N loop
  tau*der(x[i]) = (if i == 1 then u else x[i-1]) - x[i];
end for;
```

or more compactly using the concatenation operator cat:

```
tau*der(x) = cat(1, {u}, x[1:N-1]) - x;
```

The cat function concatenates the array arguments, in this case along the first dimension. It means that the x vector is shifted upwards (x[N] omitted) and u is inserted as first element. Yet another possibility is to use a reduction expression.

```
tau*der(x) = {(if i == 1 then u else x[i-1]) - x[i] for i in 1:N}
```

Restriction 2: *All elements of an array must be of the same kind (algebraic or differentiated).*

This means that a model such as a variant of the CascadedFirstOrder model from above:

```
x[1] = u;
for i in 2:N loop
  tau*der(x[i]) = x[i-1] - x[i];
end for;
```

violates **Restriction 2** because x[1] is an algebraic variable and variables x[2:N] are variables appearing differentiated. Such a model needs to be rewritten in one of the forms shown above. In similar cases, arrays need to be split in different pieces so that all elements of an array are either algebraic or states.

An equivalent model to the CascadedFirstOrder model can be made with an array of components:

```
model FiltersInSeries
  model Filter
    input Real u;
    output Real x;
    parameter Real tau = 0.1;
  equation
    tau*der(x) = u-x;
  end Filter;
  parameter Integer N = 10 "Number of filters";
  Filter f[N](each x(start=0, fixed=true),
              u=cat(1, {1}, f[1:N-1].x));
end FiltersInSeries;
```

If all components in an array of components have the same causality, all component equations can be sorted together, i.e. that each equation in an array of components can be translated to a for loop over that equation (it is not important in this simple case since there is only one equation in each component).

The outputs f.x[1:N-1] are shifted to form the inputs f.u together with the input 1 to the first filter. Handling acausal connectors is somewhat more complex. See the *Transmission Line* example in section 4.3.

In section 5 more *restrictions* are discussed and there might be still more. Modiator never performs scalarization (also not internally) and therefore *valid Modelica models might be rejected.* The user should be able to understand how to rewrite the model equations from the error messages. The benefit is that generated code never contains scalarized equations or enrolled for-loops, translation is faster and error messages are more compact since the elements of an array are not shown.

3 Assignment without Scalarization

This section gives an overview how to symbolically process array equations so that arrays can be resized after compilation - both for translated Modelica models and Functional Mockup Units (≥ version 3.0)[5]. The details of the new algorithms are presented in **Appendix A**.

3.1 Model Unification

In order to make symbolic processing simpler, the Modelica model is transformed by a model unification process similar to what is done in OpenModelica (Pop et al. 2019):

- For-equations with multiple equations in the body are converted to potentially nested for-equations with *one* equation in the body:
  ```
  for index-expression, … loop
    equation
  end for;
  ```
- Non-constant variables of arrays of components are transformed to array variables with outer indices according to the component array indices.
- Each equation of an array of components is transformed to a for-loop over the equation.

3.2 Structurally Assignable Property

Afterwards, standard symbolic algorithms are used that operate, with some modifications on variable symbols. The following notation is used:

- All variable symbols v_j are collected in a variable vector $\boldsymbol{v}$. A symbol v_j may represent a scalar or an array of one or more dimensions. The first variable v_0 has index $j = 0$ (as it is common in programming languages such as C, C++, Javascript).

[4] https://github.com/casella/ScalableTestSuite

[5] https://fmi-standard.org/docs/3.0.2/

DOI 10.3384/ecp218189

- All equations e_i are collected in an equation vector $\boldsymbol{e}$. An equation e_i may be a scalar or an array equation of one or more dimensions with potentially a for-loop around it, as sketched above. The first equation e_0 has index $i = 0$.
- The relationship between the equations e_i and the variables v_j is defined by a sparse representation of the incidence matrix that is defined as vector $\boldsymbol{G}$, where each entry G_i is a vector consisting of the indices of the variable symbols appearing in equation e_i. $\boldsymbol{G}$ can also be interpreted as a representation of a bi-partite graph.
- The relationship between variable symbols is defined by the variable association vector $\boldsymbol{A}$:
  $A_j = \textbf{if } \dot{v}_j = v_k \textbf{ then } k \textbf{ else } -1.$
- The relationship between the equations is defined by the equation association vector $\boldsymbol{B}$:
  $B_i = \textbf{if } \dot{e}_i = e_k \textbf{ then } k \textbf{ else } -1.$

Additionally, the following *new property* needs to be defined for every variable in every equation:

Definition 1: *A variable v_j is called structurally assignable with respect to an equation e_i, if a complete assignment is possible for all (scalar) elements of v_j with respect to all (scalar) elements of e_i, provided v_j and e_i are expanded to scalars and this property holds, independently of the common dimensions and dimension sizes of v_j and e_i.*

In general, **Definition 2** is difficult to apply. The following sufficient condition is easier to use:

Theorem 1: *A variable v_j is structurally assignable with respect to an equation e_i, if*

- *v_j is a scalar and e_i a scalar equation, or if*
- *v_j is an array and e_i can be expressed as $0 = s \cdot v_j + w$, where s is a scalar expression and w is a vector expression and neither s nor w are a function of v_j.*

Proof: (a) A scalar variable/equation is already expanded and therefore the scalar variable can be assigned. (b) If e_i is expressed as $0 = s \cdot v_j + w$, the k-th element of v_j can be assigned to the k-th element of e_i. ■

The (new) structural assignable property is defined with vector $\boldsymbol{G_a}$, a companion vector of $\boldsymbol{G}$, such that $G_{a,i,j} =$ true, if variable $v_{G_{i,j}}$ is structurally assignable with respect to equation e_i and otherwise $G_{a,i,j} =$ false. Note, every vector $G_{a,i}$ must have at least one true value, since otherwise no variable can be assigned for equation e_i.

The underlying DAE (Differential Algebraic Equations) of a model is defined as:

$$\boldsymbol{e}(\boldsymbol{v}(t), t) \tag{1}$$

where t is time, the independent variable.

The central goal in this section is to provide an algorithm, so that by differentiating equations and variables, appending them to $\boldsymbol{e}$ and $\boldsymbol{v}$, and updating $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{G}, \boldsymbol{G_a}$, a *complete assignment* of the *structurally assignable* highest derivative variables is provided for all highest derivative equations, that is, every highest derivative equation is uniquely assigned to a structurally assignable highest derivative variable:

$$e_i(v_j, t);\ \frac{\partial e_i}{\partial v_k} \text{ is structurally regular for } (B_i = -1,\ A_k = -1, G_{a,i,v_k} = \text{true}) \tag{2}$$

The result of the assignment is reported in vector ***assign***, such that $assign_k = i$, where i is the index of a highest derivative equation, that is $B_i = -1$, and v_k is a structurally assignable highest derivative variable, i.e., $A_k = -1$, and $G_{a,i,v_k} =$ true ($,v_k$ is the index of variable k in $G_{a,i}$). If all variables and equations are scalarized, so $G_{a,i,v_k} = true$ for all the scalarized variables and equations, then there are several (standard) algorithms to derive (2) from DAE (1), see for example (Otter, Elmqvist 2017, section 3.1).

The notation above is clarified with the cascaded first order model in vector notation from section 2, where the equation u = 1 is additionally added:

Table 1: Notation for vectorized CascadedFirstOrder model (T means true, F means False).

e_0	u = 1
e_1	tau*der(x) = cat(1, {u}, x[1:N-1]) - x
$\boldsymbol{B}$	[-1,-1] // no equation appears differentiated
v_0	u
v_1	x
v_2	der(x)
$\boldsymbol{A}$	[-1,2,-1] // the derivative of x is der(x)
G_0	[0] // e_0 is a function of u
G_1	[0,1,2] // e_1 is a function of u, x, der(x)
$G_{a,0}$	[T] // e_0 is structurally assignable for u
$G_{a,1}$	[F, F, T] // e_1 is struct. assignable for der(x)

Using the algorithm given in **Appendix A.1**, a complete assignment (2) for the problem formulation in **Table 1** is derived, yielding the following result:

assign = [0, -1, 1] // v_0 assigned to e_0, v_2 to e_1

This assignment holds for any valid dimension N (for N=1, the model is not valid due to x[1:N-1]). With the algorithms sketched in **Appendix A.4**, the following sorted and solved assignment statements can be derived:

```
u := 1
der(x) := (cat(1, {u}, x[1:N-1]) – x)/tau
```

and corresponding code generated. The translation time is independent of dimension N and N can be changed after translation!

3.3 Differentiation of Equations

In order to derive (2) from DAE (1), equations might need to be differentiated. For scalarized variables and equations several (standard) algorithms are available, see for example (Otter, Elmqvist 2017, section 3.1). These algorithms need to be extended to take the new structurally assignable property $\boldsymbol{G}_a$ into account. In **Appendix A.1**, it is shown how to extend the algorithm of Pantelides (1988) in this respect.

Consider the following simple yet non-trivial example:

Model 1: Model that needs to be differentiated.

```
Model ModelThatNeedsToBeDifferentiated
  Real n[3]; L, e[3]; r[3];
  Real s(start=1, fixed=true);
equation
    L = sqrt(n*n);
    e = 2*n/L;
    r = e*s;
    e*der(r) = der(s) - s;
    n = {1, 2, sin(time)};
end ModelThatNeedsToBeDifferentiated;
```

This model has the following structurally assignable properties:

equations	*structurally assignable*
L = sqrt(n*n);	L
e = 2*n/L;	e, n
r = e*s;	r, s
e*der(r) = der(s) - s;	der(s)
n = {1, 2, sin(time)};	n

With the enhanced Pantelides algorithm of **Appendix A.1** the following complete assignment of differentiated equations can be derived:

assigned	*highest derivative equations*
der(L)	der(L) = 0.5*(n*n)^-0.5*(der(n)*n + n* der(n))
der(e)	der(e) = (2* der(n)*L - 2*n* der(L))/L^2
der(r)	der(r) = der(e)*s + e* der(s)
der(s)	e*der(r) = der(s) - s
der(n)	n = {0,0, cos(time)}

In section **4**, several application examples are given that demonstrate the usefulness of the new approach.

3.4 Improved Error Diagnostics

If a Modelica model is erroneous, in many cases tools provide error messages that make it difficult to figure out the reason for the problem. For example, Modelica tools could provide an error message of the following kind which is quite useless: *The problem is structurally singular. It has 2046 scalar unknowns and 2045 scalar equations*. The error messages have improved over time, but they are often still not helpful enough. In **Appendix A.3**, a new, simple algorithm is sketched to significantly improve error diagnostics. Take for example the following model which contains several errors:

Model 2. Model having several errors.

```
model SeveralErrors
  Real v1,v2,v3,v4,v5,v6,v7,v8,v9,v10,v11;
equation
  der(v1) = -v1;
  der(v2) = -v3;
  v3 = -1;
  v3 = 3 + v2;

  der(v4) = -v4 + v1;
  der(v5) = -v5 - v4;
  v5 = 2;

  der(v6) = -v6 + v7;
  der(v7) = -v8 + v9;
  0 = v7 + v8 + v9;
end SeveralErrors;
```

Modiator provides the following error message.

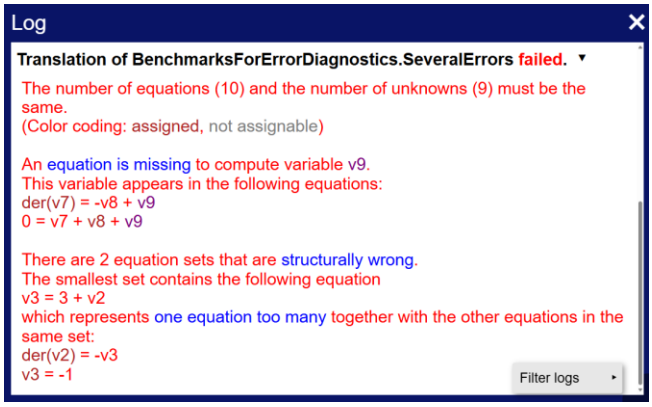


Figure 1. Modiator error message for **Model 2**.

The first message states, that one equation is missing to compute variable v9 and the equations are listed in which this variable appears. All of these equations are already assigned, so it is impossible to assign for v9. Most likely, just an equation for v9 is missing and should be added. Modiator lists these types of error messages first, because little output is expected, since a variable typically only appears in a few equations.

The second message states that two equation sets are wrong. Since many equations might be involved, more information is only given for the set with the smallest number of equations. For this set, first the equation is listed that makes the following set of equations overdetermined. This equation cannot get an assignment because there is one equation too many in this set. Afterwards, all the other (already assigned) equations are listed together with their assignment.

Note, the output of the error messages is not unique and a different initial ordering of the equations can display a different message (e.g. instead of v9, variable v8 could be listed). However, the important point is that the error message either points to *one* problematic *variable* or to *one* problematic *equation*, to help the user identifying the issue.

DOI
10.3384/ecp218189

4 Application Examples

4.1 Heat Exchanger

The ScalableTestSuite and LargeTestSuite[6] of (Casella, 2015) contain cocurrent heat exchanger models with different numbers of segments from 10 to 81920. The compilation time in Modiator is essentially independent of n: about 11 milliseconds. The simulation results for n=10 are shown in Figure 1, and n can be changed after compilation.

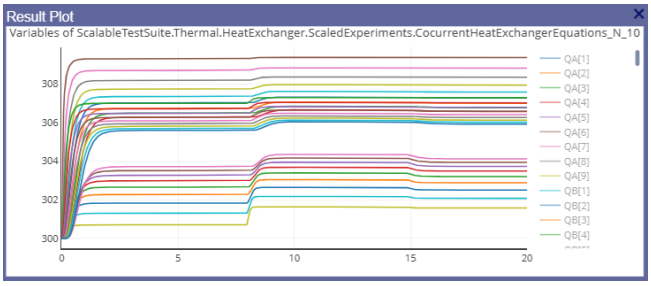


Figure 2. Simulation results for the cocurrent heat exchanger model with 10 segments from the ScalableTestSuite.

A small modification of the model was needed. The equations:

```
TA[1] = if time < 8 then 300 else 301;
TB[1] = 310;
for i in 2:N loop
  TA[i] = TAtilde[i - 1];
  TB[i] = TBtilde[i - 1];
end for;
```

were replaced by:

```
TA = cat(1, {if time < 8 then 300 else 301}, TAtilde)
TB = cat(1, {310}, TBtilde)
```

in order to comply with **Restriction 1**.

4.2 Shock Waves

Capturing shock waves typically needs a large number of grid points, i.e. benefits substantially by using the described array preserving technique. Consider the water hammer model, **Model 3**, using only array equations. The valve at the end of the pipe closes at time=0.01 inducing a shock wave. The number of grid points n can be changed after compilation.

Model 3: Modelica Water Hammer model capturing shock waves from Clément Coïc.

```
model WaterHammer
  parameter Integer n=10;
  parameter Real L=100 "Length of the pipe (m)";
  parameter Real A=3.1415*D^2/4 "Cross-sect. area of pipe (m^2)";
  parameter Real rho=1000 "Density of water (kg/m^3)";
  parameter Real B=21e8 "Bulk modulus of water (Pa)";
  final parameter Real c=sqrt(B/rho) "Speed of sound in water (m/s)";
  parameter Real tClose=0.01 "Time at which the valve closes (s)";
  parameter Real Q0=0.0002 "Initial flow rate (m^3/s)";
  final parameter Real dx=L / n "Length of each pipe segment (m)";
  final parameter Real dt=dx / c "Time step for stability (s)";
  parameter Real f=0.05 "Darcy-Weisbach friction factor (dim.less)";
  parameter Real D=0.05 "Pipe diameter (m)";
  Real p[n+1](start=fill(3e5, n+1), fixed=true) "Press. at each node (Pa)";
  Real Q[n] "Flow rate in each segment (m^3/s)";
  Real V[n](start=fill(0.1, n), fixed=true) "Speed of flow (m/s)";
  Real Qend "Flow rate at the end of the pipe (m^3/s)";
  Real QPlusEnds[n+2];
  Real valveOpen "1 if valve is open, 0 if closed";
equation
  Q = V*A;
  valveOpen = if time < tClose then 1 else 0;
  Qend = Q0 * valveOpen;
  der(V) = -(p[2:n+1] - p[1:n]) / (rho * dx) - (f / (2 * D)) .* V .* abs(V);
  QPlusEnds = cat(1, {Q[1]}, Q, {Qend});
  der(p) = B*(-(QPlusEnds[2:n+2] - QPlusEnds[1:n+1]) / (A * dx));
 annotation(experiment(StopTime=0.3));
end WaterHammer;
```

Figure 3 plots the pressures against time.

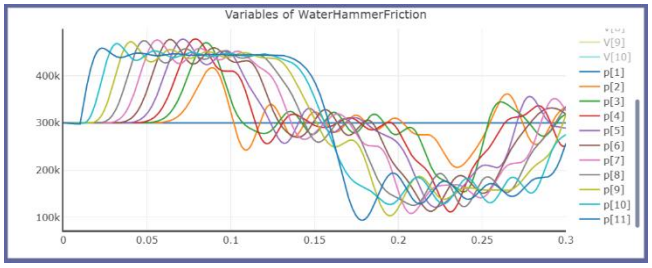


Figure 3. Pressures of WaterHammer model.

Such time plots do not convey the wave behavior very well. It is better to show the pressure profile over the pipe at different times, **Figure 4**. Spline interpolation is used to smooth the curve over the 51 spatial points (n=50). The pressure wave traverses from right to left. After about 72 milliseconds, the wave starts returning (green curve):

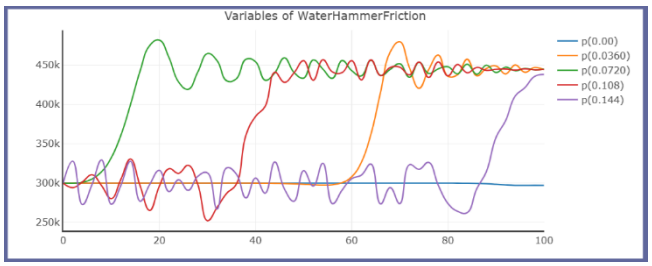


Figure 4. Spatial plot of pressures of WaterHammer model.

4.3 Transmission Line

Consider the transmission line element of **Figure 5**. It is a regular structure of connected electrical components (Resistor, Inductor, Conductor and Capacitor). Such a device can be modeled in Modelica using arrays of components and a for-loop for connections. A voltage source is connected to the first segment and a resistance load to the last segment:

[6] https://github.com/casella/ScalableTestSuite

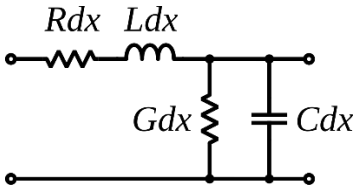


Figure 5. Image of transmission line element from Wikimedia[7]

Model 4: TransmissionLine
modeled with arrays of components.

```
model TransmissionLine
  import Modelica.Electrical.Analog.Basic;
  model TransmissionLineSegment
    parameter Real dx = 1 "Length of the segment in meters";
    parameter Real resistancePerMeter = 0.01;
    parameter Real inductancePerMeter = 0.001;
    parameter Real capacitancePerMeter = 1e-9;
    parameter Real conductancePerMeter = 1e-6;

    Modelica.Electrical.Analog.Interfaces.Pin p;
    Modelica.Electrical.Analog.Interfaces.Pin n;

    Basic.Resistor r(R=resistancePerMeter*dx);
    Basic.Inductor l(L=inductancePerMeter*dx);
    Basic.Capacitor c(C=capacitancePerMeter*dx);
    Basic.Conductor g(G=conductancePerMeter*dx);
    Basic.Ground ground;
  equation
    connect(p, r.p);
    connect(l.n, g.p);
    connect(g.p, c.p);
    connect(c.p, n);
    connect(r.n, l.p);
    connect(g.n, c.n);
    connect(c.n, ground.p);
  end TransmissionLineSegment;

  parameter Integer N = 10 "Number of segments";
  parameter Real length = 1000 "Length of in meters";

  TransmissionLineSegment segments[N](dx=fill(length/N, N));

  Basic.Ground ground;
  Basic.Resistor load(R=100);
  Modelica.Electrical.Analog.Sources.ConstantVoltage
      source(V=10);
equation
  // Connect the segments in series
  for i in 1:N-1 loop
    connect(segments[i].n, segments[i+1].p);
  end for;

  // Connect the first segment to the source
  connect(source.p, segments[1].p);
  connect(source.n, ground.p);

  // Connect the last segment to the load
  connect(segments[N].n, load.p);
  connect(load.n, ground.p);
end TransmissionLine;
```

In order to avoid recompilation of this model when N is changed, the connect statements are manually expanded. Since segments[N].n is connected to a capacitor having a state, the voltages are copied to the right (n to p) with the first element coming from source.p via a node nSource (since source.p needs to be connected, otherwise source.p.i would be zero). The currents are copied from p to n with the last current coming from load.p.i via the node nLoad.

```
segments.p.v = cat(1, {nSource.v}, segments.n.v[1:N-1]);
segments.n.i = -cat(1, segments.p.i[2:10], {nLoad.i});

nSource.i = segments.p.i[1];
nLoad.v = segments.n.v[N];

connect(source.p, nSource);
connect(source.n, ground.p);

connect(load.p, nLoad);
connect(load.n, ground.p);
```

The compilation time is about 50 milliseconds independent of n. The resulting voltages over the capacitors are shown in **Figure 6**.

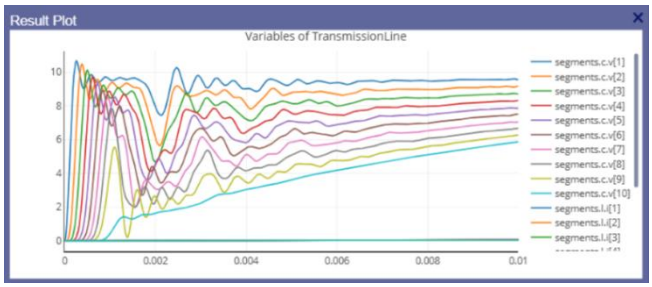


Figure 6. Capacitor voltages.

It should be noted that enabling changes to the discretization after compilation relied on certain specific properties of the model. The capacitor to the right of the TransmissionLineSegment enabled simultaneous copying of all voltages to the right in one array equation. Correspondingly due to the inductor to the left, all currents could be copied simultaneously in one array equation to the left.

4.4 Adaptive Grid

An important advantage of not recompiling when array dimensions change is that a model can refine itself, as shown in the example of this section.

An insulated rod model is shown below and a plot of a simulation in **Figure 7**. It changes the number of segments at time=2000 and time=4000. The changes could also be triggered by detecting some transient behavior.

[7] https://commons.wikimedia.org/wiki/File:Transmission_line_element.svg

DOI
10.3384/ecp218189

Model 5: Insulated rod model with adaptive grid.

```
model InsulatedRod_gradient "Insulated rod with gradient"
  import SI = Modelica.Units.SI;
  parameter SI.Length L=1 "lenght of rod";
  parameter SI.Area A=0.0004 "area of rod";
  parameter SI.Density rho=7500 "density of rod material";
  parameter SI.ThermalConductivity lambda=74
                              "thermal conductivity of material";
  parameter SI.SpecificHeatCapacity c=450
                              "specifc heat capacity";
  parameter SI.Temperature T0=293 "initial temperature";
  parameter Integer nT(min=2)=4 "number of inner nodes";
  SI.Temperature T[nT](start={300, 297.5, 295, 292.5},
              each fixed=true) "temperatures at inner nodes";
  SI.HeatFlowRate Q_flow[nT+1];
  Real dx;
  Real k1;
  Real k2;
  Real k3;
  Real port_a_Q_flow;
  Real port_b_Q_flow;
  SI.Temperature port_a_T;
  SI.Temperature port_b_T;
equation
  dx = L/nT;
  k1 = lambda*A/dx;
  k2 = rho*c*A*dx;
  k3 = 2*k1;

  // Connection equations
  port_a_T = if time<2000 then 300 else 320;
  port_b_Q_flow = 0;

  // Acausal part (assignment depends on connection)
  port_a_Q_flow = k3*(port_a_T - T[1]);
  port_b_Q_flow = k3*(T[nT] - port_b_T);

  // Causal part (assignment does not depend on connection)
  Q_flow = cat(1, {port_a_Q_flow}, k1*(T[1:nT-1] - T[2:nT]),
                  {port_b_Q_flow});
  der(T) = (Q_flow[1:nT] - Q_flow[2:nT+1])/k2;
  when time > 2000 then
    ast.setParameter("nT", 8);
    ast.setStart("T", interpolate(8, T));
  end when;
  when time > 4000 then
    ast.setParameter("nT", 6);
    ast.setStart("T", interpolate(6, T));
  end when;
  annotation(experiment(StopTime=5000));
end InsulatedRod_gradient;
```

A new function ast.setParameter is used to change the grid resolution. Since the state vector changes, new start values for the states are given using ast.setStart. A function *interpolate* is interpolating the temperatures from 4 points to 8 and back to 6. This function considers how the grid points are distributed along the staggered grid. This explains why the curves have discontinuities since T[i] refers to different positions along the rod when nT changes.

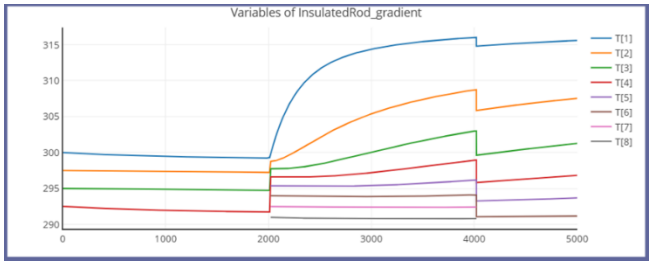


Figure 7. Simulation results of the temperature nodes along an insulated rod. At time=2000 and time=4000 the number of segments is changing from 4 to 8 to 6.

5 Reformulation of Array Equations

In this section, valid Modelica models are presented that are rejected by the algorithm of **Appendix A.1**. It is shown how to reformulate the equations in order that code can be generated.

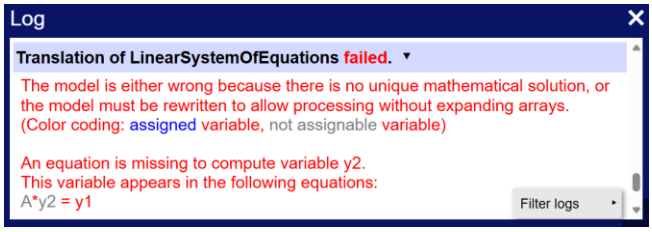
5.1 Linear Equation Systems

Consider the following valid Modelica model that contains a linear system of equations

Model 6: Model with a linear system of equations.

```
model ModelWithLinearSystemOfEquations
  parameter Real n[2] = {1,2};
  parameter Real A[2,2]= [1,2;3,4];
 Real y1[2];
 Real y2[2];
equation
  y1 = n*time;
  A*y2 = y1;
end ModelWithLinearSystemOfEquations;
```

Modiator prints the following error message:



The listed equation must be reformulated, since (a) an equation is missing to compute variable y2, (b) variable y2 appears only in this equation and no variable is assigned to this equation. Note, that y2 is not structurally assignable according to **Theorem 1**. The remedy is to explicitly solve the linear equation system in the model:

```
equation
  y1 = n*time;
  y2 = Modelica.Math.Matrices.solve(A,y1);
```

5.2 Models That Must Be Scalarized

There are various kinds of Modelica models that can only be processed if array variables and equations are scalarized in the generated code. A simple example is shown in the following model:

Model 7: Model where array variables and equations need to be scalarized.

```
model ModelThatRequiresScalarization
  parameter Real n[:]={1,2,3};
  Real s(start=1, fixed=true);
  Real r[3];
equation
    r = n*s;
    der(r)*n = -s;
end ModelThatRequiresScalarization;
```

Modiator prints the following error message:

Translation of ModelThatRequiresScalarization failed.

Equation differentiated 10 times which is too much (most likely processing would be successful if arrays would be expanded, which is not done):
der(r)*n = -s

So, the error message states that the listed equation is differentiated too often and that the likely reason is that array variables in this equation must be scalarized, which is not done in Modiator. If the array variables and equations are scalarized, the following assignment is possible:

assigned	*highest derivative equations*
der(s)	der(r[1]) = n[1]*der(s)
der(r[2])	der(r[2]) = n[2]*der(s)
der(r[3])	der(r[3]) = n[3]*der(s)
der(r[1])	der(r[1])*n[1]+der(r[2])*n[2]+der(r[3])*n[3]=-s

However, it is not possible to convert the scalarized variables and equations back into array variables and equations. One remedy is to split variable r1 and the first equation in two pieces:

Model 8: Reformulation of **Model 7** in order that array variables and equations can be kept without expansion in the generated code.

```
model ChangedModelThatRequiredScalarization
  parameter Real n1 = 1;
  parameter Real n2[:] = {2,3};
  Real s(start=1, fixed=true);
  Real r1;
  Real r2[size(n2,1)];
equation
    r1 = n1*s;
    r2 = n2*s;
    der(r1)*n1 + der(r2)*n2 = -s;
end ChangedModelThatRequiredScalarization;
```

An assignment is now possible without expanding array variables and equations:

assigned	*highest derivative equations*
der(s)	der(r1) = n1*der(s)
der(r2)	der(r2) = n2*der(s)
der(r1)	der(r1)*n1+der(r2)*n2 = -s

5.3 Mechanical Systems

In the following simple model, a mass is sliding along a given direction in 3D space:

Model 9: 3D sliding mass model that requires scalarization.

```
model SlidingMass3DrequiringScalarization
  parameter Real m=1 "Mass of body";
  parameter Real n[3] = {1,0,1}/sqrt(2) "Sliding direction";
  parameter Real g[3] = {0,0,-9.81} "Gravity acceleration";
  Real r[3] "Position";
  Real v[3] "Velocity";
  Real f[3] "Constraint force of prismatic joint";
  Real s "Generalized coordinate of prismatic joint";
equation
  r = n*s;
  v = der(r);
  m*(der(v)-g) = f;
  0 = n*f;
end SlidingMass3DrequiringScalarization;
```

Modiator prints the following error message:

Translation of SlidingMass3DthatRequiresScalarization failed.

The model cannot be processed without exanding arrays because the following equation with arrays does not have at least one unknown variable that appears linearly in the form scalar*variableSymbol:
0 = n*f

The reason is that the *scalar* equation 0 = n*f contains only *array* variables and therefore none of these variables is structurally assignable.

In (Elmqvist and Otter 2017, section 3), it is shown that array variables and equations of mechanical systems must be at least internally expanded in order that processing is possible. After symbolic processing, it is usually possible to recover the non-expanded forms. Since the algorithm in **Appendix A.1** does not expand array variables and equations, *existing* 3D mechanical models, such as those from the Modelica.Mechanics.MultiBody[8] library, cannot be processed by Modiator. For tree-structured mechanical systems it is easy to fix this issue in the following way:

A tree-structured mechanical system with $\boldsymbol{q}$ the vector of generalized minimal coordinates (= generalized coordinates in the joints) and $\boldsymbol{u}$ the generalized forces (= generalized forces in the joints) is described by the following equation:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{u}; \quad \boldsymbol{M} = \boldsymbol{M}^T \geq 0 \qquad (3)$$

In object-oriented modeling, where bodies, joints and other objects can be connected together in a nearly arbitrary fashion, it is easy to define a system where the symmetric mass matrix $\boldsymbol{M}$ is positive semi-definite (that is, it is singular) instead of positive definite as postulated by mechanical principles. Examples:

- A non-zero mass and a *zero inertia-matrix* is defined for a body, and the connection structure allows rotations of the body.

[8] https://doc.modelica.org/Modelica%204.0.0/Resources/helpDymola/Modelica_Mechanics_MultiBody.html

DOI 10.3384/ecp218189

- A chain of mechanical objects ends with a joint and not with a body.
- More as 3 revolute joints are connected directly together, without having a body between the joints.

Let's subtract a small term $\varepsilon_m \ddot{\boldsymbol{q}}$ from the generalized forces $\boldsymbol{u}$, where the scalar $\varepsilon_m \geq 0$:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{u} - \varepsilon_m \ddot{\boldsymbol{q}} \quad (4)$$

or

$$\big(\varepsilon_m \boldsymbol{I} + \boldsymbol{M}(\boldsymbol{q})\big)\ddot{\boldsymbol{q}} + \boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{u} \quad (5)$$

where $\boldsymbol{I}$ is the unit-matrix. The new mass matrix is positive definite (that is, regular) if $\varepsilon_m > 0$. This follows from its positive quadratic form:

$$\boldsymbol{x}^T(\varepsilon_m \boldsymbol{I} + \boldsymbol{M})\boldsymbol{x} = \varepsilon_m \boldsymbol{x}^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{M}\boldsymbol{x} > \boldsymbol{0}, \text{if } |\boldsymbol{x}| > \boldsymbol{0} \quad (6)$$

since $\varepsilon_m \boldsymbol{x}^T\boldsymbol{x} > \boldsymbol{0}$ and $\boldsymbol{x}^T\boldsymbol{M}\boldsymbol{x} \geq \boldsymbol{0}$. Therefore, the mentioned issues can no longer occur if $\varepsilon_m > 0$ and a mechanical model in Modelica becomes more reliable against user errors. Note, the original multibody system is unchanged, if $\varepsilon_m = 0$. Processing with unexpanded arrays becomes possible by adding a corresponding term to the joints of a Modelica model, as shown exemplarily by the following modified version of the sliding mass model:

Model 10: Changed **Model 9** in order that assignment is possible without expanding arrays.

```
model SlidingMass3D
  ...
  parameter Real eps_m(min=0) = 0 "Small mass in joint";
equation
  r = n*s;
  v = der(r);
  f = m*(der(v)-g);
  sd = der(s);
  0 = n*f + eps_m*der(sd); // der(sd) structurally assignable
end SlidingMass3D;
```

The last (scalar) equation is now structurally assignable for the scalar der(sd). With the algorithm of **Appendix A.1** the following assignment can be derived:

assigned	*highest derivative equations*
der(der(r))	der(der(r)) = n* der(der(s))
der(v)	der(v) = der(der(r))
f	f = m*(der(v)-g)
der(der(s))	der(sd) = der(der(s))
der(sd)	0 = n*f + eps_m*der(sd)

When sorting these highest derivative equations, an algebraic loop is identified that corresponds to (5). The equations can be, for example, transformed in the following linear equation in one unknown:

```
input: der(sd)  // tearing variable
output: residue
der(der(s)) := der(sd)
der(der(r)) := n*der(der(s))
der(v) = der(der(r))
f := m*(der(v)-g)
residue := n*f + eps_m*der(sd)
```

Inserting all terms in the last equation and setting residue=0, results in one linear equation with der(sd) as unknown:

```
(eps_m + m*(n*n))*der(sd) = m*(n*g)
```

Note, if arrays are no longer expanded for mechanical systems, the number of modes of *analytically described elastic bodies*, such as *beams* or *plates*, can be changed after translation!

6 Conclusions

The Modelica Web-App Modiator (Elmqvist and Otter, 2025) does not expand arrays of equations, for loops or arrays of components in order to speed up translation and avoid recompilation when array changes are made. The needed restrictions have been described, translation techniques have been outlined, and several presented model examples show the benefits of this new method. As a result, the Modiator Web-App has compilation times independent of array sizes (O(1) instead of O(n) or worse) and arrays can be resized after compilation.

Special emphasis has been put on enabling index reduction without scalarization for models described by array equations. The developed extensions of the Pantelides algorithm are utilized in Modiator and allow index reduction directly on array equations without expanding arrays (also not internally). The algorithm is described and the proofs of its properties are given. The needed changes to the symbolic processing of a Modelica tool are modest and reasonable diagnostics can be given if index reduction and/or assignment fails by pointing to the equations and arrays that need to be changed, in order that processing is possible without expanding array equations and arrays.

The enhanced algorithms presented in the **Appendix**, together with tests for some of the models in this paper, are also provided in Javascript format in file *assign-highest-derivatives-test.html* in the accompanying zip-file. *Dragging this file into a web browser will execute the tests* and display the test results in the web browser window.

Acknowledgements

The authors want to thank Clément Coïc for providing the water hammer model in **Section 4.2**.

This work was partially funded for the first author by the German Federal Ministry of Education and Research (BMBF, grant number 01IS23062C) within the European ITEA4 research project OpenSCALING[9].

[9] https://itea4.org/project/openscaling.html

References

Abdelhak, Karim, Francesco Casella and Bernhard Bachmann (2023). “Pseudo Array Causalization”. In: *Proceedings of the 15th International Modelica Conference*, DOI: 10.3384/ecp204177.

Abdelhak, Karim, and Bernhard Bachmann (2025). “Constant Time Causalization using Resizeable Arrays”. In: *Proceedings of the 16th International Modelica Conference*, Lucerne, Switzerland.

Casella, Francesco (2015). “Simulation of Large-Scale Models in Modelica: State of the Art and Future Perspectives”. In: Proceedings of the *11th International Modelica Conference*, DOI: 10.3384/ecp15118459.

Elmqvist, Hilding and Martin Otter (2025): “Modiator, a Web App for Modelica Simulation“. In: *Proceedings of* the *16th International Modelica & FMI Conference*, Lucerne.

Fritzson, Peter et al. (2020). “The OpenModelica Integrated Environment for Modeling, Simulation, and Model-Based Development”. In: *Modeling, Identification and Control*. 41(4), pp. 241-285. DOI 10.4173/mic.2020.4.1

Marzorati, Denise, Joaquin Fernandez and Ernesto Kofmann (2024): “Efficient Matching in Large DAE Models”. In: *ACM Transactions on Mathematical Software*, Volume 50, Issue 4, Article No. 18, pp. 1-25. DOI: 10.1145/3674831

Mattsson, Sven-Erik and Gustaf Söderlind (1993). “Index Reduction in Differential-Algebraic Equations using Dummy Derivatives”. In: *SIAM Journal of Scientific Computing*. 14(3), pp. 677-692.

Modelica Association (2023). *Modelica – A Unified Object-Oriented Language for Systems Modeling, Language Specification, Version 3.6*. URL: https://specification.modelica.org/maint/3.6/MLS.html.

Otter Martin and Hilding Elmqvist (2017): “Transformation of Differential Algebraic Array Equations to Index One Form”. In: *Proceedings of* the *12th International Modelica Conference*, Prag. DOI: 10.3384/ecp17132565

Pantelides, Constantinos C. (1988). “The Consistent Initialization of Differential-Algebraic Systems”. In: *SIAM Journal on Scientific and Statistical Computing* 9.2, pp. 213–231. DOI: 10.1137/0909014

Pop, Adrian et al. (2019). “A New OpenModelica Compiler High Performance Frontend”. In: Proceedings of the *13th International Modelica Conference*, pp. 689–698. DOI: 10.3384/ecp19157689

Tarjan, Robert (1972): "Depth-First Search and Linear Graph Algorithms". In: *SIAM Journal on Computing* 1(2), pp. 146-160. DOI: 10.1137/0201010

Zimmermann, Pablo, Joaquín Fernández, and Ernesto Kofman (2020). “Set-Based Graph Methods for Fast Equation Sorting in Large DAE Systems”. In: *Proceedings of the 9th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools. EOOLT’2019*. Berlin, Germany, Association for Computing Machinery. DOI: 10.1145/3365984.3365991

Appendix A

In this appendix, the enhancements of standard algorithms are sketched to find a complete assignment (2) for a DAE (1) without expanding array variables and equations.

A.1 Assign Highest Derivatives

Starting point is the algorithm of Pantelides (1988). The modified algorithm is provided in Javascript format (which is nearly the same as C/C++ format here) in the following listing. The yellow parts are deviations from (Pantelides, 1988) and are discussed below:

Listing 1: Function assignHighestDerivatives in Javascript format to determine highest derivative equations that have a complete assignment with respect to *structurally assignable* highest derivative variables. Yellow parts are deviations from (Pantelides, 1988).

```
a1  function assignEquation(i,
                            G,Ga,assign,vColor,eColor,vInspect){
      const Gi = G[i]
      const Gai = Ga[i]
      eColor[i] = true
      for (let j=0; j<Gi.length; j++) {
        const v = Gi[j]
        if(vInspect[v] === -1 && assign[v] === -1 && Gai[j]){
          assign[v] = i;
          return true}}
      for (let j=0; j<Gi.length; j++) {
        const v = Gi[j]
        if (vInspect[v] === -1 && assign[v] > -1 && !vColor[v]
                                                 && Gai[j])  {
          vColor[v] = true
          if (assignEquation(assign[v],
                             G,Ga,assign,vColor,eColor,vInspect)) {
            assign[v] = i;
            return true}}
      }
      return false
    }
a2  export function assignHighestDerivatives(G,Ga,A) {
      let nA  = A.length
      let nB  = G.length
      const assign = Array(nA).fill(-1)
      const B      = Array(nB).fill(-1)
      const eColor = Array(nB)
      const vColor = Array(nA)
a3    function colorEquation(i) {
        eColor[i] = true
        for (const v of G[i]) {
          if (A[v] === -1 && assign[v] > -1 && !vColor[v]
                          && B[assign[v]] === -1) {
            vColor[v] = true
            colorEquation(assign[v])}
        } // end for
      }
a4    const nB_initial = nB
      for (let k = 0; k < nB_initial; k++) {
        let i = k
        while (true) {
          eColor.fill(false)
```

```
    vColor.fill(false)
    if (assignEquation(i,G,Ga,assign,vColor,eColor,A)) break
a5  for (let j=0; j<nB; j++) {
      if (eColor[j] && Ga[j].some(val=>val===false)){
        colorEquation(j)}}}
a6  for (let j=0; j<nB; j++) {
      if (eColor[j]) {
        nB++; B.push(-1); B[j]=nB-1;
        for (const v of G[j]) {
          if (A[v] === -1) {
            nA++; A.push(-1); A[v]=nA-1; assign.push(-1)}}
        G.push( Gj.map(m => A[m]) )
        Ga.push( Ga[j] )}}
    for (let j = 0; j < nA; j++) {
      if (vColor[j]) assign[A[j]] = B[assign[j]]}
a7  i = B[i]
    eColor.length = nB
    vColor.length = nA
  } // end while
a8 } // end for
   return [assign,B]}
```

Utility function assignEquation(i,...), see code block (a1), is basically function AUGMENTPATH of (Pantelides, 1988). The function inspects all variables v of equation i that are *structurally assignable* and have vInspect[v] = -1. It returns true if a v is assigned to equation i and otherwise it returns false. If false is returned, the following property holds:

Theorem 2: *If* assignEquation *returns with* false, *the equations with* eColor[k] = true *are minimally structurally singular with respect to the variables with* vColor[k] = true *(these variables are structurally assignable with respect to the assigned equations).*

Proof: Follows from the proof of Lemma 3.3 of (Pantelides, 1988), with the only difference that a subset of the variables is inspected (variables that are structurally assignable and have vInspect[v] = -1, i.e. are highest derivative variables since A is passed to the function as vInspect). ■

This property means that the colored equations are the smallest subset that need to be differentiated. This subset has more equations than variables and therefore differentiating this subset will introduce more new equations as it will introduce more new variables.

The core function assignHighestDerivatives, see code blocks (a2-a9), inspects all equations in sequence. For every equation, function assignEquation is called with vInspect = A, that is only highest derivative variables are inspected, see last statement in code block (a4). If the function returns true, the next equation is inspected. Otherwise, a while loop iterates over the function call, until an assignment is found. If assignEquation returns false, the colored set of equations needs to be differentiated.

Colored equations that have at least one highest derivative variable that is *not* structurally assignable are inspected by calling the function colorEquation on them, see code blocks (a5,a3). This function colors additional equations that need to be differentiated. Assume for example, that equation j colored by assignEquation has a highest derivative variable v that is not structurally assignable with respect to equation j, is already assigned, is not colored and the equation to which v is assigned is a highest derivative equation. If equation j is differentiated, then v is also differentiated, but without differentiating the highest derivative equation assign[v], because this equation was not colored by assignEquation. The construction of code blocks (a5,a3) ensures, that these assigned equations are also colored and are thus differentiated. Note, this additional equation coloring appears, for example, in **Model 1** from section 3.3.

In code block (a6), the data structures G,Ga,A,B,assign are appended with new entries that include the incidences of the equations and the highest derivative variables that will appear when the equations are differentiated. This part is nearly identical to code block (3b-5) of ALGORITHM 4.1 of (Pantelides, 1988). The essential difference is that in (3b-5) of (Pantelides, 1988) first all colored variables are appended to A before G,B are updated. However, this is not possible here, because a colored equation j may have a variable v that is not structurally assignable and that is not assigned. When differentiating equation j, v is differentiated as well, but the differentiated v may not yet exist because it was not colored. For this reason, A is updated when the incidence of the differentiated equation j is appended to G. The remaining part of the code is identical to ALGORITHM 4.1 of (Pantelides, 1988). Function assignHighestDerivatives returns the newly constructed arrays assign,B via the return statement, and the updated arrays G,Ga,A via the argument list.

A.2 Assign Extended System

Since assignHighestDerivatives has a while loop, the question arises whether this loop will terminate for all inspected equations. In (Pantelides, 1988) an *extended* system $\boldsymbol{EG} = \begin{bmatrix} \boldsymbol{G} \\ \boldsymbol{G}_h \end{bmatrix}$ is analyzed consisting of $\boldsymbol{G}$ and of the incidence matrix $\boldsymbol{G}_h$ of additional equations $0 = h_i(v_i, \dot{v}_i)$ for every variable v_i appearing differentiated. A corresponding matrix $\boldsymbol{EG_a} = \begin{bmatrix} \boldsymbol{G}_a \\ \boldsymbol{G}_{h,a} \end{bmatrix}$ defines whether the elements of $\boldsymbol{EG}$ are structurally assignable or not. Hereby, all elements of $\boldsymbol{G}_{h,a}$ representing the new equations h_i are true.

Theorem 3: *If* $\boldsymbol{G}_a$ *has only true values then the* while *loop in function* assignHighestDerivatives *terminates for all inspected equations after a finite number of iterations if and only if the extended system* $\boldsymbol{EG}$ *is structurally nonsingular. In such a case, the highest derivative equations of* $\boldsymbol{G}$ *have a complete assignment with respect to the highest derivative variables. If* $\boldsymbol{EG}$ *is structurally singular, then the underlying DAE (1) has no solution since it is impossible to find a set of consistent initial conditions.*

Proof: This is Theorem 4.2 of (Pantelides, 1988) formulated with the notation of this paper. ■

Unfortunately, this strong property does no longer hold, if one or more elements of $\boldsymbol{G}_a$ are false, that is, not all variables are structurally assignable. This is demonstrated with the simple example of **Model 7** of section 3.3. A complete assignment of the extended system $\boldsymbol{EG}$ with respect to all structurally assignable variables is possible:

assigned	*equations of extended system*
r	r = n*s
s	der(r)*n = -s
der(r)	0 = h(r, der(r))

Analyzing the equations of **Model 7** with function assignHighestDerivatives results in the following sequence of assignments: Function assignEquation returns false for r = n*s because r is not a highest derivative variable and s is not structurally assignable. Therefore, this equation is differentiated leading to the new highest derivative equations:

assigned	*highest derivative equations*
der(r)	der(r) = n*der(s)
---	der(r)*n = -s

Function assignEquation returns false for der(r)*n = -s because der(r) is not structurally assignable and s is no highest derivative variable. Differentiating this equation and additionally the first equation due to code block (a5), results in the following new highest derivative equations:

assigned	*highest derivative equations*
der(der(r))	der(der(r)) = n*der(der(s))
---	der(der(r))*n = - der(s)

A similar situation as in the previous step occurs: The second equation needs to be differentiated, because der(der(r)) is not structurally assignable and der(s) is no highest derivative variables. Also, the first equation is differentiated due to code block (a5). As a result, an infinite number of differentiations occurs.

If instead, variables r and n are expanded, the highest derivative equations are identified and assigned as shown in section 3.3. As can be seen, it is not possible for this model to have an assignment of structurally assignable variables without expanding array variables and array equations. ■

The consequence is that the number of iterations of the while loop in function assignHighestDerivatives must be explicitly limited and when this limit is reached, the function either returns with an error or the model is processed again with all arrays expanded. Assume that the following assumption holds (which is checked before function assignHighestDerivatives is called):

Assumption 1: *The number of equations e_i of the underlying DAE (1) is identical to the number of highest derivative variables, so variables v_j that have $A_j = -1$.*

then the following theorem states the property of the return result of function *assignHighestDerivative*:

Theorem 4: *The highest derivative equations of $\boldsymbol{G}$ have a complete assignment with respect to their structurally assignable highest derivative variables, if Assumption 1 holds and the while-loop in function assignHighestDerivatives terminates after a finite number of iterations for all inspected equations.*

Proof: If the function returns successfully, then all underlying equations have been inspected once, due to the for-loop of code block (a4). Since every equation gets an assignment for a structurally assignable highest derivative variable that is not yet assigned, due to code block (a1), all equations are assigned for highest derivative variables that are structurally assignable. This assignment does not influence already assigned equations, because otherwise these equations would have been colored and then differentiated together with the not-yet assigned equation and their assigned variables would have been differentiated correspondingly. Due to **Assumption 1**, all highest derivative variables or derivatives of them are assigned because the same number of equations get an assignment. ■

If the *while*-loop does not terminate, then either the DAE (1) has no unique solution, or a complete assignment with respect to *expanded* arrays is possible.

In order that meaningful diagnostics can be given, the following theorem is utilized:

Theorem 5: *If the extended System $\boldsymbol{EG}$ is structurally singular with respect to the structurally assignable variables, the while-loop does not terminate for at least one of the inspected equations.*

Proof: Follows from the proof of Theorem 4.2 in (Pantelides, 1988), Part B until "the algorithm will keep differentiating until infinitum". ■

Note, if the extended System **EG** is structurally singular with respect to the structurally assignable variables, it is still possible that the DAE is solvable if the array variables and array equations are expanded. This can be easily seen from **Model 6** in section 5.1. The extended System **EG**

assigned	*equations of extended system*
y1	y1 = n*time
--	A*y2 = y1

is *structurally singular* with respect to the structurally assignable variables because the first equation can be assigned to y1 and the second equation can be also assigned to y1 but not to y2, because y2 is not structurally assignable according to **Theorem 1**. Therefore, the while loop does not terminate, but this can be detected beforehand. If arrays are expanded, the elements of the second equation can be assigned to the elements of y2 and the extended system is *structurally regular*. ■

DOI 10.3384/ecp218189

Listing 2 shows the code for function assignExtendedSystem that builds and checks the extended system for structurally assignable variables.

Listing 2: Function assignExtendedSystem in Javascript format to determine a complete assignment of the extended system with respect to the *structurally assignable* variables. Yellow parts are deviations from (Pantelides, 1988).

```
b1  export function assignExtendedSystem(G,Ga,A) {
      // Build extended system
      const EG  = [...G]
      const EGa = [...Ga]
      const nA  = A.length
      for (let i=0; i<nA; i++) {
        const Ai = A[i];
        if (A[i] > -1) {
          EG.push([i, A[i]]); // h-equation
          EGa.push([true, true])}
      }
b2    // Assignment of extended system
      const assign   = Array(nA).fill(-1)
      const vColor   = Array(nA)
      const eColor   = Array(EG.length)
      const vInspect = Array(nA).fill(-1)  // inspect all variab.
      const unassignedEquations = []
      for (let i=0; i<nA; i++){
        eColor.fill(false)
        vColor.fill(false)
        if ( !assignEquation(i,EG,EGa,
            assign,vColor,eColor,vInspect) && i<G.length){
          unassignedEquations.push(i)}
      }
b3    // Collect unassigned variables
      const unassignedVariables = []
      for (let j=0; j<assign.length; j++) {
        if (assign[j] === -1) unassignedVariables.push(j)
      }
      return [unassignedEquations, unassignedVariables]
    }
```

In code block (b1) the extended system for G and Ga is constructed. In code block (b2), function assignEquation is called for all equations of the extended system and for *all* of its variables (due to vInspect = Array(nA).fill(-1)). If assignEquation returns false, and the inspected equation is no h-equation, it is pushed on stack unassignedEquations. (beause h-equations should not be reported to the user and if a h-equation is not assigned, there is also an unassigned variable). In code block (b3) the unassigned variables are collected in stack unassignedVariables. Both stacks are returned. If one of them is not empty, the complete assignment of the extended system failed and the unassigned equations and unassigned variables are returned and reported to the user.

The practical application of the theorems requires additionally the following small extension:

Function assignHighestDerivatives(G, Ga, A, maxDer =-1) gets an additional optional argument that defines the maximum number of allowed differentiations for an equation (if maxDer =-1, the number of differentiations is not limited) . If this limit is reached, the function terminates. It returns [assign,B,eqDiffTooMuch]. If the maximum number of differentiations is reached, eqDiffTooMuch is the equation of the underlying DAE that was differentiated too often. This equation can be reported in the error message.

A.3 Improved Error Diagnostics

The information for the error messages presented in section 3.4, is deduced from assignExtendedSystem(…) of **Listing 2** by extending this function a bit:

- The function already collects all unassigned variables. For every unassigned variable, all equations are inspected where this variable appears and then these equations are printed together with their assignments.
- The function also collects all unassigned equations which is not very useful. Instead, when no assignment can be found, *all colored equations* are collected together because they constitute the set of equations that, together with the unassigned equation, creates an overdetermined system. This combined set is stored. If more overdetermined sets of equations appear, only the one with the smallest number of equations is kept.

A.4 Further Symbolic Processing

A tool could call the presented functions in the following order:

1. Call assignExtendedSystem. If the function returns with unassigned equations and/or unassigned variables either print an error message showing the unassigned equations/variables ("*Either DAE is not solvable or processing not possible without expanding arrays, which is not supported*") or expand all arrays and equations and perform the standard processing of the tool for expanded equations/arrays.
2. If assignExtendedSystem returns successfully, call assignHighestDerivatives. If the function terminates because the maximum number of differentiations reached, either print an error message showing the equation that was differentiated too often ("*Equation eqDiffTooMuch differentiated until the limit of maxDer differentiations reached. Either DAE is not solvable or processing is not possible without expanding arrays, which is not supported.*") or expand all arrays and equations and perform the standard processing of the tool for expanded equations/arrays.
3. If assignHighestDerivatives returns successfully, a full assignment of the highest derivative equations with respect to the structurally assignable highest derivative variables was found and standard processing continues with only symbol information (so with unexpanded equations/arrays).

After the complete assignment of the highest derivative equations with respect to the structurally assignable highest derivative variables is determined with function

assignHighestDerivatives the standard approach of object-oriented modeling can be used for further processing and code generation by using basically only the symbol information (without expanding arrays):

1. *Analytically differentiating the equations*
   The equations are analytically differentiated based on the information provided in $\boldsymbol{G}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{assign}$.
2. *Equation sorting*
   $\boldsymbol{G}$ and ***assign*** of the highest derivative variables $\boldsymbol{A}$ with $A_k = -1$ define a directed graph where the nodes are the highest derivative equations together with the assigned highest derivative variables. The edges are defined by the rows of the incidence matrix $\boldsymbol{G}$ that correspond to the highest derivative equations. With the algorithm of (Tarjan, 1972) the strong components (algebraic loops) in the directed graph are determined and the equations are sorted.
3. *State selection*
   From the sorted highest derivative equations, the constraint equations are determined by the algorithm of (Mattsson and Söderlind, 1993) based on the highest derivative equation systems determined in step 2. From this information states are statically and/or dynamically selected.
4. *Code generation*
   Code is generated. Arrays with small, fixed dimensions and corresponding array equations with small, fixed dimensions might be expanded. If arrays are kept unexpanded, then the generated code must support array operations where dimension sizes might be only known at run-time and can be changed after translation to object code. Typically, Modelica tools support this already, because code generation for Modelica functions need a similar infrastructure to operate on arrays.

DOI 10.3384/ecp218189

Constant Time Causalization using Resizable Arrays

Karim Abdelhak[1] Bernhard Bachmann[1]

[1]Institute for Data Science Solutions, Bielefeld University of Applied Sciences and Arts, Germany,
{karim.abdelhak, bernhard.bachmann}@hsbi.de

Abstract

Equation-based modeling that utilizes reusable components to represent real-world systems can result in excessively large models. This, in turn, significantly increases compilation time and code size, even when employing state-of-the-art scalarization and causalization techniques. This paper presents an algorithm that leverages repeating patterns and uniform causalization to enable array-size-independent constant time processing. Allowing structural parameters that govern array sizes to remain resizable during and after the causalization process enables the formulation of an integer-valued nonlinear optimization problem. This approach identifies the minimal model configuration that preserves the required structural integrity, which can subsequently be resized as needed for simulation. The proposed method has been implemented in OpenModelica and builds upon preliminary work aimed at preserving array structures during causalization, while still resolving the underlying problem in a scalarized manner.

Keywords: Equation-based modelling, Array-preserving, Causalization, Resizable, Nonlinear programming, Integer programming

1 Introduction

Current state-of-the-art approaches for compiling large-scale models with scalable structures typically rely on full scalarization. Although this method yields correct results, it often incurs substantial compilation times and leads to significantly larger code sizes. *OpenModelica*'s (Fritzson, Pop, Abdelhak, et al. 2020) pseudo-array compilation algorithm (Abdelhak, Casella, and Bachmann 2023) addresses the challenges of large code sizes and prolonged compilation times by enabling array-size-independent symbolic manipulation, while still performing causalization on scalarized graphs. Building on this foundation, this paper presents an algorithm that systematically reduces scalable structures to a theoretical minimal form, allowing causalization to be performed efficiently. Following compilation, the resulting model can be resized to any desired scale suitable for simulation.

Thematically related research exploring the exploitation of resizable array structures includes the work presented in (Neumayr and Otter 2023), which introduces predefined acausal components as a means of enabling flexible array configurations. Additionally, an alternative to the pseudo-array compilation algorithm has been proposed in (Marzorati, Fernández, and Kofman 2024), where a set-based graph approach is employed. This method is inherently designed to operate independently of array sizes, representing a structurally distinct approach to addressing the challenges associated with scalable model compilation. Although the paper does not explicitly address the resizability of set-based nodes within the set-based graph, the algorithms presented in this paper for determining the general resizability of for-loop equations should be equally applicable, given that the underlying model remains unchanged.

1.1 Resizable Parameters

First and foremost, structural parameters that determine the array sizes within a model must be preserved and not evaluated throughout the flattening process. These parameters are *resizable* but will be simply referred to as *parameters* in all subsequent discussions. While the concept of minimizing model size may appear to align directly with minimizing the values of these parameters, such a correlation is not guaranteed. The modeling language *Modelica* permits array sizes to be defined by any expression that yields a non-negative integer (Modelica Association 2023). As a result, the relationship between parameter values and overall model size can even be nonlinearly correlated. Additionally, a single parameter may influence multiple array sizes, and conversely, multiple parameters may contribute to determining a single array size. *OpenModelica* offers two different ways to indicate that a parameter is *resizable*, which will be described in detail in Section 3.3.

1.2 Resizable Equations

Any equation whose size is determined by a resizable parameter is itself considered resizable. In the modeling language *Modelica*, there are two primary constructs for expressing such equations: *array equations* and *for-equations*. While array equations are a conventional method for representing multidimensional relationships, the latter are more specific to Modelica. For-equations employ a surrounding for-loop construct—similar to those used in algorithmic programming—to define a set of structurally similar relationships using array variables indexed by one or more iterator variables.

The primary advantage of identifying resizable equations lies in the ability to reduce their size as much as pos-

sible during the process of causalization. This enables the transformation of the flattened model into target code to be performed independently of the resizable array sizes defined within the model. It is important to note that in order to keep the system structure parameters and equations must satisfy specific conditions and cannot be resized arbitrarily.

Assumption 1 *The following discussion operates under the assumption that the model remains balanced for every feasible combination of parameter values. In other words, each parameter is assumed to influence variable dimensions and corresponding equation sizes in a consistent manner.*

Example 1 *As an example of an invalid resizable parameter, consider the following model. While the original definition with $p = 2$ yields a balanced and valid model, any other value of p results in an invalid configuration.*

```
model invalid_resizable
  parameter Integer p = 2;
  Real[p*p] x
equation
  for i in 1:2*p loop
    x[i] = sin(time) * i;
  end for;
end invalid_resizable;
```

2 The Optimization Problem

The primary objective of the optimization is to minimize the overall model size. Owing to Assumption 1, this objective can be simplified to minimizing either the total number of variables or equations, as their sizes are assumed to be proportionally aligned. In the following, the optimization problem is formulated with a focus on minimizing the sizes of the equations. As will be demonstrated in the following sections, the decision variables are integer-valued, and the objective function is inherently nonlinear. The constraint set includes both inequalities and equalities, which—although infrequently—may also exhibit nonlinear characteristics.

2.1 Objective Function

Let $x \in \mathbb{Z}^n$ denote the vector of decision variables representing resizable parameters. The goal is to determine the values of x that minimize the total number of equations, subject to structural and semantic constraints.

$$\arg\min_{x} \quad F(x) = \sum_{k} f_k(x) \tag{1}$$

where $F(x)$ is the total model size and f_k being the local size objectives of equation k as product of their dimension objective functions:

$$f_k(x) = \prod_{j} d_j^k(x) \tag{2}$$

The dimension objective function evaluates whether the j-th dimension D satisfies condition (a_j), namely, that it is defined by an iterator within a for-loop whose range $D_{\text{start}} : D_{\text{step}} : D_{\text{stop}}$ is governed by one or more resizable parameters. If (a_j) holds, the objective is set to the number of elements traversed by the iterator. Otherwise, if the size is not explicitly defined by an iterator, the equation must be an array-based equation whose dimensions can be inferred from the array variables it contains. This inference is performed by the function $\dim_j^k(x)$ which may either yield a constant size or, in the case of parameterized array equations, represent a function dependent on one or more resizable parameters.

$$d_j^k(x) = \begin{cases} (D_{\text{stop}}(x) - D_{\text{start}}(x))/D_{\text{step}} + 1 & \text{if } (a_j) \\ \dim_j^k(x) & \text{else} \end{cases} \tag{3}$$

2.2 Constraints

This optimization problem is subject to several constraints, which can be classified into three main categories:

equation dimension constraints. The first set of constraints arises from the requirement that each resizable equation must have a size of at least one to ensure its representation in the underlying bipartite graph. Consequently, each dimension associated with an equation must be of minimum size one. For every iterator i with a range defined as $D^i_{\text{start}} : D^i_{\text{step}} : D^i_{\text{stop}}$, this requirement can be expressed through the following constraint:

$$0 \leq (D^i_{\text{stop}}(x) - D^i_{\text{start}}(x))/D^i_{\text{step}} \tag{g_1}$$

Furthermore, the same condition must hold for each j-th dimension of the k-th array equation that depends on resizable parameters imposing the following constraint:

$$1 \leq \dim_j^k(x) \tag{g_2}$$

structural variable constraints. For each resizable parameter $x_p \in x$ the minimum and maximum value constraints (if given by the underlying model) have to be taken into account:

$$x_p \geq x_p^{\min} \tag{g_3}$$
$$x_p \leq x_p^{\max} \tag{g_4}$$

In addition to being box-constrained by their minimum and maximum values, parameters may also impose mutual restrictions. When a resizable variable is indexed using a resizable parameter, it is essential to ensure that this indexing operation does not result in an out-of-range access.

Example 2 *Let y be a resizable variable whose size is determined by the parameter x_1. Additionally, consider a component reference to y in the model as*

DOI 10.3384/ecp218203

$y[x_2]$, where x_2 is another parameter. This imposes a constraint on x_2, requiring that $x_2 \leq x_1$ to prevent out-of-range access.

The same logic applies to non-parameter indexed variables, constant valued restrictions have to be collected as well. For each component reference that references a resizable variable in the model, each dimension has to be checked if it imposes a constraint.

$$\text{sub}_c^j(x) \leq \text{cdim}_c^j(x) \qquad (g_5)$$

where c refers to c-th component reference and j to the j-th dimension $\text{cdim}_c^j(x)$ with $\text{sub}_c^j(x)$ being the subscript of c in dimension j. If a constraint does not involve any parameters or if the parameters cancel out, its validity must be verified. If the constraint is valid, it can be safely omitted; otherwise, an error should be raised, as this indicates an issue with the underlying model.

structural equation constraints. The final set of constraints is derived from array equations and, in contrast to the previous inequality constraints, consists of equality constraints. For each array equation k, the j-th dimension on the left-hand side, represented by the function $\text{lhs}_j^k(x)$, must be equal to the corresponding right-hand side dimension rhs_j^k. This ensures consistency between both sides of the equation and results in the following equality constraint:

$$\text{lhs}_j^k(x) = \text{rhs}_j^k(x) \qquad (h)$$

3 Solving the Problem

Solving the general integer-valued nonlinear optimization problem is known to be NP-hard (Kannan and Monma 1981). However, for the majority of practical models, the problem structure is predominantly linear and can be efficiently addressed under typical conditions. Moreover, obtaining the global optimum is not strictly necessary. Any method that consistently yields a feasible, finite solution—while maintaining computational complexity independent of the values of resizable parameters—successfully achieves the objective of array-size-independent compilation time. For the results presented in Section 4, a simple greedy hill-climbing algorithm was implemented as a proof of concept, using the original model-defining parameter configuration as the initial starting point.

3.1 Utilization of Optimization Results

Once a feasible parameter configuration has been identified—yielding a sufficiently small, array-size-independent model—all affected model sizes must be updated accordingly. In addition, auxiliary structures must be generated and stored to enable recovery of the original array sizes. The bipartite graph used for equation-variable matching, as well as the directed graph used for equation sorting, are both constructed based on these reduced model sizes. This results in a causalization problem that is independent of array sizes. After the causalization is completed, the original array structures are reconstructed using the three-step sorting procedure described in (Abdelhak, Casella, and Bachmann 2023). Notably, the third sorting step differs for resizable equations, as these do not require resolution through the underlying graph. Instead, it is sufficient to analyze the indexed variables they contain and their usage patterns.

resizable array-equation If a resizable array equation is solved for the same variable instance in each of its scalar sub-equations, it poses no additional challenge, as the equation body can be resolved using established techniques. However, when explicit solving is not possible, or when different variable instances are involved, the equation should be reformulated in residual form and handled implicitly during simulation.

resizable for-equation Let $0 = f(X,Y,I)$ denote the for-equation expressed in residual form, where $\hat{x} \in X$ is the component reference to be solved for, and X is the set of all component references corresponding to the same variable x as $\hat{x}$. The set Y contains all other component references appearing in the equation, and I represents the set of for-equation iterators along with their respective ranges. To determine the execution order of the individual scalar sub-equations within f, one examines the elements of X and compares their positions relative to $\hat{x}$, using the indexing information provided by I. The component references in Y do not influence the execution order and can therefore be disregarded for this purpose. If multiple variable instances are intended to be solved, the current implementation does not support retaining the equation in resizable form (see Section 3.2). The complete method is presented in Algorithm 1, which determines the execution order for each iterator. The execution order for a given iterator can fall into one of three categories:

a) in the original order (forwards)
b) reverse to the original order (backwards)
c) in any order (arbitrary)

Once determined, the inner equations do not need to be sorted based on the underlying scalar dependency graph, as their execution order is fully defined by the enclosing structure. As shown in Algorithm 1, the algorithm may revert the resizable status of variables used to determine the size of the observed equation. In such cases, this behavior must either be reported to the modeler via an error message—aborting compilation to allow for appropriate model adaptation—or issued as a warning, allowing compilation

Algorithm 1 Execution Order

Input: component reference to solve for $\hat{x}$
Input: set of component reference of the same variable X
Input/Output: set of iterators, their ranges and markings I
if there is only one component reference $X = \{\hat{x}\}$ **then**
    Mark all iterators in I with **arbitrary**
else
    for each dimension d in x **do**
        Find the set O of all subscripts in dimension d for all elements of X
        Let I be the set of all iterators occuring in O
        if there is only one iterator $I = \{i\}$ **then**
            compute the set of partial derivatives $D \leftarrow \{\delta o/\delta i | o \in O\}$
            if $|D| = 1$ **then**
                compute the set of all offsets F for the elements of O
                let $\hat{f} \in F$ be the offset for the occurence of $\hat{x}$
                if $\hat{f} \leq f \; \forall f \in F$ and i is not marked **forwards then**
                    Mark i with **backwards**
                else if $\hat{f} \geq f \; \forall f \in F$ and i is not marked **backwards then**
                    Mark i with **forwards**
                else
                    Mark i with **arbitrary**
            else
                The equation is not resizable
                return
        else
            The equation is not resizable
            return

to proceed. In the latter case, the resizable status of the affected variables is reverted, the modified optimization problem is resolved, and the causalization pipeline is subsequently executed again.

Example 3 *In the trivial case where a for-equation is solved for a variable and no other instances of that variable appear within the equation, the execution order is arbitrary. The following example illustrates a non-trivial case involving two possible scenarios. The equation may be solved either for $x[i]$ or for $x[i+1]$:*

```
for i in 1:p loop
  x[i+1] = x[i]*2;
end for;
```

If $\hat{x} \equiv x[i+1]$, the structure of the equation body remains unchanged after solving, and the iterator i can be traversed in its original forward order, since each computed element depends only on a previously computed one. Conversely, if $\hat{x} \equiv x[i]$, the iterator range must be traversed in reverse order, as each element otherwise would depend on a value that is computed in a later iteration.

Expanding the idea by adding a third indexed instance of x leads to following examplary for-equation:

```
for i in 1:p loop
  x[i] = x[i+1]*2 + x[i+2]*3;
end for;
```

If $\hat{x} \equiv x[i+2]$, the execution order is forward, and if $\hat{x} \equiv x[i]$, it is backward—consistent with the behavior described previously. However, if $\hat{x} \equiv x[i+1]$, the computation would depend on both a preceding and a subsequent value, resulting in an algebraic loop. In such cases, since residual equations in an implicit system do not require a specific execution order, the iterator traversal is considered arbitrary.

3.2 Limitations and Future Work

Although this approach is designed to be broadly applicable to general *Modelica* models, certain limitations in its current implementation remain, warranting further investigation and development.

connect equations The modeling language *Modelica* allows the construction of complex models through hierarchical composition, using sub-components interconnected via specialized connectors. These connectors implicitly define a set of equations that are incorporated into the model during the flattening process, which transforms the component-based structure into a single mathematical representation suitable for symbolic manipulation and simulation. A

DOI 10.3384/ecp218203

graph-based algorithm has been developed to determine these connection equations. As with other *Modelica* equations, connect-equations can be enclosed within for-equation constructs, potentially depending on resizable parameters. To enable array-size-independent compilation of such models using the optimization approach presented in this paper, the graph-based connection algorithm must be extended to handle undetermined array sizes during compilation.

split equations If a multi-dimensional equation must be solved for multiple distinct variable instances, it is partitioned into several slices, with each slice being solved for one of the targeted variables. The current implementation of the optimization does not guarantee that the original equation is of sufficient size to ensure that all resulting slices are fully represented in the minimal version of the model. In principle, additional constraints could be incorporated into the optimization problem to guarantee slice-safety; however, doing so would likely require a comprehensive prior analysis to anticipate all potential slicing operations.

entwined equations The concept of entwined equations refers to the evaluation of two or more, potentially multi-dimensional, equations in an alternating sequence (as detailed in (Abdelhak, Casella, and Bachmann 2023)). While, in principle, it is feasible for one or more of these equations to be resizable, the current implementation requires structural enhancements to support the generation of correct and efficient code.

non-trivial index accessing The proposed method supports only linear index access, requiring a consistent multiplicative factor for all accesses to solved variables and their occurrences within resizable equations. This constraint is necessary to ensure a clear and strict dependency structure, which is essential for classification into one of the defined execution order categories. While the algorithm could be extended to handle certain fringe cases, such enhancements are generally unnecessary, as the majority of practical *Modelica* models do not demand a more comprehensive approach.

3.3 Implementation in OpenModelica

As previously mentioned, the proposed approach has been integrated into the OpenModelica compiler and can be activated using the `--newBackend` compiler flag. Resizable array-defining parameters can be specified individually by applying the annotation
`__OpenModelica_resizable=true`.
Alternatively, all array-defining parameters can be treated as resizable by enabling the `--resizableArrays` compiler flag.

4 Results

All experiments were conducted on a laptop with a 12th Gen Intel® Core™ i7-12800H x 20 @ 4.80GHz, 31 GB RAM and a 1TB NVMe SSD, running Ubuntu 20.04.6 LTS. The evaluation was conducted using OpenModelica version 1.24.5-dev, compiled from source.
The `--newBackend` and `--resizableArrays` compiler flags were enabled for the relevant tests.

The presented proof-of-concept implementation was evaluated on several scalable models and consistently produced satisfactory results. The model illustrated in Figure 1, originally developed by Martin Otter, is a simplified variant of the model presented in (Neumayr and Otter 2023). It represents an insulated rod discretized into an array of segments, with the level of discretization controlled by a single parameter, nT, which can be designated as resizable. Specifically, nT determines the sizes of the variables T and Q_{flow}, and also defines the bounds for two for-equations, with iterators i in $2 : nT$ and i in $1 : nT$, respectively. As illustrated, Assumption 1 holds, since the resizable parameter nT uniformly influences both variable and equation sizes. The objective function can be derived from Equations 1, 2, and 3, based on the two iterators that define the sizes of the corresponding for-equations. The sizes of all other equations are omitted from the formulation, as they remain constant.

$$\underset{nT}{\arg\min} \quad F(nT) = 2 \cdot nT - 1 \tag{4}$$

The equations and variables impose a variety of constraints on the optimization problem, the majority of which can be omitted due to being trivially satisfied. However, they are presented in the following as illustrative examples of the method. The resizable parameter nT is restricted by a minimal size, therefore a constraint g_3 can be formulated:

$$nT \geq 1 \tag{5}$$

which is not the only variable constraint in this model. The equation

```
port_a.Q_flow = 2*k1*(port_a.T-T[1]);
```

does not pose a restriction itself, but the indexed variable $T[1]$ results in a constraint g_5:

$$1 \leq nT \tag{6}$$

The same holds for all remaining equations, which involve indexed variables but can be evaluated as trivially satisfied. As such, they do not provide additional illustrative value for the purposes of this example. Each of the two for-equations introduces one constraint, both derived from Equation g_1:

$$0 \leq nT - 2 \tag{7}$$

$$0 \leq nT - 1 \tag{8}$$

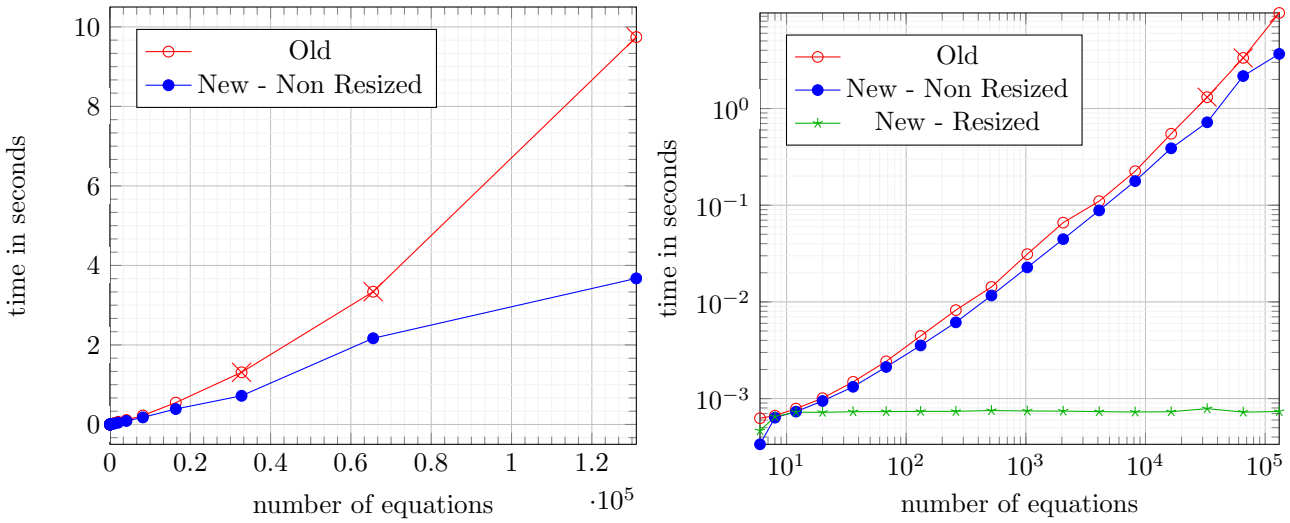
```
model InsulatedRod_equations
"Standard implementaton of an insulated rod with equations"
  import Modelica.Units.SI;
  Modelica.Thermal.HeatTransfer.Interfaces.HeatPort_a port_a;
  Modelica.Thermal.HeatTransfer.Interfaces.HeatPort_b port_b;
  parameter SI.Length L=1 "lenght of rod";
  parameter SI.Area A=0.0004 "area of rod";
  parameter SI.Density rho=7500 "density of rod material";
  parameter SI.ThermalConductivity lambda=74 "thermal conductivity of material";
  parameter SI.SpecificHeatCapacity c=450 "specifc heat capacity";
  parameter SI.Temperature T0=293.15 "initial temperature";
  parameter Integer nT(min=1)=1000 "number of inner nodes";
  SI.Temperature T[nT](each start=T0, each fixed=true) "temperatures at inner nodes";
  SI.HeatFlowRate Q_flow[nT+1];
protected
  parameter Real dx = L/nT;
  parameter Real k1 = lambda*A/dx;
  parameter Real k2 = rho*c*A*dx;

equation
  port_a.Q_flow = 2*k1*(port_a.T - T[1]);
  port_b.Q_flow = 2*k1*(T[nT] - port_b.T);

  Q_flow[1] =port_a.Q_flow;
  for i in 2:nT loop
    Q_flow[i] = k1*(T[i-1] - T[i]);
  end for;
  Q_flow[nT+1] =-port_b.Q_flow;

  for i in 1:nT loop
    der(T[i]) =(Q_flow[i] - Q_flow[i + 1])/k2;
  end for;
end InsulatedRod_equations;
```

Figure 1. Scalable Modelica model representing an insulated rod being heated.



(a) Linear Comparison

(b) Logarithmic Comparison

Figure 2. Comparison of causalization times across three configurations: the old backend, the new backend, and the new backend with support for resizable variables. Nodes marked with a strikethrough indicate that causalization was successfully performed and timed; however, subsequent model building or simulation failed.

DOI 10.3384/ecp218203

Since the model does not contain any array equations, no constraints of the form described in Equation h are present. After omitting all trivially satisfied conditions, the resulting optimization problem can be formulated as follows:

$$\arg\min_{nT} \quad F(nT) = 2 \cdot nT - 1 \tag{9}$$

$$0 \leq nT - 2 \tag{10}$$

The minimal solution to this optimization problem, $nT = 2$, is straightforward to derive and is effectively identified and applied by the *OpenModelica* implementation.

The model presented in Figure 1 has been compiled using different versions of *OpenModelica* to highlight the improvements introduced by the research in this paper. Three configurations are compared: the old backend, the new backend, and the new backend with support for resizable variables. The results of this comparison are presented in Figures 2a and 2b. Figure 2a illustrates the impact of the new backend implementation on causalization time using a linear scale, showing an improvement by a factor of nearly three. Additionally, the updated backend enabled the compilation of larger models, overcoming the scalability limitations of the old backend, which failed to generate code for large models due to memory exhaustion (indicated by 'x' markers in the plots).

While these improvements appear significant, a logarithmic comparison (Figure 2b) reveals that scalability challenges persist: causalization time still increases with array size, despite the reduction of overhead[1]. However, with the introduction of resizable variable support, a fully array-size independent causalization process was achieved, as demonstrated in Figure 2b.

5 Conclusions

This work presents an extension to the OpenModelica compiler backend, introducing enhanced support for resizable equations and improving causalization performance for complex array-based models by ensuring that compilation time no longer depends on array sizes. This is accomplished by formulating and solving an integer-based nonlinear optimization problem to determine the minimal feasible model size, which is then treated as resizable prior to simulation, allowing the final model to adapt to arbitrary sizes before runtime.

The method has four key limitations: first, the graph-based connection algorithm needs extension to handle undetermined array sizes during flattening. Second, the optimization may not fully represent all slices when partitioning multi-dimensional equations, requiring additional constraints. Third, entwined equations need structural improvements for efficient code generation. Finally, the method currently supports only linear index access, which is sufficient for most practical *Modelica* models but could be extended for more complex patterns. Despite these, the current implementation is robust enough for most feasible *Modelica* models.

Although the presented algorithm remains in a proof-of-concept stage—with potential improvements in the choice of optimization strategy and possible relaxation of the underlying problem formulation—the results obtained are highly promising. The algorithm has been evaluated on a wide variety of representative Modelica models, including both synthetic benchmarks and simplified real-world industrial examples. In the case of synthetic benchmarks, full resizability was assumed to stress-test the algorithm's capabilities, while for real-world industrial models, only selected array sizes were treated as resizable to reflect practical usage scenarios.

The development of the new OpenModelica backend has led to substantial performance enhancements. Experimental evaluations demonstrate an almost threefold reduction in causalization time, alongside improved scalability that enables the successful compilation of significantly larger models. The principal contribution of this work, however, lies in the introduction of an optimization-based approach for handling resizable variables. This method effectively decouples compilation time from array sizes, yielding additional performance benefits and paving the way for more efficient large-scale model compilation.

In summary, the integration of resizable variable support marks a significant step forward in improving both the robustness and efficiency of the OpenModelica compilation pipeline. Future work may explore more generalized access patterns, improved diagnostic capabilities, and broader support for dynamic model structures.

Acknowledgements

This work was conducted as part of the OpenSCALING project (Grant No. 01IS23062E) at the University of Applied Sciences and Arts Bielefeld, in collaboration with Linköping University. The authors would like to express their sincere appreciation to both the OpenSCALING project and the Open Source Modelica Consortium (OSMC) for their support, collaboration, and shared commitment to advancing open-source modeling and simulation technologies.

References

Abdelhak, Karim, Francesco Casella, and Bernhard Bachmann (2023-12). "Pseudo Array Causalization". In: *Modelica Conferences*, pp. 177–188. DOI: 10.3384/ecp204177.

Fritzson, Peter, Adrian Pop, Karim Abdelhak, et al. (2020-10). "The OpenModelica Integrated Environment for Modeling, Simulation, and Model-Based Development". In: *Modeling, Identification and Control: A Norwegian Research Bulletin* 41, pp. 241–295. DOI: 10.4173/mic.2020.4.1.

Kannan, Ravindran and Clyde L. Monma (1981). "On the Computational Complexity of Integer Programming Problems". In: *Optimization and Operations Research*. Vol. 157. Lecture Notes in Economics and Mathematical Systems. Springer,

[1] All other methods in the new backend were already array-size independent, as detailed in (Abdelhak, Casella, and Bachmann 2023)

pp. 161–172. DOI: 10.1007/978-3-642-95322-4_17. URL: https://doi.org/10.1007/978-3-642-95322-4_17.

Marzorati, Denise, Joaquín Fernández, and Ernesto Kofman (2024-06). "Efficient Matching in Large DAE Models". In: *ACM Transactions on Mathematical Software* 50. DOI: 10.1145/3674831.

Modelica Association (2023). *Modelica® Language Specification, Version 3.6*. https://specification.modelica.org/maint/3.6/MLS.pdf. Accessed: 2025-04-18.

Neumayr, Andrea and Martin Otter (2023-12). "Variable Structure System Simulation via Predefined Acausal Components". In: *Modelica Conferences*, pp. 511–520. DOI: 10.3384/ecp204511.

Modiator - A Web App for Modelica Simulation

Hilding Elmqvist[1] Martin Otter[2]

[1] Mogram AB, Sweden, hilding.elmqvist@mogram.net

[2] DLR, Institute of Vehicle Concepts, Germany, martin.otter@dlr.de

Abstract

The Modelica language (www.modelica.org) has become a de facto standard for systems modeling and many tools exist. This paper describes certain modern enhancements and a static web app implementation called *Modiator (Modelica Instant Simulator)*. It allows an improved immediate first-time user experience since the web app is available in seconds and simulations can be done directly in the browser. State of the art numerical solvers from the Sundials suite have been compiled into WebAssembly. The Modelica model is translated into Javascript code using techniques such as sorting, tearing, index reduction, state selection, etc. A subset of Modelica is supported with some extensions, for example, support for self-modifying models. This paper also presents the Stream and Model3D prototype libraries.

Keywords: Modelica, Modia, Modia3D, compilation, web app, WebAssembly, simulation in the browser, cloud computing, serverless functions, Monte Carlo simulation

1 Introduction

The Modelica language (Modelica Association 2023) makes it easy to build large, complex models since instantiation of reusable component models is possible. However, current tools[1] usually make full expansion of the equation sets including scalarization of arrays for all instances and transform the equations to C-Code that is compiled and linked. This means that translation time can be minutes. There is also a need for self-modifying models for example to support adaptable grid for finite volume models, variable model structures such as multi-stage rockets or just turning on and off subsystems. This means that separate translation would be beneficial, i.e. only translating the parts of the model that has been changed. This will also speed up experiments involving variant selection by simulation when certain subsystems are redeclared to analyze alternative designs.

Web technology allows an improved immediate first-time user experience since the web app needs no installation (and also no administrator rights), is available in seconds and faster separate translation and simulations done directly in the browser means less waiting. This will also make object-oriented modeling and simulation more accessible to students, hopefully increasing the popularity of Modelica.

Mobile phones are becoming more popular than computers and they have sufficient computing power for simulation. The drawback of mobile phones is the limited screen size (except for foldable phones). One possible solution is to use an *infinite canvas* and convenient and fast zooming, scrolling, and panning.

For multi-simulations for variant selection or Monte Carlo simulations, cloud should be possible to utilize, for example by serverless lambda functions, and there are needs to visualize results in different ways than just trend plots, such as scatter diagrams and histograms.

Many of the innovations and findings in this paper regarding translating models come from the experiences from the Modia project[2], see, e.g., (Elmqvist et al. 2021). Modia is a language with many similarities to Modelica. However, the language is much simpler than the Modelica language, and implementation of the Modia environment is based on Julia[3] (Bezanson et al. 2017). The effort is now moved from the Modia language to the Modelica language for three reasons: (1) It takes too long to get the first plot when using Modia (installing Julia and many used packages, as well as compiling generated Julia model code the first time). (2) There are many existing Modelica model libraries in wide-spread use and it is too much effort to develop and maintain a thorough translator from Modelica to Modia. (3) New technologies with WebAssembly [4] and fast just-in-time compilation of Javascript enables immediate and fast simulations.

The paper introduces *Modiator (Modelica Instant Simulator)*, a static web-app implemented with Javascript and WebAssembly providing support for a sub-set of the Modelica language with some extensions of Modelica based on experience with Modia/Julia and taking advantage of modern web technology.

The commercial Modelica environment *Modelon Impact*[5] *(Elmqvist et al. 2018)* also provides a browser GUI. However, compilation and simulation of models are not performed in the browser but via a cloud service that has to be paid for. The default behavior of Modiator is that models are compiled and simulated locally in the browser.

[1] https://modelica.org/tools/
[2] https://github.com/ModiaSim/Modia.jl
[3] https://julialang.org/
[4] https://webassembly.org/
[5] https://modelon.com/modelon-impact/

This requires different techniques, e.g., to generate model code directly in Javascript, instead of in C or C++ (as done by other Modelica tools).

Compiling and simulating a Modelica model in a web browser has been proposed by (Franke 2014) and demonstrated with a tiny, very partial prototype. (Short 2014) demonstrates a tool chain to translate Modelica models to Javascript based on the OpenModelica [6] compiler and Emscripten[7] and run the Javascript code locally in the browser. (Kulhánek et al. 2023) provide the Bodylight.js 2.0 toolchain to build in-browser web simulators by translating a Modelica model into an FMU[8] (Functional Mockup Unit), use Emscripten to transform the C-source code inside the FMU into WebAssembly format and embed this code in a web page with a specific user-friendly GUI for this particular Modelica model.

2 A Modernized Modelica Syntax

There are trends concerning the syntax of new programming languages (e.g. C#, Go, Julia, Python, Swift) which have been adopted by Modia. The following proposals for the Modelica language are inspired by Modia, based on these new trends:

- Using Greek letters in identifiers is natural for users making mathematical models.
- Semicolon is becoming optional which makes models easier to read. Traditionally, semicolons were used to enable resynchronization after the first error in order to continue parsing. The speed of parsing is now much faster, so it is better to abort after the first error is found.
- Type and unit inference and checking can be introduced if a notation is utilized to associate units to numeric literals. In the example below, units are appended to parameters and start values enclosed in single quotes. Most type specifiers are omitted since all parameters are Real and the states must be Real.
- The class identifier after end is made optional.

Below is shown a second order model utilizing the simplified, more readable syntax in Modiator:

Listing 1. Modelica model with the proposed condensed syntax and Greek identifiers.

```
model SecondOrder
  τ = 1.0 's'  // instead of: parameter Real tau(unit="s") = 1.0;
  ζ = 0.5
  input u
  x1(start=0)
  x2(start=0 '1/s')   // instead of: Real x2(start=0, unit="1/s");
  output y=x1
equation
  der(x1) = x2
  τ*τ*der(x2) + 2*ζ*τ*x2 + x1 = u
end   // instead of: end SecondOrder;
```

3 Self-Modifying Models

A much-wanted Modelica feature is to be able to change array dimensions without recompiling when, for example, higher fidelity of discretization is needed. Other needs are to be able to redeclare component classes or start and stop subsystems. These scenarios can be achieved by scripting repeated simulations where the start value for the next simulation is set to the end value of the previous simulation. At certain conditions, parameters can be changed, redeclarations made and some transformation algorithm is applied between final state and new state. Writing such scripts is not object oriented though. To make it reusable, the transformations should be made in the model classes.

A similar situation exists when updating HTML pages. The HTML DOM (Document Object Model) is a tree of various kinds of nodes which can be modified by Javascript functions which are called when different kinds of events occur. The DOM tree can also be traversed and a node id can be search for. Nodes can then be modified either by changing attributes or making a new subtree by providing an *innerText* definition. Nodes can also be added or removed by built-in JavaScript function calls.

The ModelManagement.Structure.AST functions of Dymola[9] can manipulate the AST (Abstract Syntax Tree) from Modelica scripts (not from Modelica models). This functionality can serve as inspiration for models manipulating themselves.

For this feature, Modelica needs to have an event detection mechanism (c.f. addEventListener in HTML). Modelica already has a when-statement which can be generalized. The model below shows some of the functions which can be used to manipulate the **ast** (reserved Modiator identifier for the Abstract Syntax Tree) of a model.

Listing 2. Model that modifies its definition during simulation.

```
model TestSelfModifications
  parameter Real T=1;
  Real x(start=0);
equation
  T*der(x) + x = 1;
  when x>0.3 then
    ast.setParameter('T', 0.1);
  end when;
  when time>0.5 then
    ast.setStart('x', 2);
  end when;
  when time>0.7 then
    ast.addEquation('0.1*der(y) + y = 1');
  end when;
end TestSelfModifications;
```

If a significant event occurs, the modified model ast is translated to Javascript code which is just-in-time compiled before the simulation continues.

[6] https://openmodelica.org/
[7] https://emscripten.org/
[8] https://fmi-standard.org/
[9] https://www.3ds.com/products/catia/dymola

DOI 10.3384/ecp218211

The semantics of the above function calls are explained by the plot below.

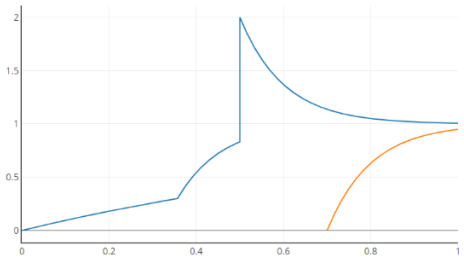


Figure 1. Plot of model TestSelfModifications

Note, that when the equation was added, also a new variable was introduced, y. By default, all state variables are currently plotted, i.e. y was also plotted during its existence. Adopted from Modia, Modiator does not require declaration of variables and the start value is 0 by default.

Array dimensions and the number of states can also be changed at events which is illustrated by the following model:

Listing 3. Model that modifies array dimensions and number of states during simulation.

```
model VaryingDimensions
   Integer n=2;
   Real x(start=1:n, fixed=true);
equation
   der(x) = -diagonal(1:n)*x;
   when time > 0.1 then
      ast.setParameter('n', 3);
      ast.setStart('x[3]', (x[1]+x[2])/2);
   end when;
   when time > 0.2 then
      ast.setParameter('n', 2);
      ast.setStart('x[1]', (x[1]+x[3])/2);
      ast.setStart('x[2]', (x[2]+x[3])/2);
   end when;
end VaryingDimensions;
```

Note, the new start values defined with ast.setStart(…) are calculated before the model is evaluated at the event. This gives a nice mechanism for transferring the state information to a new state representation. See the results below.

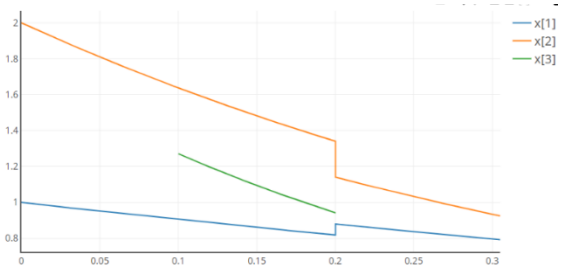


Figure 2. Plot of model VaryingDimensions.

In the companion paper on resizable arrays (Otter et al. 2025), this technique is used for adaptive grid modeling of an insulated rod.

4 Web App for Immediate Results

Web technology allows an immediate first-time user experience since a web app is available in seconds and simulations can be done directly in the browser.

Mobile phones are becoming more popular and powerful than laptop computers and they have sufficient computing power for simulation. The drawback is the limited screen size. One possible solution is to use an *infinite canvas* (inspired by Apple Photos) and convenient and fast zooming, scrolling, and panning. The DOM/CSS-feature *style.transform* [10] provides the fundamental rendering functionality to implement infinite canvas in the browser. Modiator uses two-finger touches on touch screens to zoom and move around.

The Modiator User Interface is based on a set of widgets (text editor, plots, 3D animator, model diagrams, parameter dialog, etc.) which can be freely placed and sized on the *infinite model board*, see Figure 3:

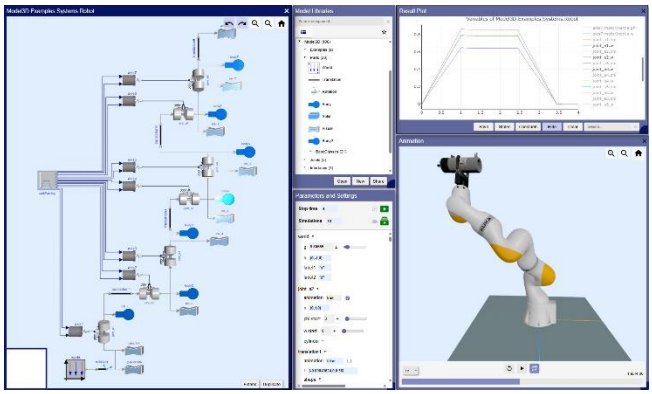


Figure 3. Example of Modiators User Interface.

The model diagram rendering is inspired by the 2.5D look of Playmola (Elmqvist et al. 2015), i.e., that icons have a 3D representation which are placed in a 2D diagram.

Multiple plot widgets can be used to visualize simulation results in different ways such as line plots, scatter diagrams, etc.

To enable AI techniques for Modelica authoring, external text editors such as VS Code with Github Copilot plug-in, can be used. When the external file is changed, Modiator will load and simulate the updated model.

The Modiator web app is a PWA (Progressive Web App), i.e., it can be installed on your device and by using service workers it can be used offline without internet connection.

[10] https://www.w3schools.com/jsref/-prop_style_transform.asp

5 Simulation in the Browser

Modiator uses a mixed execution model for simulation. Modelica models are translated to JavaScript functions which calculate the derivatives given time, states and parameters. The integration of the differential equations is performed by CVODE from the SUNDIALS suite [11] (Hindmarsh et al. 2005, Gardner et al. 2022). CVODE has been translated from C to WebAssembly [12] utilizing Emscripten [13] and is called from Javascript when simulation starts. When derivatives need to be calculated, a Javascript callback function is called from CVODE. Special considerations had to be taken especially regarding allocation and transfer of arrays.

Modiator translates the model equations to a JavaScript function by using the *new Function()* concept[14]. The Function constructor takes two arguments. The first one is a string of comma-separated arguments of the function and the second is the body of the function as a string, for example:

```
const func = new Function('a,b', 'return a + b')
console.log(func(1, 2)) // 3
```

Such functions are usually Just-In-Time-compiled. Array and matrix equations are not expanded (Otter et al. 2025), but evaluated by calls to run-time functions and math.js[15].

In order not to disturb user interactions, parsing, translation and simulations are performed in concurrent web workers.

6 Simulation on the Cloud

For multi-simulations for variant selection or Monte Carlo simulations, it should be possible to utilize cloud computations. So far, *Netlify functions* [16] have been utilized. They are based on Amazon Web Services Lambdas.

Essentially the same simulation code as for web workers can be used. Instead of posting simulation data to a web worker, *fetch* requests are made to a simulate API function with a *body* of simulation data. The simulation data contains parameters and the JavaScript function string for calculating derivatives. To support multi-simulations with redeclares or self-modifying models, the Modelica translator module is also available in the serverless functions.

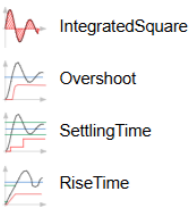
So far, limited testing has been made. 10000 simulations have been performed on the cloud. The startup time for each simulation was less than a second.

7 Monte Carlo Simulation

Model-based product design involves determining product topology, component selection, component sizing, and parameters (tuners) in such a way that the product has good performance, low production cost and low cost of ownership (objectives) while being robust with regards to variations in its environment and insensitive to variations in parameters (uncertainties). Since topology, component selection and component sizing are important, gradient based optimization tools can't be used.

Modiator uses Monte Carlo simulation and randomly selects tuner values typically from uniform distributions and selects uncertainties typically from truncated normal distributions. Redeclarations to model design selections or discrete environment variations have associated discrete distributions.

A set of Modelica *Objective* blocks is available for inserting into a model to define what are good designs. A subset of these blocks is shown on the right.



Since several possibly conflicting objectives can be defined, the *Pareto frontier* in scatter diagrams can help with compromise decisions and the *parallel axis coordinates plot* can help with assigning different weights to the objectives.

Equality and inequality constraints are enforced by including assert statements that cause simulations to fail if the specified conditions are not met. Functions of the following kind are provided for defining tuners and uncertain parameters:

```
p=uniform(default, minimum, maximum, tuner)
```

Such expressions can be used in modifiers. The *default* value is used for tools that do not provide Monte Carlo simulation in this way. *Minimum* and *maximum* gives the range and *tuner* is true for a tuner, otherwise an uncertainty is defined. The user interface has built-in menus for defining such function invocations (± button).

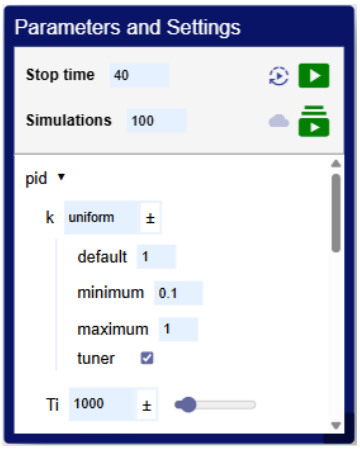


Figure 4. Tuner and Uncertainty definitions.

[11] https://webassembly.org/
[12] https://webassembly.org/
[13] https://emscripten.org/
[14] https://javascript.info/new-function
[15] https://mathjs.org/
[16] https://www.netlify.com/platform/core/functions/

DOI 10.3384/ecp218211

7.1 Design Example – Servo

To illustrate the design method used by Modiator, a well-known control problem will be employed: PID control of a double integrator. The motor consists of an ideal torque generator and an inertia and behaves like a double integrator.

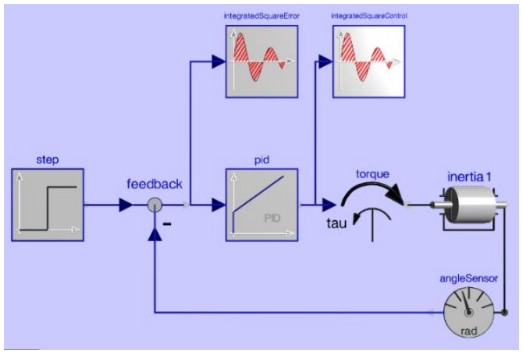


Figure 5. Controlled double integrator.

Two integral square objectives are defined. In addition to the usual objective of small control error, it is desired to keep the control effort small to save energy. The gain, k, and derivative time constant, Td, should be tuned, i.e. uniform distributions are defined and tuner is set to true for them:

```
pid(k=uniform(1, 0.1, 1, true), Td=uniform(2, 0.5, 5, true))
```

By running 100 simulations, the following spaghetti plot is obtained if angleSensor.phi and step.y were plotted during a previous single simulation:

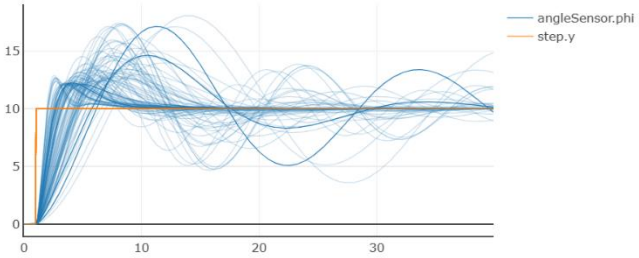


Figure 6. Spagetti plot for 100 simulations.

The time to make one thousand simulations is 2.24 seconds on a HP ZBook Ultra G1a laptop with an AMD RYZEN AI MAX+ PRO 95 3.0 GHZ CPU with 16 cores/32 threads. The corresponding time for MacBook Pro M4 is 1.56 seconds and for iPad Pro 11" M4 is 2.71 seconds. Amazingly, the same task takes only 4.32 seconds on a Samsung Galaxy Z Fold6 mobile phone.

In addition, Modiator presents a matrix of plots, see **Figure 7**. The rows and columns represent the variables: integratedSquareError.y, integratedSquareControl.y, pid.k and pid.Td. The diagonal shows the histograms for the variables (objectives: blue, tuners: green. Plot 1,2 shows the scatter plot for the objectives. The Pareto frontier is marked in red. Plot 1,3 shows that increasing gain is good for the first objective and plot 2,3 shows that decreasing gain is good for the second objective. The plots of the forth column shows that Td needs to be above 3. This is consistent with the well-known fact that the derivative term is needed for stabilization when controlling a double integrator with a PD controller.

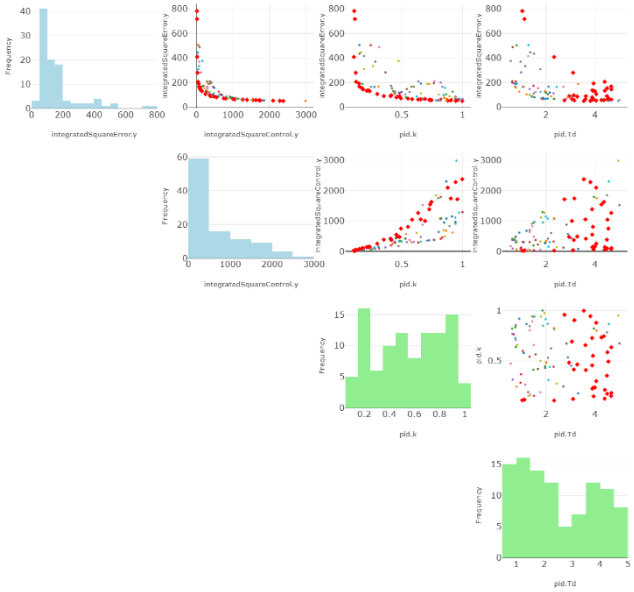


Figure 7. Multi-plot matrix with histogram and scatter plots.

By hovering over the plots, the actual values of objectives and tuners are presented:

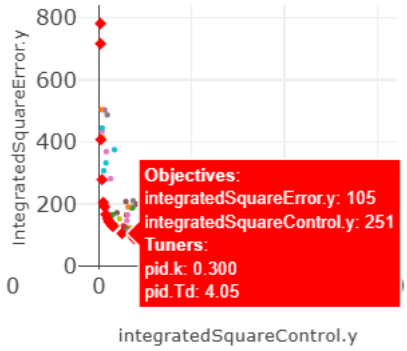


Figure 8. Tool tips showing objectives and tuners.

By zooming in on the Pareto frontier, the corresponding runs are plotted:

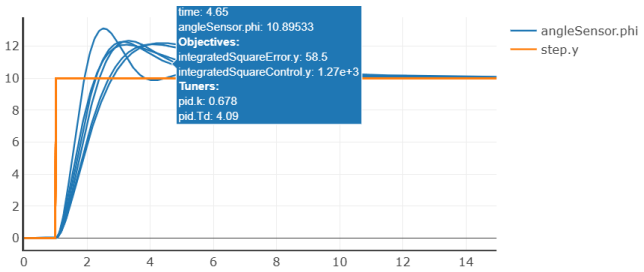


Figure 9. Selected responses.

If uncertainties are defined, for example, as a range of loads:

```
inertia1(J=uniform(1, 0.1, 1))
```

the plots include error bars and the tooltip shows the variations:

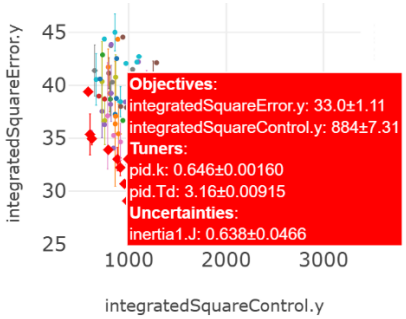


Figure 10. Scatter plot with error bars for uncertainties.

The error bars are obtained by defining bins for the tuner values and for each bin, the mean and standard deviation are presented. For accurate analysis of uncertainty influences many simulations are needed. The use of serverless functions on the cloud fits this use case.

8 Separate Translation of Models

The object-oriented modeling approach of Modelica allows building large models with millions of equations. However, the semantic specification is based on flattening, i.e., recursive cloning of variables and equations of each component instance. Typically, Modelica tools also expand matrix equations to scalar equations. Flattening means that a lot of memory is needed for variables and equations during translation of the Modelica code. It also means that translation time is long since the same analysis (flattening, symbolic processing, etc.) is performed repeatedly for each instance. Furthermore, the C-code becomes large, and compilation takes a long time. In addition, the execution of the C-code gets slower since the code might be larger than the size of the code cache.

However, parts of the equations of a component are always executed in the same order and with the same causality independently of how the component is connected. Such sequences of equations can be put into functions: less code gives shorter compilation time; less machine code gives shorter simulation time.

Finding such sequences can be made once for each model class. The technique is based on forming the most general environment to a model class as illustrated in **Figure 11**.

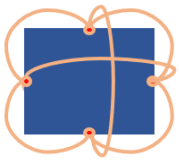


Figure 11. Generic environment of a model.

The generic environment is constructed by adding a set of dummy equations, env, which all depend on all connector variables (c_p: potential variables, c_f: flow variables) All inputs u are assumed to be known. The number of equations of env is the number of flow variables.

The Block Lower Triangular (BLT) form obtained during structural analysis of the model plus the generic environment has the structure shown in **Figure 12**.

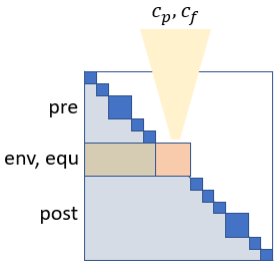


Figure 12. BLT structure of model plus generic environment.

The block in the middle contains the env equations since they all have the same incidences. It also contains certain equations, equ, which are related to the variables of the environment.

An example is given in the Appendix which shows how the equations are partitioned for a second order low pass filter.

9 Technology and Development Environment

The implementation of Modiator relies on *ECMAScript* modules[17]. Type inference and checking is done by *Visual Studio Code* [18] with *TypeScript* engine. Module dependency diagrams are generated by a *Visual Studio Code* plug-in, *Dependency Cruiser*.

The Modiator parser for Modelica is generated with the PEG-parser generator Peggy [19]. The handling of extends and merging of modifiers is done by deep merging of Javascript objects in a way inspired from merging in Modia.

Modelica model diagrams and 3D animations are created with three.js[20]. Plotting is done with Plotly[21].

Alias handling, tearing, index reduction, state-selection, BLT, symbolic processing of expressions, etc. have been ported from Modia (Julia code) to Javascript and some parts significantly improved. This has in some cases been done with a semi-automatic process using AI techniques with tool *Code Converter*[22] and the ChatGPT-4 tools *Microsoft Copilot*[23] and *Perplexity*[24]. Overall, the automatic translation reduced significantly the work to transform Modia functions to Javascript functions.

[17] https://nodejs.org/api/esm.html
[18] https://code.visualstudio.com/
[19] https://peggyjs.org/
[20] https://threejs.org/
[21] https://plotly.com/
[22] https://codeconverter.com/
[23] https://copilot.microsoft.com
[24] https://www.perplexity.ai/

DOI 10.3384/ecp218211

In order to enable building a full Modelica compiler and simulator running in the browser, C-code must be handled since Modelica has an interface to C-functions. Successful experiments have been made to use clang running under Wasmer[25] in the browser to compile C-functions into WebAssembly.

Modeling and simulation workflows typically involve scripting, for example with Python. For this reason, successful experiments have been made to use PyScript[26] and Pyodide[27] which enables running Python in the browser together with Modiator.

10 Stream with Media Propagation

When simulating fluid systems, there is currently the inherent drawback in Modelica that a medium has to be defined *at every component* and that every change of the Medium requires *recompilation* of the model. A prototype Modelica package called *Stream* does no longer has these drawbacks.

In this new approach, all media are defined with *external functions* and the internal memory of the functions by *ExternalObjects*[28] (so pointers to memory allocated inside the functions). An ExternalObject is constructed at one component and then propagated through connections. Below, first the principle of the approach is explained based on media *C-functions* and afterwards it is sketched how media *Javascript-functions* are supported in current Modiator.

Stream is a small subset of Modelica package *ThermofluidStream*[29] (Zimmer 2020, Zimmer et al. 2022) but with (a) different handling of the media inspired by (Otter et al. 2019) and (b) equations of the thermodynamic state formulated with *Linear Implicit Equilibrium Dynamics* (LIED) as proposed by Zimmer (2024, 2025). The latter means that all states are fixed in the library and no nonlinear equations appear in Modelica models due to Stream (but might be present inside the external functions). Separate translation, as proposed in section 8, becomes particularly easy in this case. A simple example of a system with boundaries, flow resistances and a volume are shown in the next figure.

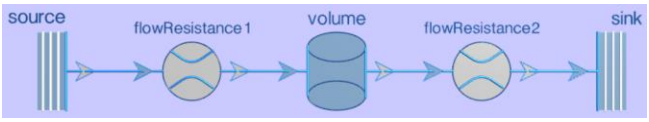


Figure 13. Simple Stream model in Modiator. The *medium* and its variables are defined in the *source* and propagated through the connections. The medium variables are changed in every component. *Media states* are defined in the *volume*. *Mass flow rate states* are defined in the *source* and in the *volume*.

The core LIED equations of the volume are shown in the next Listing.

Listing 4. Stream model of a volume with fixed size.

```
model Volume "Volume of fixed size"
  Interfaces.Inlet  inlet; Interfaces.Outlet outlet;
  parameter SI.Volume V_par "Fixed volume";
    ...
  Real xs[2](each fixed=false, each unit="1")
                                        "Scaled pair (p,T), (p,h), ...";
  Real Xi[size(Xi_start,1)](start=Xi_start, each fixed=true,
                    each unit="1") "Independent mass fractions";
protected
  parameter Integer mediumID(fixed=false);
  functions.Media.MediumAndState obj =
                    functions.Media.MediumAndState(mediumID);
  Real obj_r "Dummy, for correct sorting of obj";
initial equation
  mediumID = functions.Media.mediumID_from_obj(inlet.obj);
  xs       = functions.Media.get_xs(obj,p_start,T_start,...);
equation
  // Update obj with xs, Xi
  obj_r = functions.Media.updateThermodynamicState(obj, xs, Xi);
  outlet.obj = obj;  outlet.obj_r = obj_r;

  // Mass balance
  der_M = inlet.m_flow - outlet.m_flow ;
  der_d = der_M/V_par;

  // Energy balance
  d_in  = functions.Media.density(inlet.obj, inlet.obj_r );
  h_in  = functions.Media.specificEnthalpy(inlet.obj, inlet.obj_r );
  u_out = functions.Media.specificInternalEnergy(
                                          inlet.obj,inlet.obj_r );
  d_out = functions.Media.density(obj, obj_r);
  h_out = functions.Media.specificEnthalpy(obj, obj_r);
  M     = V_par*d_out;
  der_u = (inlet.m_flow*h_in - outlet.m_flow*h_out -
                                          der_M*u_out)/M;
  der(xs) = functions.Media.get_der_xs(obj, obj_r,der_d, der_u);

  // Mass fraction balance
  Xi_in   = functions.Media.massFractions(obj, obj_r);
  der(Xi) = (Xi_in*inlet.m_flow - Xi*(outlet.m_flow - der_M))/M;

  // Mass flow rate and inertial pressure (as in ThermofluidStream)
  ...
end Volume;
```

The state vector of a single substance compressible fluid medium is defined by two intensive quantities. In Modelica.Media, the state pairs (p,T), (p,h), (p,s), (d,T) are used. The pair selected for a particular medium in Stream is *hidden* and is implicitly defined by the medium name. In **Listing 4**, vector xs[2] is defined as a scaled pair, so that the two elements have magnitudes around 1 which means no nominal values need to be set for xs. For example, Modelica.Media.Air.SimpleAir uses the scaling xs[1] = p/10^5, xs[2] = T/273.15.

For a *multi-substance medium*, additionally the independent mass fractions Xi[:] are used as states. The dimension of Xi is identical to the dimension of its start value vector, i.e., Real Xi[size(Xi_start,1)].

[25] https://wasmer.io/
[26] https://pyscript.net/
[27] https://pyodide.org/
[28] https://specification.modelica.org/maint/3.6/functions.html, section 12.9.7
[29] https://github.com/DLR-SR/ThermofluidStream

The internal medium states and the internal medium equations are defined by the declaration of the ExternalObject:

```
MediumAndState obj = MediumAndState(mediumID);
```

The input argument to the constructor is the Integer parameter mediumID – a unique identification of a Medium. It is deduced from the inlet input variable obj that references the medium and the medium variables in the connector:

```
parameter Integer mediumID(fixed=false);
initial equation
  mediumID = functions.Media.mediumID_from_obj(inlet.obj);
```

Since inlet.obj is formally a time varying variable propagated through connectors, the mediumID is determined during initialization via the given initial equation. This is not strictly Modelica compliant because alias equations for an ExternalObject are present (such as outlet.obj = obj). This tiny issue is not detected by Dymola[30] and by Modiator. The implementation of the ExternalObject constructor is (media.c contains the C-functions of the media):

```
class MediumAndState
  extends ExternalObject;
function constructor
  input Integer mediumID "ID of medium";
  output MediumAndState obj;
  external "C" obj = MediumAndState_constructor(mediumID)
    annotation(IncludeDirectory=
                  "modelica://Stream/Resources/C-Sources",
      Include="#include \"media.c\"");
end constructor;
```

Since der(xs) and der(Xi) appear in the volume model, xs and Xi are treated as known. With statement

```
obj_r = functions.Media.updateThermodynamicState(obj, xs, Xi);
```

the states xs and Xi are copied into obj. The function returns the Real dummy variable obj_r (short for obj_ready). Whenever, an access function is called with obj, obj_r must be provided as well to guarantee that obj is used after it was updated. The derivative of xs – der(xs) – is computed with the external function call

```
der(xs) = get_der_xs(obj, obj_r, der_d, der_u);
```

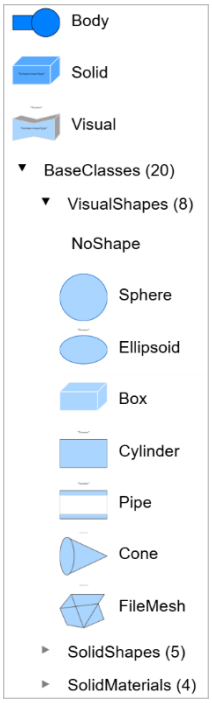
which has obj, obj_r, der_d (derivative of density) and der_u (derivative of specific internal energy) as arguments.

Modiator can currently not utilize C-functions. Therefore, Javascript functions that are equivalent to the functions in media.c are provided in file media-functions.js. This file is included whenever Modiator is started. During parsing of a Modelica model, all function names that start with "functions." are interpreted to be Javascript functions and the look-up is not done in the Modelica model but internally in Modiator.

11 Model3D with 3D Geometries

The prototype Modelica package *Model3D* enables incorporation of 3D geometries into Modelica models. Currently, this capability is applied only to 3D mechanical systems but is intended to extend to other domains in the future. The design of Model3D is inspired by Modia3D[31] (Neumayr and Otter 2018, Neumayr 2025) and Modelica.Mechanics.MultiBody[32] (Otter et al. 2003). It utilizes the new approach "Dialectic Mechanics" introduced in (Zimmer and Oldemeyer 2023) and further developed in (Zimmer 2024, 2025). In particular, the states of a multibody system are fixed in the library and are the generalized joint coordinates of the spanning tree. Kinematic loops can be optionally modelled with elastic cut-joints. By introducing "kinetic velocities" which are filtered position derivatives, very stiff elastic joints can be reasonably simulated. The *analytic* handling of many kinematic loop types supported by the MultiBody library, will be ported to Model3D.

Primitive geometries, such as box, sphere etc. as well as meshes (.stl, .obj/mtl, .dae, .glb format) can be used in a model both for visualization and to compute properties such as mass and inertia, similarly as it is done in Modia3D. A screenshot from Modiators parameter menu of the basic parts of Model3D are shown at the right side. BaseClasses are replaceable models that can be utilized in Body and Visual (VisualShapes) as well as in Solid (SolidShapes, SolidMaterials). An example of a double pendulum with object diagram and animation is shown in **Figure 14**.



Model3D stores layout and settings parameters no longer in the World and other models, but in an external hierarchical JSON file. Standard Modelica tools read this file and extract content via Modelica package *ExternData*[33] (Beutlich and Winkler 2021).

Modiator accesses the settings JSON file directly from Javascript. In order that both approaches work seamlessly together, a few access routines of ExternData are replicated in Model3D as functions.getJSONxxx. If the Modiator parser recognizes function names starting with "functions", it utilizes its Javascript implementation.

[30] https://www.3ds.com/products/catia/dymola

[31] https://github.com/ModiaSim/Modia3D.jl

[32] https://doc.modelica.org/Modelica%204.0.0/Resources/helpDymola/Modelica_Mechanics_MultiBody.html

[33] https://github.com/modelica-3rdparty/ExternData

DOI 10.3384/ecp218211

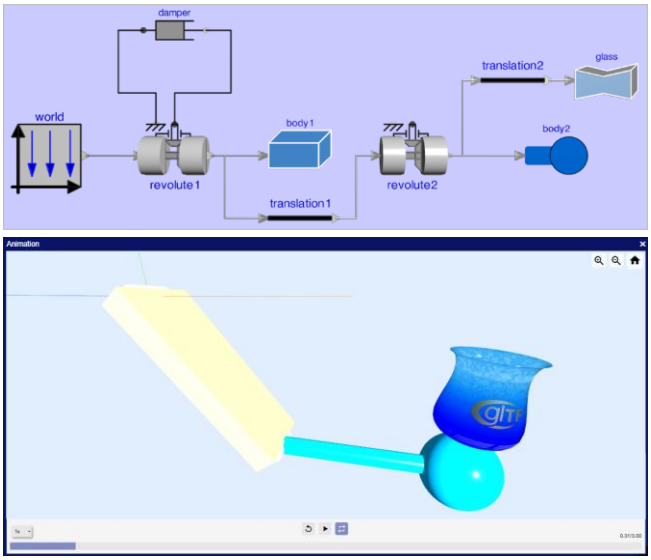


Figure 14. Model of a double pendulum in the diagram and animation widget of Modiator: *body1* is a "Solid" (here: Box) where mass properties are computed from the geometry and from the selected material, *body2* is a "Body" defined by mass, inertia and center of mass together with a "Visual" object (default: Cylinder) and optional marking of the center of mass with a sphere; *glass* is a "Visual" object (here: FileMesh).

As an example, part of the layout parameterization of a Revolute joint is shown in the next Listing.

Listing 5. Layout parametrization of Revolute joint.

```
constant String resource = Modelica.Utilities.Files.loadResource(
                  "modelica://Model3D/Resources/settings.json");
constant Real length = functions.getJSONReal(resource,
                                    "shape_in_m.jointLength",0.1);
constant Real diameter = functions.getJSONReal(resource,
                                   "shape_in_m.jointWidth",0.05);
constant Integer color[3] = functions.getJSONIntegerArray1D(
                               resource,"color.Revolute",3,{255,0,0});
  …
// settings.json file
  {"shape_in_m" : {
       "jointLength": 0.1,
       "jointWidth" : 0.05,
       "forceLength": 0.1,
       "forceWidth" : 0.05,
       "bodyDiameter": 0.11},
    "color": {
       "Revolute" : [255,0,0],
         …}
```

The functions.getJSONxxx act as a thin layer over the ExternData API. They have an additional default argument which is used if the settings file is unavailable. This technique results in much cleaner menus. Notably, all the many parameters of the World object are removed, except for the gravity definition which remains in the World object and is propagated via the connections. Consequently, global data is no longer present, so the inner prefix of World is removed.

12 Limitations and Future Work

The status of Modiator can be categorized as a Proof-Of-Concept for a complete Modelica compiler and simulator running in a web browser. So far, the focus has been to investigate an entire tool chain for the fundamental semantics of Modelica: model classes, component declarations with modifiers and redeclare, and connections.

It is sometimes assumed that the Modelica model is correct since many checks are not performed in the interest of translation speed. Scalability has not been in focus yet, i.e. only models with less than 1000 states and 5000 variables have been tested. The translator can be optimized in many ways, for example, by introducing caching and the separate translation technique outlined in section 8.

Only models which don't have nonlinear algebraic equation systems or mixed simultaneous equations with Real and Boolean unknowns can currently be simulated. Only very limited event handling is currently available. Other features not supported yet are:

- Modelica functions and external C-functions
- inner/outer
- complete set of matrix expressions
- algorithms
- initial equations/algorithms
- stream connectors (not planned)
- overconstrained connections (not planned)
- …

There are no plans to support the following parts of the Modelica Standard Library because they shall be replaced by better approaches:

(1) Modelica.Media shall be replaced by the Stream library sketched in section 10.

(2) Modelica.Fluid shall not be supported but instead the Stream library shall be used, and/or a future version of the ThermofluidStream library based on the Stream media handling and on LIED equations.

(3) Modelica.Mechanics.MultiBody shall be replaced by Model3D. Potentially, a nearly MultiBody compliant library will be provided as thin layer over Model3D to allow reasonable conversion of the many libraries and models that are based on the MultiBody library.

Since the current version of Modiator already (a) is useable on any device, (b) can be utilized just with the Modiator URL within a few seconds without installation, and (c) can translate and simulate smaller Modelica models locally in the browser nearly instantaneously, Modiator is well suited for *tutorials* and *university courses* on Modelica and for dedicated *web apps with a special user-oriented Web GUI* where the Modiator engine is running in the background.

13 Conclusions

Modiator is the next phase of experimentation and evolution of object-oriented modelling conducted by the authors after the Modia project. New web technologies provide the means to enable quick and easy access to object-oriented modeling and cloud simulation by Modelica Instant Simulation.

Furthermore, Modiator provides a platform for further investigations of new modeling capabilities to meet more and more demanding modelling needs.

Acknowledgements

The authors want to thank Johan Furuhjelm (Lund Institute of Technology) for the implementation of the infinite canvas user interface, Andreas Pfeiffer (DLR-FK) for valuable help regarding WebAssembly and Sundials and Dirk Zimmer (DLR-RM) for useful discussions regarding LIED and dialectic mechanics.

References

Bezanson, Jeff, Alan Edelman, Stefan Karpinski and Viral B. Shah (2017). "Julia: A fresh approach to numerical computing". In: *SIAM review* 59.1, pp. 65–98. DOI: 10.1137/141000671

Beutlich, Thomas and Dietmar Winkler (2021): „Efficient Parameterization of Modelica Models". In: Proceedings of the *14th International Modelica Conference*. DOI: 10.3384/ecp21181141

Elmqvist, Hilding, Alexander D. Baldwin and Simon Dahlberg (2015): "3D Schematics of Modelica Models and Gamification". In: Proceedings of *11th International Modelica Conference*, pp. 527–536. DOI: 10.3384/ecp15118527

Elmqvist, Hilding, Martin Malmheden and Johan Andreasson (2018). "A Web Architecture for Modeling and Simulation", In: Proceedings of the 2nd Japanese Modelica Conference, Tokyo, Japan, May 17-18, 2018

Elmqvist, Hilding, Martin Otter, Andrea Neumayr and Gerhard Hippmann (2021). "Modia - Equation Based Modeling and Domain Specific Algorithms". In: Proceedings of *14th International Modelica Conference*, pp. 73–86. DOI: 10.3384/ecp2118173.

Franke, Rüdiger (2014): "Client-side Modelica powered by Python or Javascript". In: *Proceedings of 10th International Modelica Conference*, pp. 1105-1112. DOI: 10.3384/ecp140961105

Gardner, David J. et al. (2022): "Enabling new flexibility in the SUNDIALS suite of nonlinear and differential/algebraic equation solvers". In: *ACM Transactions on Mathematical Software (TOMS)*. 48.3, pp. 1-24, DOI: 10.1145/3539801

Hindmarsh, Alan C. et. al. (2005): "SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers". In: *ACM Transactions on Mathematical Software (TOMS)*, 31.3, pp. 363–396. DOI: 10.1145/1089014.1089020

Kulhánek Tomáš, Arnošt Mládek, Filip Ježek and Jirí Kofránek (2023): "Bodylight.js 2.0 - Web components for FMU simulation, visualisation and animation in standard web browser". In: *Proceedings of 15th International Modelica Conference*, pp. 443-451. DOI: 10.3384/ecp204443

Modelica Association (2023). *Modelica – A Unified Object-Oriented Language for Systems Modeling, Language Specification, Version 3.6*. URL: https://specification.modelica.org/maint/3.6/MLS.html.

Neumayr, Andrea and Martin Otter (2018): "Component-Based 3D Modeling of Dynamic Systems". In: Proceedings of the *1st American Modelica Conference*. DOI: 10.3384/ECP18154175

Neumayr, Andrea (2025): *Modeling and Simulation of Physical Systems with Variable Structure*. Doctoral Dissertation, Technical University of Munich.

Otter, Martin, Hilding Elmqvist and Sven Erik Mattsson (2003): "The New Modelica MultiBody Library". In: Proceedings of the 3rd International Modelica Conference. URL: http://www.Modelica.org/Conference2003/papers.shtml

Otter, Martin, Hilding Elmqvist, Dirk Zimmer and Christopher Laughman (2019), "Thermodynamic Property and Fluid Modeling with Modern Programming Language Constructs". In: Proceedings of the *13th International Modelica Conference*, pp. 589–598. DOI: 10.3384/ecp19157589.

Otter, Martin and Hilding Elmqvist (2025), "Resizable Arrays in Object-Oriented Modeling", In: Proceedings of the *16th International Modelica and FMI Conference.*

Short, Tom (2014): *OpenModelica models in Javascript*. URL: https://github.com/tshort/openmodelica-javascript

Zimmer, Dirk (2020): "Robust object-oriented formulation of directed thermofluid stream networks". In: *Mathematical and Computer Modelling of Dynamical Systems*", Taylor & Francis. DOI: 10.1080/13873954.2020.1757726

Zimmer, Dirk, Michael Meißner and Niels Weber (2022): The DLR ThermoFluid Stream Library. Electronics, Vol. 11, nr. 22, DOI: 2079-9292/11/22/3790

Zimmer, Dirk and Carsten Oldemeyer (2023): "Introducing Dialectic Mechanics". In: Proceedings of the *15th International Modelica Conference 2023*, pp. 167–176. DOI: 10.3384/ecp204167.

Zimmer, Dirk (2024): "Object-Oriented Implementation of a Simulator for Linear Implicit Equilibrium Dynamics". In: *ASIM 2024 Tagungsband*. DOI: 10.11128/arep.47.a4735.

Zimmer, Dirk (2025): "The Value of Enforcing a Strict Modeling Methodology within Modelica". In: Proceedings of the *16th International Modelica & FMI Conference.*

Appendix
Example for Separate Translation

This appendix shows an example of separate translation of models as sketched in section 8. Consider the second order low pass filter[34] in **Figure 15**

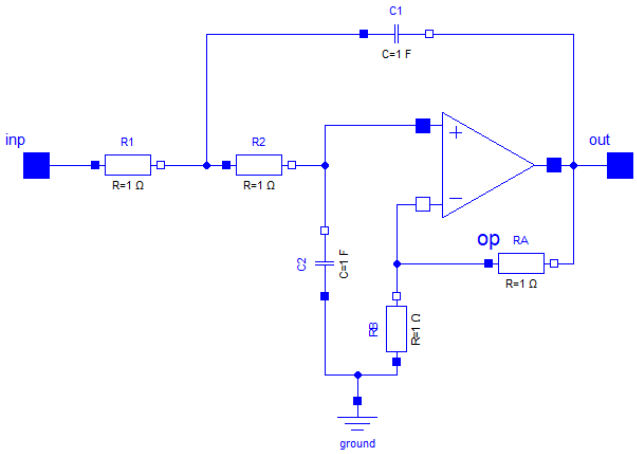


Figure 15. Second order low pass filter.

A corresponding Modelica model is shown below.

Listing 6. Modelica model of **Figure 15**.

```
model LowPass2nd
  Modelica.Electrical.Analog.Basic.Resistor R1(R=1);
  Modelica.Electrical.Analog.Basic.Resistor R2(R=1);
  Modelica.Electrical.Analog.Ideal.IdealOpAmp3Pin op;
  Modelica.Electrical.Analog.Basic.Ground ground;
  Modelica.Electrical.Analog.Basic.Capacitor C1(C=1);
  Modelica.Electrical.Analog.Basic.Capacitor C2(C=1);
  Modelica.Electrical.Analog.Basic.Resistor RA(R=1);
  Modelica.Electrical.Analog.Basic.Resistor RB(R=1);
  Modelica.Electrical.Analog.Interfaces.Pin inp;
  Modelica.Electrical.Analog.Interfaces.Pin out;
equation
  connect(R1.n, R2.p);
  connect(R2.n, C2.n);
  connect(ground.p, C2.p);
  connect(op.out, RA.n);
  connect(ground.p, RB.p);
  connect(R1.n, C1.p);
  connect(C1.n, op.out);
  connect(R2.n, op.in_p);
  connect(RB.n, op.in_n);
  connect(RA.p, RB.n);
  connect(R1.p, inp);
  connect(op.out, out);
end LowPass2nd;
```

If a voltage source is connected between ground and connector inp, the translated model with alias elimination applied will consist of a sequence of 10 solved equations.

If the filter is separately translated according to the algorithm outlined in section 8, an equivalent Modelica model is created (parameter handling simplified):

Listing 7. Modelica model of **Listing 6** transformed into a mix of acausal equations and causal functions.

```
model LowPass2nd
  Modelica.Electrical.Analog.Interfaces.Pin inp;
  Modelica.Electrical.Analog.Interfaces.Pin out;
equation
  LowPass2nd_init = ast.newFunction("C1_v, C2_v",
   "return {
     RA: {R: 1, p:{}, n:{}},
     RB: {R: 1, p:{}, n:{}},
     R1: {R: 1, p:{}, n:{}},
     R2: {R: 1, p:{}, n:{}},
     C1: {C: 1, v: C1_v, p:{}, n:{}},
     C2: {C: 1, v: C2_v, p:{}, n:{}},
     op: {in_p: {}, in_n: {}, out: {}},
     ground:{p:{}},
     out: {}}");

  LowPass2nd_pre = ast.newFunction("M", "
     M.RB.i = M.C2.v / M.RB.R;
     M.RA.v = M.RA.R * (-M.RB.i);
     M.RA.n.v = -((M.RA.v - M.C2.v));
     M.C1.p.v = M.C1.v - (-M.RA.n.v);
     return [M.RA.n.v, M.C1.p.v];");

  LowPass2nd_post = ast.newFunction("M, R1_p_i", "
     M.R2.v = M.C1.p.v - M.C2.v;
     M.R2.i = M.R2.v / M.R2.R;
     M.C1.p.i = -((-R1_p_i + M.R2.i));
     M.C1.der_v = M.C1.p.i / M.C1.C;
     M.C2.der_v = M.R2.i / M.C2.C;
     return [M.C1.der_v, M.C2.der_v];");

     M = LowPass2nd_init(C1_v, C2_v);
     (out.v, R1_n_v) = LowPass2nd_pre(M);

     R1_R = 1;
     R1_R * inp.i = R1_v;
     R1_v = inp.v - R1_n_v;

     (der(C1_v), der(C2_v)) = LowPass2nd_post(M, inp.i);
end LowPass2nd;
```

The equations are then partitioned into

- function LowPass2nd_pre with 4 assignments,
- function LowPass2nd_post with 5 assignments,
- and 2 remaining model equations.

If many instances of LowPass2nd are created, the 9 equations of the functions LowPass2nd_pre and LowPass2nd_post need to be compiled one time.

The functions LowPass2nd_... were defined using a built-in function ast.newFunction with the same semantics as the Javascript constructor Function().

[34] https://electronics.stackexchange.com/questions/271857/second-order-low-pass-filter-configuration

DOI
10.3384/ecp218211

OpenIWPI: Open-Instance Wave-Phasor Interface Library for Power System Simulation Studies in Modelica

Marcelo de Castro[1] Luigi Vanfretti[2]

[1]Mitsubishi Electric Power Products, Inc., USA marcelo.decastro@meppi.com
[2]Rensselaer Polytechnic Institute, USA vanfrl@rpi.edu

Abstract

In the past decades, a large number of power electronic converter-based energy resources have been connected to grids around the world. The increasing number of such devices is pushing for both electromechanical and electromagnetic dynamic simulations to be part of routine stability studies for transmission planning studies. Hybrid wave-phasor simulations have been proposed in the literature to address this challenge. This paper describes Open-Instance Wave-Phasor Interface (OpenIWPI), a Modelica library used to couple phasor-based and electromagnetic power system models, allowing simulation and linearization of such hybrid models to be performed altogether. Examples are presented to describe how the library can be useful in power system studies.

Keywords: power grid, power systems, electromagnetic transients, phasor-based simulation, RMS simulation

1 Introduction

1.1 Background and Motivation

The planning of the grid and its safe operation requires many different simulation-based studies to be performed. The dynamic stability analysis of electric grids is one of these studies, and it is usually assessed using two different approaches, depending on the time scale of the phenomena that need to be assessed. In electromechanical transient stability (usually named just Transient Stability, or TS) simulations, the power system is represented in its positive sequence component equivalent to the fundamental frequency (Kundur, Balu, and Lauby 1994). In essence, a phasor-based representation of the power system is used to create a system of differential and algebraic equations (DAEs). The time scale of electromechanical system dynamics is in the order of milliseconds to tens of seconds and, with appropriate time step selection, a large number of DAEs can be solved via low-order integration methods with high computational efficiency (Kundur, Balu, and Lauby 1994; Chow and Sanchez-Gasca 2020). Electromagnetic transient stability (EMT) simulations have more detailed models of the components of the power system, capturing ultra-fast switching dynamics due to power line tripping or those of the power electronic converter (Dommel 1986; Watson and Arrillaga 2003; Mahseredjian, Venkata Dinavahi, and Martinez 2009). Once again, with an appropriate time step selection, the differential equations describing the EMT simulation are solved for a large frequency spectrum using trapezoidal integration methods. However, the time scale of electromagnetic phenomena is very small, and solving these large systems of differential equations with very small time steps is cumbersome for computers, limiting the study of EMT simulations in extremely large systems (Watson and Arrillaga 2003). Moreover, the information required for the parameterization of EMT models is substantially larger than that used for phasor-based models, and some of this information may not be available, making it challenging to develop large-scale grid models, with some exceptions (Mahseredjian, Venkata Dinavahi, and Martinez 2009; Dennetiere et al. 2019; Thapa et al. 2023).

Traditionally, TS and EMT studies were carried out separately, in different tools. However, there have already been efforts, since the 1980s, for hybrid EMT-TS simulations motivated by the need to understand the interaction between high voltage direct current (HVdc) transmission lines and the power grid (Heffernan et al. 1981; Adapa and Reeve 1988). In recent decades, the electric power system has grown not only in size but also in complexity. Currently, many converter-based resources are operating in modern power systems, and many more will be in service in the next years. These ubiquitous devices deal with large amounts of power and are crucial for the operation of the current power grid. Therefore, it is important for studies to consider how the electromagnetic response of such devices can affect the electromechanical transient stability of the entire system and vice versa.

Many power system libraries have already been developed in Modelica for both TS and EMT studies (Winkler 2017; de Castro, Winkler, et al. 2023; Masoom, Ould-Bachir, et al. 2020; Masoom, Mahseredjian, et al. 2021). Engineers have created these libraries to explore the rich features of Modelica (Modelica Association 2025) while performing EMT or TS studies. Therefore, it is attractive to explore how these can be combined and exploited for hybrid EMT-TS simulations without the typical drawbacks of co-simulation (Schweiger et al. 2019). To this end, hybrid wave-phasor interfaces using Modelica would enable one to adequately model and simulate modern power systems that are rich in converter-based re-

sources (Chang and Vanfretti 2024; Fachini et al. 2025).

Furthermore, in recent years hybrid EMT-TS studies have become more common (Watson and Arrillaga 2003). However, they are usually conducted mainly as co-simulation, with the different parts being simulated as if they were standing alone (Heffernan et al. 1981; Adapa and Reeve 1988; Theodoro, Tomim, Lima, et al. 2017; Theodoro, Tomim, Barbosa, Lima, and Santiago 2018; Theodoro, Tomim, Barbosa, Lima, and Barros 2019). While the EMT part is discretized to be solved using trapezoidal integration method, the TS part uses modified Euler algorithms. This is not done explicitly; instead, the dominant approach used by the power systems community over the last 15 years (V. Dinavahi et al. 2009) is to interface two separate environments that exchange data at specific time instants, i.e., *ad-hoc* co-simulation. However, using Modelica, hybrid EMT-TS simulation is possible without the need of two different tools — one can create a single hybrid EMT-TS model that will be solved using variable step-size algorithms (Gibert et al. 2018; de Castro and Vanfretti 2022), such as DASSL for example (Petzold 1982). Meanwhile, if needed, co-simulation can also be achieved if desired, by exporting portions of the proposed interface and models as a Functional Mock-up Unit (FMU) and interfacing them with FMI-compliant tools (Ramirez-Gonzalez et al. 2024).

It is also worth noting that the ability to create a single hybrid EMT-TS model in Modelica provides a unique added value: it allows the user to perform a small signal stability analysis using linearized models, e.g., by using the `Modelica_LinearSystems2` (Baur, Otter, and Thiele 2009). This is simply not possible with existing power system EMT simulators and certainly not feasible with EMT-TS co-simulation frameworks. Observe that even most TS simulators require additional tools for this analysis task (Martins et al. 2000; Nikolaev et al. 2020).

1.2 Objectives

In this paper, the focus will be in presenting a Modelica library that contains an interface that can be used for hybrid EMT-TS simulations. This wave phasor interface is designed to be coupled with the Open Instance Power Systems Library (OpenIPSL) (de Castro, Winkler, et al. 2023) and the Modelica Standard Library (MSL). While the phasor-based portion of a system can be represented by OpenIPSL, the electromagnetic behavior is represented by the appropriate Electrical sub-package within the MSL. The wave-phasor interface present in this library is based on a positive-sequence component filtering approach (Theodoro, Tomim, Lima, et al. 2017; Theodoro, Tomim, Barbosa, Lima, and Santiago 2018; Theodoro, Tomim, Barbosa, Lima, and Barros 2019) originally developed for co-simulation.

1.3 Paper Structure

The remainder of this paper is structured as follows: Section 2 presents the structure of the library and its organization; Section 3 shows the auxiliary models that are used in the library; Section 4 describes the mathematical description of the wave-phasor interface and its implementation in Modelica; Section 5 brings two simulation examples and their results; Final discussion and conclusions are presented in Section 6.

2 Library Overview and Organization

The Open-Instance Phasor-Wave Interface (OpenIWPI) library was developed to help engineers simulate hybrid power system models composed of phasor-based circuits and detailed components represented in the electromagnetic-transient fashion. The main idea is to be able to interface detailed components, using EMT-like representations, with power system grids, which are usually modeled in their phasor-domain positive sequence equivalents. In other words, this library should help engineers in creating hybrid EMT-TS models and simulation using a wave phasor interface (Theodoro, Tomim, Lima, et al. 2017; Theodoro, Tomim, Barbosa, Lima, and Santiago 2018; Theodoro, Tomim, Barbosa, Lima, and Barros 2019) implemented in Modelica (de Castro and Vanfretti 2022).

The interface uses electrical connectors from the OpenIPSL (de Castro, Winkler, et al. 2023) and from the Modelica Standard Library - MSL. Therefore, the library is ideal for interconnecting phasor-based representations built using the OpenIPSL, for TS studies, with EMT-like models that are built using the Electrical library within the MSL. As an outcome, the user will need to have both libraries loaded (OpenIPSL and MSL) before loading the OpenIWPI library. The OpenIWPI library has six sub-packages as shown in Figure 1.

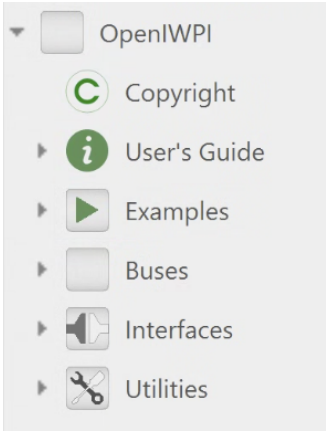


Figure 1. Library sub-packages overview.

The sub-packages are briefly described below:

- **Copyright:** the library's copyright notice informing users that the library is subject to a 3-clause BSD license.
- **User's Guide:** package containing informative items for the users, such as a starting guide and a list of

DOI 10.3384/ecp218223

reference papers used by the library.

- **Examples:** package to demonstrate the application of the interface using simple examples of the power system for simulation.
- **Buses:** contains the electrical bus that uses the hybrid wave-phasor interface to connect phasor-based models with EMT-like models.
- **Interfaces:** contains the filter-based wave-phasor interface that allows the simulation of hybrid wave-phasor (EMT-TS) circuits.
- **Utilities:** contains auxiliary models that are used in the library, such as a Phase-Locked Loop (PLL) model, reference frame transformations, as well as resettable and second-order generalized integrators.

The OpenIWPI has two important sub-packages imported from its dependencies. They are listed in the library's main `package.mo` file as shown below. The first import is related to OpenIPSL's `Types`, which allows OpenIWPI to maintain dimension consistency in calculations made in the library. The second import is related to the mathematical constants listed in MSL and that are used in the calculations performed within the OpenIWPI.

Listing 1. Importing OpenIPSL Types and Modelica Constants.

```
import OpenIPSL.Types;
import C = Modelica.Constants;
```

3 Auxiliary Models and Blocks

The `Utilities` sub-package of the OpenIWPI library is composed of many auxiliary blocks and models that are essential for assembling the wave-phasor interface. The following subsections describe each auxiliary block/-model in detail, highlighting its contribution to the proper operation of the interface.

3.1 Reference Frame Transformations

In the library, two reference frame transformations are needed: Clarke and Park transformations. Both transformations are implemented as models in the OpenIWPI library. The Clarke transformation model uses three real inputs (v_A, v_B, and v_C) and outputs two real variables (v_α and v_β) (Yazdani and Iravani 2010). In Modelica, this transformation is implemented as shown in Listing 2.

Listing 2. Clarke transformation implementation.

```
equation
    v_alpha = (2/3)*(v_A - (1/2)*v_B -
        (1/2)*v_C);
    v_beta = (2/3)*(sqrt(3)/2)*(v_B - v_C);
```

Similarly, the Park transformation model uses three real inputs (v_α, v_β and δ) and outputs two real variables (v_d and v_q) (Yazdani and Iravani 2010). The Park transformation transforms a reference frame composed of two waves in quadrature ($\alpha\beta$) into a set of two stationary values (dq) using a reference frame that rotates with the grid's frequency. Its Modelica implementation is as depicted in Listing 3.

Listing 3. Park transformation implementation.

```
equation
    v_d = v_alpha*Modelica.Math.cos(delta)
        + v_beta*Modelica.Math.sin(delta);
    v_q = -v_alpha*Modelica.Math.sin(delta)
         + v_beta*Modelica.Math.cos(delta);
```

3.2 Resettable Integrator

The integrator with a reset option was created in order to be used in the PLL present in the library. This integrator is a re-implementation of the regular integrator used in the MSL. The only difference is that it resets its output every time it reaches the value of 2π. The reason behind this reset is that it allows the output to always be within one angular revolution, avoiding an unnecessary large-angle variable to be used in calculations. This reset integrator is common in hardware implementations of PLLs based on the Synchronous Reference Frame (SRF) to avoid an integer overflow issue in the digital signal processor. In Modelica, the resettable integrator equations are implemented as shown in Listing 4. Note that this integrator uses the same parameters as the basic integrator found in MSL. Also note that Listing 4 is built under the assumption that `y_start` is 0, and that `k*u` is greater than 0.

Listing 4. Resettable integrator's equations in Modelica.

```
initial equation
    y = y_start;
equation
    der(y) = k*u;
    when y > 2*C.pi then
        reinit(y,y_start);
    end when;
```

3.3 Second-Order Generalized Integrator

The Second-Order Generalized Integrator (SOGI) (Rodriguez et al. 2006) is a filter with the purpose of generating two signals in quadrature with each other. The input of the filter, v_{in}, is filtered, and its fundamental component is output as v. The second output is a signal in quadrature with the first output and is named qv. If the transfer function between v_{in} and v is called $D(s)$ and the transfer function between v_{in} and qv is $Q(s)$, then it can be described as Equation (1) (Rodriguez et al. 2006; de Castro and Vanfretti 2022).

$$\begin{cases} D(s) &= \dfrac{v}{v_{in}} = \dfrac{k\omega_s s}{s^2 + k\omega_s s + \omega_s^2}, \\ Q(s) &= \dfrac{qv}{v_{in}} = \dfrac{k\omega_s^2}{s^2 + k\omega_s s + \omega_s^2}, \end{cases} \tag{1}$$

where the gain $\omega_s = 2\pi f_s$ is the angular frequency of the wave that must be filtered and k is a gain that can be adjusted to change the characteristics of the filter. These

two parameters can be set by the user. In this application, $f_s = 60$ hertz, while the gain is set to $k = \sqrt{2}$ because it provides a dynamic response with a good compromise between the settling time and the overshoot of the filtering process under disturbances (Rodriguez et al. 2006; de Castro and Vanfretti 2022). Note that $D(s)$ describes a band-pass filter, while $Q(s)$ describes a low-pass filter. In OpenIWPI, the SOGI has been implemented as a block diagram. It is shown in Figure 2.

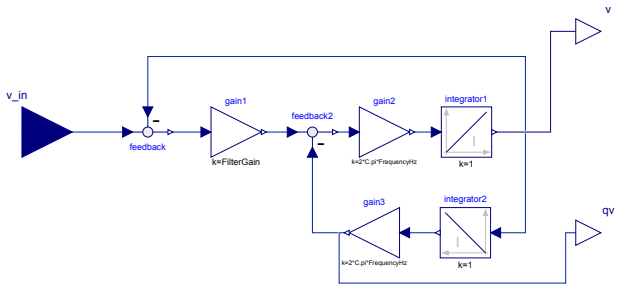


Figure 2. Sencond-order generalized integrator implementation.

3.4 Phase-Locked Loop

The PLL modeled in the library is based on a Synchronous Reference Frame (SRF) loop with double SOGI filters, described below (Rodriguez et al. 2006). The PLL can be divided into two main parts. The first part acts as a positive sequence filter, having three-phase quantities as input (voltage or current), and having as output the positive sequence equivalent for the corresponding $\alpha\beta$ components of the input quantities (Rodriguez et al. 2006; de Castro and Vanfretti 2022).

Suppose the three-phase quantities v_A, v_B, and v_C. They are used as inputs for a Clarke block. Each output, say $v_{\alpha,in}$ and $v_{\beta,in}$, is used as the input of a SOGI filter. The SOGI filter that receives the α component, there will be two outputs: v_α and qv_α. Similarly, in the case of the SOGI that filters the β component, the outputs are v_β and qv_β. The four variables can be used to calculate the positive sequence component in the $\alpha\beta$ reference frame as shown in Equation (2) (Rodriguez et al. 2006; de Castro and Vanfretti 2022). The output of this positive sequence calculation can then be used in a SRF loop designed to track one of the rotating axes. The DSOGI-based SRF-PLL (DSOGI-PLL) block diagram is shown in Figure 3.

$$\begin{cases} v_\alpha^+ = \frac{1}{2}\left(v_\alpha - qv_\beta\right) \\ v_\beta^+ = \frac{1}{2}\left(v_\beta + qv_\alpha\right) \end{cases} \tag{2}$$

The SRF-PLL (Kaura and Blasco 1997) uses a Park model and PI control to synchronize the rotating frame's and the $\alpha\beta$ positive sequence component's frequencies, such that the Park model's outputs are stationary. The SRF-PLL is described in Figure 4.

The compensator $C(s)$ is usually a PI controller with Kp and Ki that can be defined as

$$\begin{cases} K_p = & \frac{2\omega_n\xi}{\hat{X}} \\ K_i = & \frac{\omega_n^2}{\hat{X}}, \end{cases} \tag{3}$$

where $\hat{X}$ is the nominal peak value of the filtered wave, while ξ is the damping factor and ω_n is the cut-off frequency. These parameters can be tuned by the user and, for 60 hertz waves, $\xi = 0.7$ and $\omega_n = 100$ rad/s result in good dynamic performance (Rodriguez et al. 2006).

The implementation of DSOGI-PLL is carried out using block diagrams in Modelica, as shown in Figure 5. Note that the integrator located at the bottom right in the block diagram is the resettable integrator described in previous sections. This means that the output `delta` is always in the interval between 0 and 2π.

4 Wave-Phasor Interface Model

The integration between models created under different paradigms, that is, EMT and TS-like models, requires a component capable of adequately interfacing both representations. In this approach (Theodoro, Tomim, Lima, et al. 2017; Theodoro, Tomim, Barbosa, Lima, and Santiago 2018; Theodoro, Tomim, Barbosa, Lima, and Barros 2019), the TS-type power system with phasor-based models acts similarly to a variable voltage source on the interface bus. Transformation of a voltage phasor into a wave-like voltage source requires a phasor-to-wave transformation. In contrast, the EMT-type power system model will act as a variable current injection on the interface bus. This phasor-type current injection must be calculated from the waveforms obtained from the detailed EMT-type model, and thus a wave-to-phasor transformation is needed.

4.1 Voltage Phasor-to-Wave

This interface takes all necessary information from a phasor and translates it into waveform equations. The phasor information is carried out through OpenIPSL's `connector` with a Cartesian representation. The phasor $\tilde{V}$ is represented in OpenIPSL as

$$\tilde{V} = v_{re} + jv_{im}. \tag{4}$$

Hence, given the nominal voltage of the bus V_b and the base frequency of the system f_s, it is possible to calculate the wave information (amplitude A, frequency ω_s, and angle ϕ_v) as

$$\begin{cases} A = \frac{V_b\sqrt{2}}{\sqrt{3}}\left(\sqrt{v_{re}^2 + v_{im}^2}\right), \\ \omega_s = \omega_0 + \dfrac{d\phi_v}{dt}, \\ \phi_v = tg^{-1}\left(\dfrac{v_{im}}{v_{re}}\right), \end{cases} \tag{5}$$

where $\omega_0 = 2\pi f_s$. The calculated angle ϕ_v is assumed to be the phase from phase A's voltage, and the angles for

DOI 10.3384/ecp218223

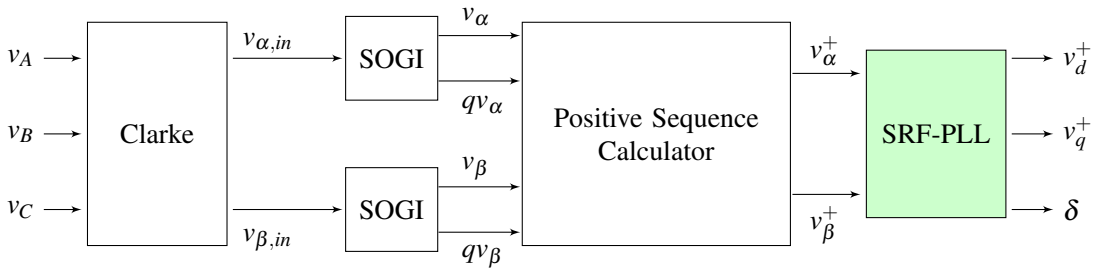


Figure 3. Diagram for the DSOGI-PLL.

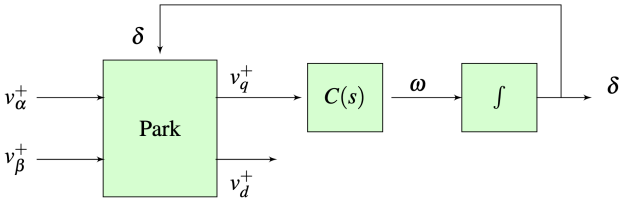


Figure 4. Diagram representing a PLL based on Synchronous Reference Frame (SRF).

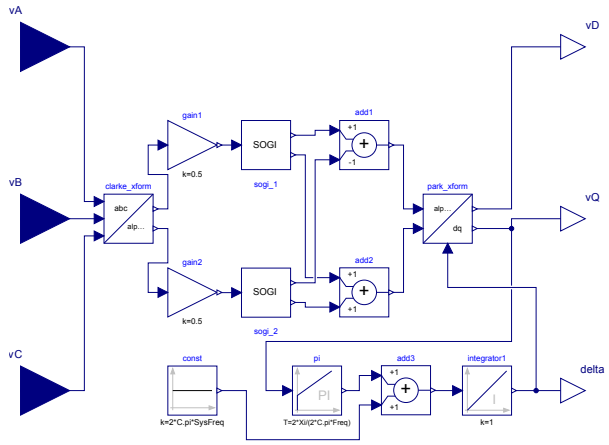


Figure 5. Modelica implementation of the DSOGI-PLL.

the other phase voltages are calculated displacing phase B and C of $\frac{2\pi}{3}$ radians. Hence, the three-phase waves that will be apparent for the EMT portion of the interface can be described as

$$\begin{cases} v_A = A\cos\left(\omega_s t + \phi_v\right), \\ v_B = A\cos\left(\omega_s t + \phi_v - \dfrac{2\pi}{3}\right), \\ v_C = A\cos\left(\omega_s t + \phi_v + \dfrac{2\pi}{3}\right). \end{cases} \tag{6}$$

4.2 Current Wave-to-Phasor

This part of the interface uses a filter to find the current phasor that will be shown at the phasor-domain portion of the interface. This can be done by using two PLLs: one DSOGI-PLL to be coupled with voltage waves described in Equation (6), and one modified PLL coupled with the currents that flow in the EMT portion of the interface. The DSOGI-PLL, coupled with voltage waves, is used to find the δ that varies with the appropriate speed ω_s. The second PLL can be considered as a modified DSOGI-PLL and it uses the current waves as inputs. The first modification is that the PLL can use a DSOGI filter as it is shown in Figure 3, but it could also use other filters to obtain i_α^+ and i_β^+. The second modification comes from the fact that the SRF-PLL part, as depicted in Figure 4 is reduced to a simple Park block that has inputs i_α^+ and i_β^+ as well as δ which is the output from the DSOGI-PLL coupled to the voltage waves. This procedure yields a current phasor

$$\tilde{I} = i_d^+ + j i_q^+. \tag{7}$$

where i_d^+ and i_q^+ are the outputs of the Park block of the modified PLL coupled to current waves. However, the angle δ is such that

$$\left(v_{re} + j v_{im}\right) e^{-j\delta} = v_d + j v_q = v_d. \tag{8}$$

Therefore, the current phasor $i_d^+ + j i_q^+$ produces an angle difference between the voltage and the current phasors. Consequently, to calculate the appropriate positive sequence current phasor value, it must be displaced from the voltage angle ϕ_v (see Equation (6)).

4.3 Modelica Implementation

In Modelica, the model contains one `connector` from the OpenIPSL and four from the MSL (three connectors for the voltage phases and one connector for the ground).

Listing 5. Connector instantiation.

```
// pins for wave signal:
Modelica.Electrical.Analog.Interfaces.
    PositivePin PinA;
Modelica.Electrical.Analog.Interfaces.
    PositivePin PinB;
Modelica.Electrical.Analog.Interfaces.
    PositivePin PinC;
```

```
Modelica.Electrical.Analog.Interfaces.
    NegativePin PinGrd;
// pin for phasor signal
OpenIPSL.Interfaces.PwPin PSPin;
```

Three system parameters are necessary for the calculations and they are entered as parameters for the interface as shown below.

Listing 6. System parameter declaration.

```
parameter Types.ApparentPower S_b = 100e6 "
    System base power";
parameter Types.Voltage V_b = 230e3 "Bus
    base voltage";
parameter Types.Frequency fn = 60 "System
    frequency";
```

The DSOGI-PLL from Figure 3 as well as a derivative block from MSL are instantiated in the model as presented below.

Listing 7. Instantiation of PLL and Derivative block.

```
OpenIWPI.Utilities.PhaseLockedLoop.Dsogi
    pll (SysFreq = fn, VLine = V_b);
Modelica.Blocks.Continuous.Derivative derA(
    T=0.1, y_start=0);
```

In addition, variables that are used in internal calculations are declared as protected. Note that the variables `type` are assigned attributes from those in `OpenIPSL.Types`.

Listing 8. Protected variables.

```
protected
    Types.AngularVelocity w
    // voltage variables
    Types.Voltage MagV;
    Types.Angle AngV;
    // current variables
    Types.PerUnit ir;
    Types.PerUnit ii;
    Types.PerUnit MagI;
    Types.Angle AngI
```

In the **`equation`** environment, the voltage phasor-to-wave transformation can be implemented in Modelica as

Listing 9. Voltage phasor-to-wave implementation

```
equation
    // voltage information from phasors
    MagV = V_b*sqrt(2/3)*sqrt(PSPin.vr^2+
        PSPin.vi^2);
    AngV = Modelica.Math.atan2(PSPin.vi,
        PSPin.vr);
    derA.u = AngV;
    w = 2*C.pi*fn + derA.y;
    // voltage waves
    PinA.v-PinGrd.v = MagV*Modelica.Math.
        sin(w*time+AngV);
    PinB.v-PinGrd.v = MagV*Modelica.Math.
        sin(w*time+AngV-2*C.pi/3);
    PinC.v-PinGrd.v = MagV*Modelica.Math.
        sin(w*time+AngV+2*C.pi/3);
```

The implementation of the current equations is described using Listing 10. The Clarke transformation result, `ialpha` and `ibeta`, is instantiated as `RealOutput`, while the inputs for the Park transformation, `iapos` and `ibpos`, are instantiated as `RealInput`. This allows the user to create other hybrid buses by using other filters and methods to extract the positive sequence of the $\alpha\beta$ frame.

Listing 10. Current wave-to-phasor implementation

```
equation
    // voltage PLL synchronization
    pll.vA = PinA.v-PinGrd.v;
    pll.vB = PinB.v-PinGrd.v;
    pll.vC = PinC.v-PinGrd.v;
    // clarke transformation
    ialpha = (2/3)*(PinA.i - (1/2)*PinB.i -
        (1/2)*PinC.i);
    ibeta = (2/3)*((sqrt(3)/2)*PinB.i - (
        sqrt(3)/2)*PinC.i);
    // Real and imaginary parts of the
        current phasor using Park transf.
    ir = -sqrt(3/2)*(V_b/S_b)*(iapos*cos(
        pll.delta)+ibpos*sin(pll.delta));
    ii = sqrt(3/2)*(V_b/S_b)*(iapos*sin(pll
        .delta)-ibpos*cos(pll.delta));
    // current magnitude and angle
    MagI = sqrt(ii^2+ir^2);
    AngIV = Modelica.Math.atan2(ii,ir);
    // grounding
    PinGrd.i = - (PinA.i+PinB.i+PinC.i);
    // current phasor waves
    PSPin.ir = MagI*Modelica.Math.cos(AngIV
        +AngV);
    PSPin.ii = MagI*Modelica.Math.sin(AngIV
        +AngV);
```

4.4 Hybrid Bus

Finally, the hybrid bus model is assembled using the double SOGI strategy, followed by the use of Equation (2), similar to what is used in the DSOGI-PLL (Theodoro, Tomim, Lima, et al. 2017; Theodoro, Tomim, Barbosa, Lima, and Santiago 2018; Theodoro, Tomim, Barbosa, Lima, and Barros 2019). The hybrid bus model is implemented in the library as shown in Figure 6. Note that the `pwPin` on the left hand side is that of OpenIPSL; while those on the right hand side (`pin_A, pin_B, pin_C`) and the bottom (`pin_grd`) are those from the MSL.

5 Simulation Examples

The OpenIWPI library has two simple examples that are used to show the appropriate behavior of the wave-phasor interface.

5.1 Simple Circuit

This first example shows a comparison between a power system completely represented in phasor-domain models and a circuit that is partially represented in phasor-domain and partially in EMT models. Figure 7 shows the two circuits that are present in the example: the upper circuit is

DOI
10.3384/ecp218223

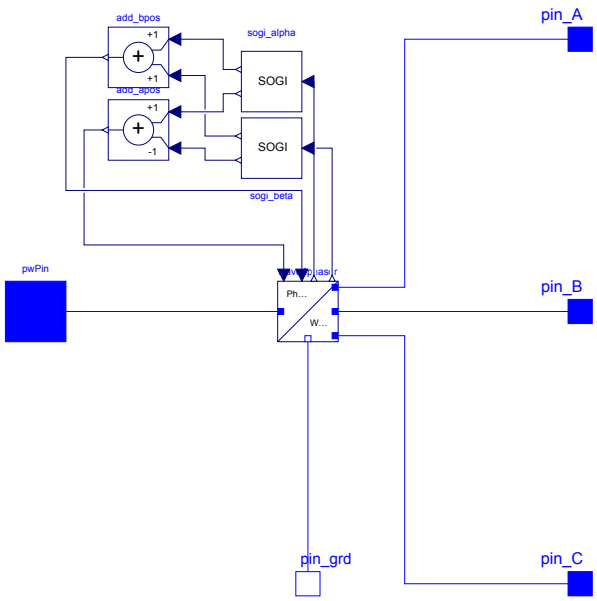


Figure 6. Modelica implementation of the hybrid bus.

the positive sequence (phasor-based model), while the bottom circuit is the hybrid TS-EMT model.

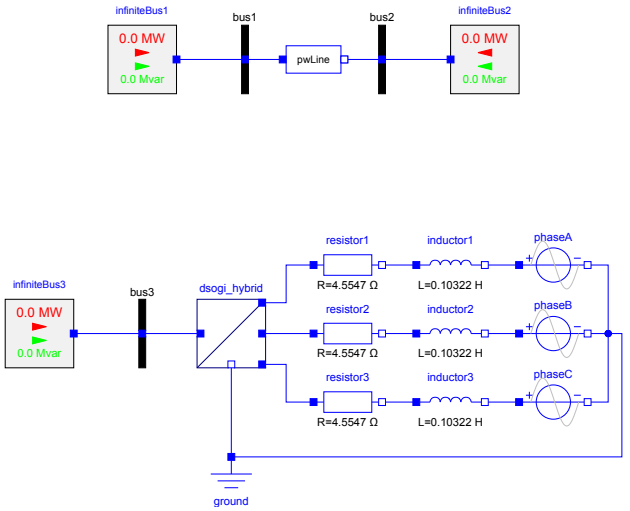


Figure 7. Simple circuit example in Modelica.

The goal of this example is to demonstrate the filtering of the three-phase current wave, showing that both circuits will yield the same result. The example is set to start with zero current on all inductors, which challenges the filter to reach the steady-state response and allows one to observe its behavior. The results presented in Figure 8 show that even under that condition, the filter is able to track the correct real and imaginary values of the current phasor.

5.2 Converter Circuit

The second example describes an application of the hybrid interface, in which a power-electronics-based device

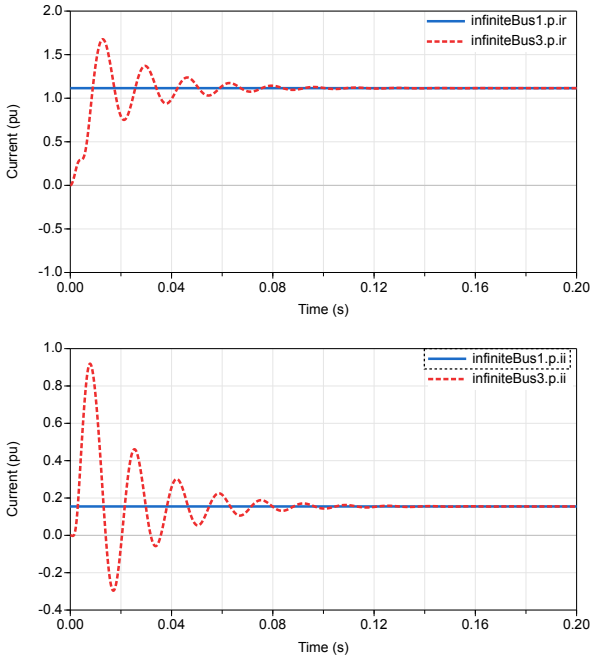


Figure 8. Current filtering simulation result for real (upper plot) and imaginary (bottom plot) components of the positive sequence current phasor.

is represented in detail using its EMT model, while the rest of the grid is built using the phasor-domain (TS) models. The Static Compensator (STATCOM) is an inverter-based resource that can be used to provide reactive power support to the power grid (Hingorani and Gyugyi 1999). The example circuit is presented in Figure 9 showing the interconnection of STATCOM to bus `B2`.

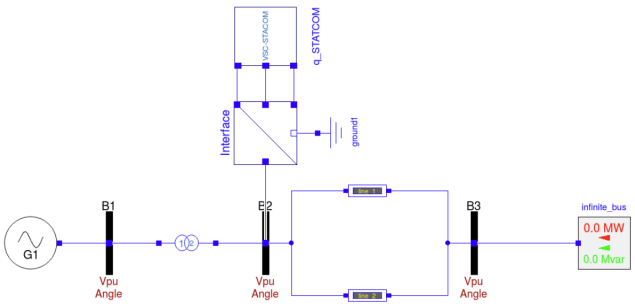


Figure 9. Diagram for converter circuit example.

This test system was developed to show the operation of a STATCOM and its controllers modeled in detail using the EMT approach, while the rest of the system is modeled in phasor-domain for a TS study. The STATCOM controllers used in this example are explained in more detail in (de Castro and Vanfretti 2022). To assess this behavior, the reference to the STATCOM's reactive power injection changes from 0 to 90 Mvar at $t = 1$ second, as shown in Figure 10. The three-phase modulation indices for each phase are shown in Figure 11. Note that the modulation indices change almost instantaneously at $t = 1.0$

sec., showing the appropriate control action for the desired reactive power injection.

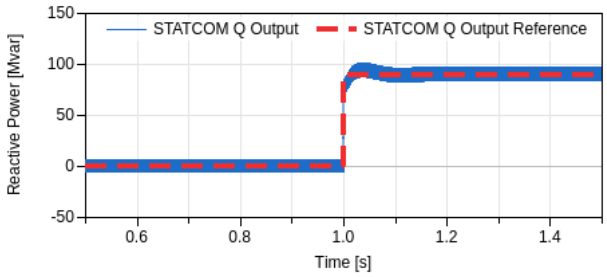


Figure 10. Reference value and reactive power injection by STATCOM.

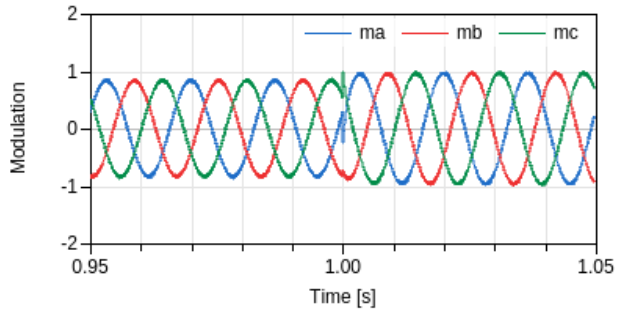


Figure 11. Three-phase modulation indices for the STATCOM's controller.

The voltage behavior on the STATCOM's DC bus is presented in Figure 12. The DC voltage is controlled throughout the simulation at its reference value of 60 kV. Note how the voltage suffers a disturbance exactly at $t = 1$ sec. Another disturbance in the voltage occurs at $t = 0.05$ sec., which is the time instant when the STATCOM connects to the grid. The bus `B2` voltage is depicted in Figure 13. Note that the bus voltage increases its magnitude as the reactive support offered by the STATCOM increases.

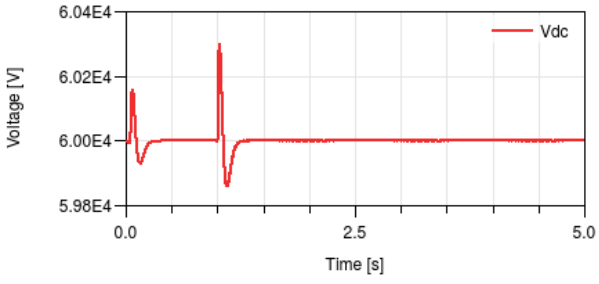


Figure 12. STATCOM DC bus voltage during simulation.

6 Conclusion

This work presented the Open-Instance Wave-Phasor Interface, or OpenIWPI, a Modelica library designed to allow coalesced simulation of hybrid EMT-TS models, using a wave-phasor interface. The interface is built using SOGI-based filters originally proposed in (Theodoro, Tomim, Barbosa, Lima, and Barros 2019). Although the library is small, it provides all the essential components to successfully link EMT and TS models, providing auxiliary models and reference frame transformations that are used by the interface. The interface bus presented in this paper uses connectors from the OpenIPSL, which has power system models for phasor-domain simulations, and it also uses connectors from the Electrical package within the MSL, which allows part of the power system to be modeled for EMT.

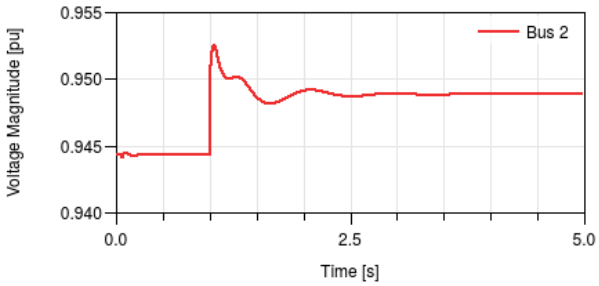


Figure 13. Bus `B2` voltage level during simulation.

Furthermore, this paper describes two example models that are part of the OpenIWPI library and presents their simulation results. The results should provide the user with details on how the library is expected to work and how to assemble larger models using the interface. The wave-phasor interface could be used as a magnifying glass for parts of the system to be modeled in detail (EMT), while other parts of the power grid can continue to have their phasor-based positive-sequence equivalent model (TS). This can be particularly useful when there is not enough information to model the whole power grid in EMT, or when one is interested in a particular plant's response when faced with system-wide dynamics (Boersma et al. 2021) that can only be captured with large-scale simulation models that are typically only represented for TS.

The OpenIWPI library has been tested with Dymola 2025x and requires OpenIPSL release v.3.1.0, which can be found online at: `http://openipsl.org/`. The OpenIWPI library is released as open source, and is available here: `https://github.com/ALSETLab/OpenIWPI`.

References

Adapa, Rambabu and John Reeve (1988). "A new approach to dynamic analysis of AC networks incorporating detailed modeling of DC systems. II. Application to interaction of DC and weak AC systems". In: *IEEE Trans. Power Del.* 3.4, pp. 2012–2019.

Baur, Marcus, Martin Otter, and Bernhard Thiele (2009-09). "Modelica Libraries for Linear Control Systems". In: *7th International Modelica Conference*. 43. Como, Italy:

Linköping Electronic Conference Proceedings, pp. 593–602. DOI: 10.3384/ecp09430068. URL: https://ep.liu.se/en/conference-article.aspx?series=ecp&issue=43&Article_No=68.

Boersma, Sjoerd et al. (2021). "Probing signal design for enhanced damping estimation in power networks". In: *International Journal of Electrical Power & Energy Systems* 129, p. 106640. ISSN: 0142-0615. DOI: 10.1016/j.ijepes.2020.106640. URL: https://www.sciencedirect.com/science/article/pii/S0142061520341855.

Chang, Hao and Luigi Vanfretti (2024). "Power Hardware-in-the-Loop Smart Inverter Testing with Distributed Energy Resource Management Systems". In: *Electronics* 13.10. ISSN: 2079-9292. DOI: 10.3390/electronics13101866.

Chow, Joe H and Juan J Sanchez-Gasca (2020). *Power system modeling, computation, and control*. John Wiley & Sons.

de Castro, Marcelo and Luigi Vanfretti (2022). "Multi Time-Scale Modeling of a STATCOM and Power Grid for Stability Studies using Modelica". In: *2022 Open Source Model. Simul. of Energy Syst. (OSMSES)*, pp. 1–7.

de Castro, Marcelo, Dietmar Winkler, et al. (2023). "Version [OpenIPSL 2.0.0] - [iTesla Power Systems Library (iPSL): A Modelica library for phasor time-domain simulations]"". In: *SoftwareX* 21, pp. 1–3.

Dennetiere, Sebastien et al. (2019). "Supporting Energy Transition in Transmission Systems: An Operator's Experience Using Electromagnetic Transient Simulation". In: *IEEE Power and Energy Magazine* 17.3, pp. 48–60. DOI: 10.1109/MPE.2019.2897179.

Dinavahi, V. et al. (2009). "Interfacing techniques for simulation tools". In: *2009 IEEE Power & Energy Society General Meeting*, pp. 1–6. DOI: 10.1109/PES.2009.5275553.

Dommel, Hermann W (1986). *Electromagnetic Transients Program Reference Manual: EMTP Theory Book*. Portland, OR, USA: Bonneville Power Administration.

Fachini, Fernando et al. (2025-05). "Customized open source renewable energy models validated through PHIL lab experiments". In: *Renewable Energy* 244, p. 122627. ISSN: 0960-1481. DOI: 10.1016/j.renene.2025.122627. (Visited on 2025-04-09).

Gibert, Pierre-Marie et al. (2018). "Use of the sinusoidal predictor method within a fully separated modeler/solver framework for fast and flexible EMT simulations". In: *SIMULTECH*, pp. 34–43.

Heffernan, MD et al. (1981). "Computation of ac-dc system disturbances-Part I. Interactive coordination of generator and convertor transient models". In: *IEEE Trans. Power App. Syst.* PAS-100.11, pp. 4341–4348.

Hingorani, Narain G and Laszlo Gyugyi (1999). *Understanding FACTS: Concepts and Technology of Flexible AC Transmission Systems*. Piscataway, NJ, USA: IEEE Press.

Kaura, Vikram and Vladimir Blasco (1997). "Operation of a phase locked loop system under distorted utility conditions". In: *IEEE Trans. Ind. Appl.* 33.1, pp. 58–63.

Kundur, Prabha S, Neal J Balu, and Mark G Lauby (1994). *Power System Dynamics and Stability*. 1st ed. New York, NY, USA: McGraw Hill Inc.

Mahseredjian, Jean, Venkata Dinavahi, and Juan A. Martinez (2009-07). "Simulation Tools for Electromagnetic Transients in Power Systems: Overview and Challenges". In: *IEEE Transactions on Power Delivery* 24.3, pp. 1657–1669. ISSN: 1937-4208. DOI: 10.1109/TPWRD.2008.2008480. URL: https://ieeexplore.ieee.org/document/4907257 (visited on 2025-05-01).

Martins, Nelson et al. (2000). "A small-signal stability program incorporating advanced graphical user interface". In: *Proceedings of the VII SEPOPE*.

Masoom, Alireza, Jean Mahseredjian, et al. (2021). "MSEMT: An advanced Modelica library for power system electromagnetic transient studies". In: *IEEE Trans. Power Del.* 37.4, pp. 2453–2463.

Masoom, Alireza, Tarek Ould-Bachir, et al. (2020). "Simulation of electromagnetic transients with Modelica, accuracy and performance assessment for transmission line models". In: *Electr. Power Syst. Res.* 189. Art. no. 106799.

Modelica Association (2025). *Modelica language*. https://modelica.org/language/ (accessed Apr. 28, 2025).

Nikolaev, Nikolay et al. (2020). "PSS/E Based Power System Stabilizer Tuning Tool". In: *2020 21st International Symposium on Electrical Apparatus & Technologies (SIELA)*. IEEE, pp. 1–6.

Petzold, Linda R (1982). *Description of DASSL: a differential/algebraic system solver*. Tech. rep. Sandia Nat. Labs., Livermore, CA, USA.

Ramirez-Gonzalez, M. et al. (2024). "Real-Time Simulation of the Initial Dynamic Model of CE in ePHASORsim: Issues and Troubleshooting". In: *2024 IEEE 65th International Scientific Conference on Power and Electrical Engineering of Riga Technical University (RTUCON)*, pp. 1–6. DOI: 10.1109/RTUCON62997.2024.10830830.

Rodriguez, Pedro et al. (2006). "New positive-sequence voltage detector for grid synchronization of power converters under faulty grid conditions". In: *2006 37th IEEE Power Electron. Specialists Conf.* Pp. 1–7.

Schweiger, Gerald et al. (2019). "An empirical survey on co-simulation: Promising standards, challenges and research needs". In: *Simulation modelling practice and theory* 95, pp. 148–163.

Thapa, Jitendra et al. (2023). "Plug-and-Play Regional Models for Real-Time Electromagnetic Transient Simulations of Large-Scale Power Grids: A Case Study of New York State Power Grid". In: *IEEE Access* 11, pp. 87600–87614. DOI: 10.1109/ACCESS.2023.3305394.

Theodoro, Thainan, Marcelo Tomim, Pedro Barbosa, Antonio Lima, and Maria de Barros (2019). "A flexible co-simulation framework for penetration studies of power electronics based renewable sources: A new algorithm for phasor extraction". In: *Int. J. Elect. Power Energy Syst.* 113, pp. 419–435.

Theodoro, Thainan, Marcelo Tomim, Pedro Barbosa, Antonio Lima, and Juan Santiago (2018). "A hybrid simulation tool for distributed generation integration studies". In: *2018 Power Syst. Comput. Conf. (PSCC)*, pp. 1–8.

Theodoro, Thainan, Marcelo Tomim, Antonio de Lima, et al. (2017). "A hybrid simulation tool for penetration studies of distributed generation in smartgrids". In: *2017 Brazilian Power Electron. Conf. (COBEP)*, pp. 1–7.

Watson, Neville and Jos Arrillaga (2003). *Power Systems Electromagnetic Transients Simulation*. Vol. 39. London, UK: IET.

Winkler, Dietmar (2017). "Electrical power system modelling in modelica–comparing open-source library options". In: *Proc. 58th Conf. Simul. Model. (SIMS 58)*, pp. 263–270.

Yazdani, Amirnaser and Reza Iravani (2010). *Voltage-Sourced Converters in Power Systems: Modeling, Control, and Applications*. Hoboken, NJ, USA: John Wiley & Sons.

DOI
10.3384/ecp218223

Expanding an Open-Source Modelica-Compliant Package of Generic Renewable Energy Source Models: Implementation of the REEC_D and REGC_B WECC Models in OpenIPSL

Srijita Bhattacharjee[1] Fernando Fachini[2] Luigi Vanfretti[1]

[1]Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, NY, USA
{bhatts10, vanfrl}@rpi.edu
[2]Electric Transmission Strategic Initiatives, Dominion Energy, VA, USA
fernando.fachini@dominionenergy.com

Abstract

Power systems undergoing large-scale renewable deployment require accurate dynamic models of inverter-based technologies such as solar photovoltaic plants, wind turbine generators, and battery energy storage systems to perform the require studies that would allow to assess and maintain power grid stability. The second generation of the Western Electricity Coordinating Council (WECC) generic renewable energy system (RES) models provides a general framework for modeling inverter-based resources in power system dynamics and stability studies. In this paper, we expand the existing Modelica-compliant open-source `OpenIPSL.Electrical.Renewables` package by implementing the Renewable Energy Generator/Converter B (`REGC_B`) model along with the Renewable Energy Electrical Controller (`REEC_D`) model, according to the second-generation WECC RES framework. Using Modelica's object-oriented features, the implementation emphasizes modularity and reusability in building scalable power system models using the OpenIPSL library. This work highlights the potential of using Modelica and OpenIPSL to support a standardized, scalable development of inverter-based RES models within the WECC framework, and extends the only fully Modelica-compliant open-source package that implements these models.

Keywords: Modelica, OpenIPSL, WECC RES models, inverter-based resources, power systems

1 Introduction

1.1 Motivation

The modeling of inverter-based resources (IBRs) is critical to ensuring reliable and realistic power system simulations, particularly with the growth of renewable energy integration. The development and validation of WECC generic RES models have played a key role in enabling dynamic studies of IBRs by different domain-specific platforms. Meanwhile, recent implementation of these models in OpenIPSL (Vanfretti et al. 2016; Baudette et al. 2018; Castro et al. 2023) using the Modelica language has demonstrated the benefits of modular object-oriented modeling for renewable energy systems (Fachini, Vanfretti, et al. 2021). Complementing this, power-hardware-in-the-loop (PHIL) testing has shown the practical relevance of the WECC framework in representing typical inverter responses under real-world operating conditions(Fachini, Chang, et al. 2025) and the value of implementing WECC models in Modelica to facilitate customization and extensibility.

However, as inverter-based systems continue to be increasingly deployed in weak grid environments, characterized by low short-circuit ratios and limited system strength, modeling accuracy becomes more critical. This is of particular importance to capture negative control interactions and potential stability issues (Adib et al. 2018; Wang et al. 2022; Mishra, Vanfretti, Delaree, and Jones 2024; Pudasaini et al. 2025). To support a more accurate representation of the inverter dynamics under such conditions, a voltage source interface model was proposed in (Ramasubramanian et al. 2016). Based on this foundation, the updated `REGC_B` and `REEC_D` models were developed to enhance the accuracy of the WECC modeling framework in dynamic performance and stability analysis of the grid. Naturally, extending our previous implementation of the WECC models in Modelica (Fachini, Vanfretti, et al. 2021) with these advancements is attractive and can provide additional benefits; including providing accurate IBR models in large-scale grid modeling that favor tools that support the Functional Mock-up Interface (FMI) standard and do not have these models in their internal libraries (Ramirez-Gonzalez et al. 2024) or provide simulation performance benefits through FMI-based parallelization (Ouafi et al. 2023).

1.2 Background and Related Works

The transformation of traditional power systems, driven by the global shift toward decarbonization and decentralization, has accelerated the integration of inverter-based resources such as solar photovoltaic (PV) plants, wind turbines (WT) and battery energy storage systems (BESS). Although crucial for enabling cleaner and flexible power systems, these sources exhibit dynamic responses that depart significantly from those of traditional generation tech-

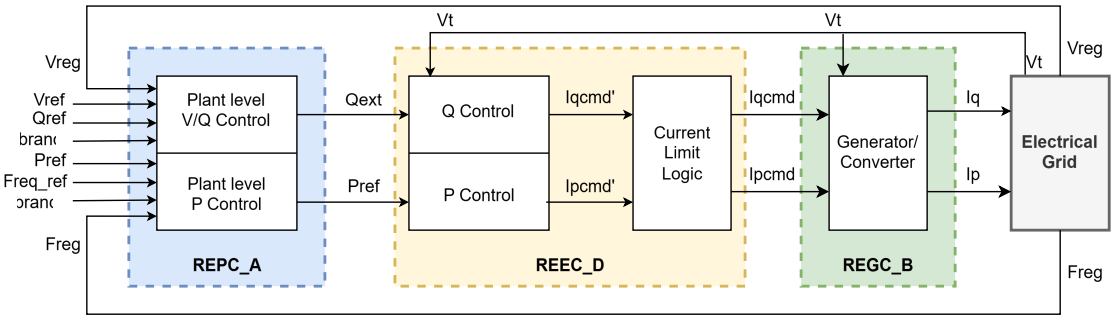


Figure 1. WECC RES Model Framework (Fachini, Vanfretti, et al. 2021)

nologies, posing new challenges to system stability and control. To address these challenges, the Western Electricity Coordinating Council (WECC), through its Renewable Energy Modeling Task Force (REMTF) (Sanchez-Gasca 2015), developed a standardized framework of second generation generic renewable energy system (RES) models for dynamic studies of inverter-based resources. Building on earlier first-generation modeling efforts, this framework aligns with broader international initiatives such as those of the International Electrotechnical Commission (IEC) (Fortmann et al. 2015), to improve modularity, broaden control representation, and support a wider range of technologies, including both wind and photovoltaic systems. Although these modeling efforts offer a consistent framework for representing inverter-based resources, their implementation in Modelica (Fritzson and Engelson 1998; Fritzson 2014) provides the added advantages of open access, modularity, and cross-platform compatibility that are not easily achieved in traditional domain-specific tools (Chang and Vanfretti 2024). When combined with the Open Instance Power System Library (OpenIPSL) (Baudette et al. 2018; Castro et al. 2023), Modelica offers a versatile and open-source language to develop standardized power system models suitable for dynamic simulation and analysis (Winkler 2017; Bhattacharjee, Vanfretti, and Fachini 2024).

The WECC RES models including the renewable energy generator / converter `(REGC_A)`, renewable energy electrical controller `(REEC_A/B/C)`, and renewable energy plant controller `(REPC_A)` were developed and validated against the commercial tool PSS®E in (Fachini, Vanfretti, et al. 2021), setting a viable proof of concept for open-source development of generic inverter-based models in Modelica using OpenIPSL. This effort led to the development of the `OpenIPSL.Electrical.Renewables` (here-forth referred as to the `*.Renewables`) package, which provides the only Modelica language-compliant and Modelica-tool–compatible implementation of the WECC Generic Renewable Energy Models — the models have been validated and tested in multiple Modelica-based simulation environments. Our implementation fully leverages templates with replaceable classes and other object-oriented features of the Modelica language, while also adopting the latest version of the Modelica Standard Library (MSL). Furthermore, the models in `*.Renewables` are compatible with multiple simulation tools such as Dymola, Wolfram System Modeler, Modelon Impact and OpenModelica, and can the models can be translated and simulated directly within these platforms. This is in contrast to other Modelica-based implementations that rely on MSL 3.2.3 and can only be viewed but not simulated in Modelica tools, requiring a domain-specific tool chain for simulation (Nuschke et al. 2021; Franke, Guironnet, and Cardozo 2024). In addition to these advantages, our present work expands the capabilities of the `*Renewables` package by incorporating the latest variants of WECC components, establishing it as the most up-to-date open-source collection of generic models of renewable energy sources available.

To support this expansion, updated versions of the WECC RES models have been introduced to better represent inverter-based resources under challenging grid conditions. The `REGC_B` model improves on `REGC_A` by introducing a voltage-source interface and enhanced current-limiting logic, offering better performance and numerical stability in weak grid scenarios. Likewise, the `REEC_D` improves upon the `REEC_A/B/C` with refined voltage-dependent limits, improved blocking, and more flexible control freeze logic (Pourbeik 2018).

This paper builds on our previous implementations of the WECC generic RES models in (Fachini, Vanfretti, et al. 2021) by extending the framework to include the `REGC_B` and `REEC_D` using the Modelica language and the OpenIPSL library. It presents a modular and reusable approach that uses Modelica's object-oriented feature to build standardized dynamic models of inverter-based resources. This work aims to extend the existing Modelica-based WECC framework and highlights how OpenIPSL can be used to support model development by leveraging open access standardization (i.e., the Modelica language), re-usability, and extensibility for power system studies.

1.3 Contributions

This paper focuses on demonstrating how key features of the Modelica language and the OpenIPSL library can be used to implement and extend standardized WECC RES models for power system studies. Thus, the contributions of this paper are as follows:

DOI
10.3384/ecp218233

- To implement next generation WECC RES models using the OpenIPSL Modelica Library.
- To exploit the replaceable template in Modelica to build the integrated models.
- To demonstrate the simulation setup and verify the features implemented in the new models.

1.4 Paper Structure

This paper is structured as follows. Section 2 describes the development of the new WECC generic models. In Section 3, we explain how Modelica is leveraged to integrate the new models within the WECC framework implemented in OpenIPSL. Section 4 illustrates the experimental setup and the simulation result. Finally, Section 5 concludes the work.

2 Development of the new WECC Generic Renewable Energy Models

Accurate representation of renewable energy sources based on inverters in dynamic power system studies requires standardized and adaptable modeling frameworks. The WECC generic RES models fulfill this need through a modular architecture that separates system-level coordination, electrical control, and grid interfacing into distinct components, as shown in Figure 1. At the plant level, the `REPC_A` module manages voltage and frequency regulation by generating real and reactive power references in response to system conditions. These references are passed to the electrical control module, historically modeled by `REEC_A/B/C`, which translates them into active and reactive current commands. Finally, the `REGC_A` generator/converter model interfaces with the grid, injecting current based on these commands. Although effective under normal operating conditions, this model structure lacks certain capabilities required to accurately simulate inverter behavior during certain types of faults, grid disturbances, or in weak-grid environments. To address these limitations in simulating realistic inverter behavior under faults and weak-grid conditions, the WECC framework has introduced more advanced controller and converter models —`REEC_D` and `REGC_B`. Figure 1 illustrates the updated RES modeling framework, which retains the original modular structure while replacing `REEC_A/B/C` and `REGC_A` with their improved counterparts.

This section discusses the development of the new models - `REEC_D` and `REGC_B` implemented in the Modelica language and integrated into the OpenIPSL library, following WECC specifications and ensuring compatibility with existing plant-level controllers such as `REPC_A`.

2.1 `REGC_B` Generator/Converter Model

The `REGC_B` model serves as the generator/converter component within the second-generation WECC generic RES framework. It replaces the earlier `REGC_A` model and introduces functional enhancements that improve the accuracy and numerical stability of inverter-based resource

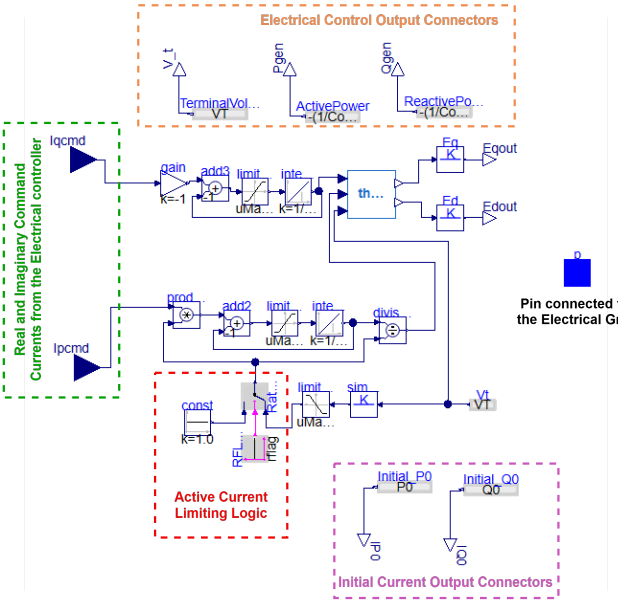


Figure 2. REGC_B model in Modelica and OpenIPSL

(IBR) simulations, particularly under conditions involving grid disturbances and low short-circuit strength.

The original `REGC_A` model is based on a current-source representation, wherein the model directly injects active (I_p) and reactive current (I_q) into the network. While effective under strong grid conditions, it has two key limitations. These include ignoring the voltage and impedance at the connection point; which leads to poor performance in weak grids with low short-circuit ratios (SCR), often causing numerical instability in domain-specific tools that are limited to fixed time-step solvers such as the trapezoidal integration method. According to the WECC REMTF, `REGC_A` becomes unreliable when the SCR drops below 2–3, frequently requiring artificial numerical stabilization measures.

To address these limitations, `REGC_B`, as shown in Figure 2, models the converter as a voltage source. The transition from `REGC_A` to `REGC_B` introduces several key enhancements aimed at improving numerical stability, dynamic performance, and modeling flexibility under weak grid conditions. The essential ones are as follows:

- **Voltage-Source Interface:** `REGC_B` replaces the current-source representation of `REGC_A` with a voltage source interface, improving behavior in low short-circuit ratio (SCR) scenarios. The injected current is computed using:

$$\vec{I}_{dq} = \frac{\vec{E}_{dq} - \vec{V}_t}{R_e + jX_e}$$

- **Active Current/Power Rate Limiting:** REGC_B introduces a `RateFlag` to select between active current or power ramping, using a rate limit parameter

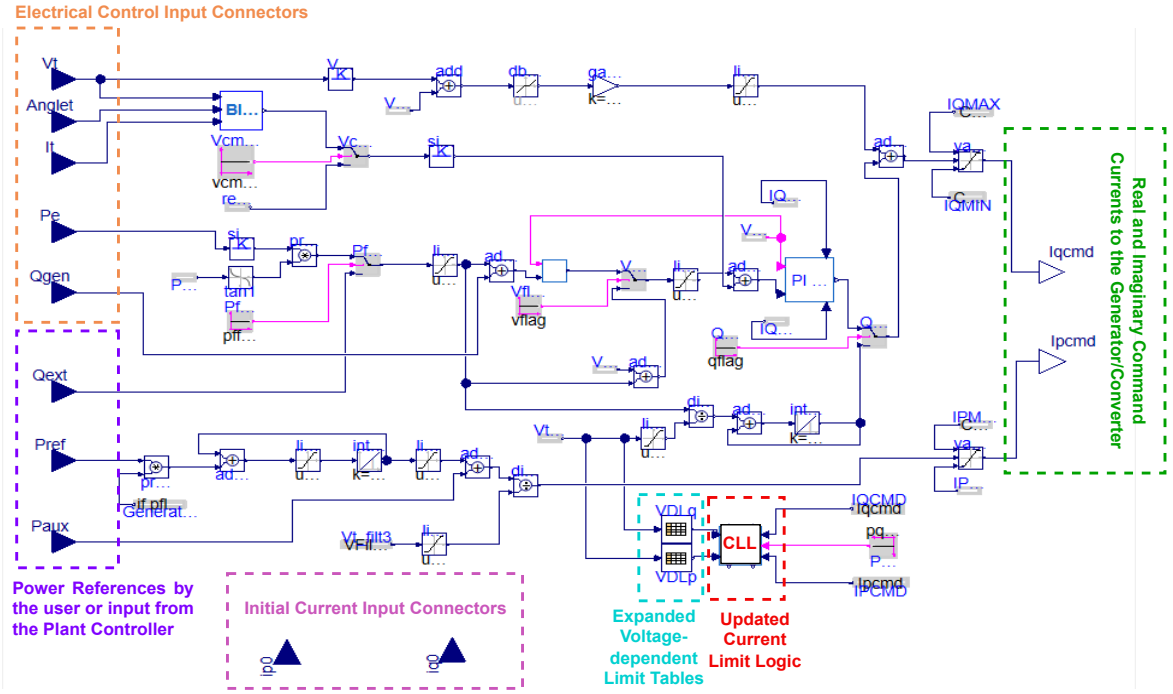


Figure 3. REEC_D model in Modelica and OpenIPSL

`rrpwr`. For current ramping:

$$\frac{dI_p}{dt} = \begin{cases} +rrpwr & \text{if } \frac{dI_p}{dt} > rrpwr \\ -rrpwr & \text{if } \frac{dI_p}{dt} < -rrpwr \\ \frac{dI_p}{dt} & \text{otherwise} \end{cases}$$

- **Removal of LVPL Block:** The LVPL block present in `REGC_A` is removed, as its behavior is now handled more effectively in upstream control logic (e.g., `REEC_D`).
- **Reactive Current Rate Limiting:** Ramp rate limits are also applied to the reactive current path, using bounds defined by `Iqrmax` and `Iqrmin`.
- **Current Magnitude Limiting:** The total current output is constrained by a maximum limit `Imax`. Priority between active and reactive current is set using `pqflag`, with:

$$I^2 = I_p^2 + I_q^2 \leq I_{\max}^2$$

when this limit is exceeded, the model applies prioritization logic (via the `pqflag`) to limit either I_p or I_q, depending on the configuration. This algebraic approach avoids the need for dynamic differential limiters and improves the simulation performance.

Together, the voltage-source interface and current-limiting logic enable `REGC_B` to operate reliably in low SCR environments (SCR $\approx$ 1), respond more accurately to fault conditions, and avoid numerical convergence issues common to current-source models. These enhancements make `REGC_B` a more robust and practical choice for dynamic simulation of inverter-based resources.

2.2 `REEC_D` Electrical Controller Model

The `REEC_D` model builds on the structure of the earlier electrical controller model `REEC_A`, introducing a more flexible framework for handling voltage-dependent behavior. `REEC_D` integrates a flexible set of control mechanisms, including an improved current limiting logic that allows a more realistic representation of inverter behavior under abnormal or low-voltage grid conditions. Figure 3 illustrates the entire model in Modelica using OpenIPSL. It shows key input signals such as *Pref*, *Qext*, *Pe*, and *Qgen* and the resulting current commands *Ipcmd* and *Iqcmd* that are fed to the generator/converter model. Internal blocks such as the updated current limit logic and expanded voltage-dependent limit tables are highlighted to show their roles. These advancements are discussed below.

- **Expanded Voltage-Dependent Limit (VDL) Tables:** `REEC_D` defines both reactive and active current limits using 10-point VDL tables:

$$\texttt{VDLq} = \{(V_{q,i}, I_{q,i})\}, \quad i = 1, \ldots, 10$$

$$\texttt{VDLp} = \{(V_{p,i}, I_{p,i})\}, \quad i = 1, \ldots, 10$$

Expanding the tables allow detailed shaping of the inverter's current response to different voltage levels.

DOI 10.3384/ecp218233

- **Enhanced Current Limit Logic:** The new limit equations are defined as:

$$I_{q\max} = \min(\mathrm{VDL}_q, I_{\max})$$

$$I_{q\min} = \begin{cases} I_{q\max}, & I_{q\max} < 0 \\ -I_{q\max}, & \text{otherwise} \end{cases}$$

$$I_{p\max} = \min(\mathrm{VDL}_p, \sqrt{I_{\max}^2 - I_{q\mathrm{cmd}}^2})$$

$$I_{p\min} = -K_e \cdot I_{p\max}$$

To implement these limit equations, a new logic as shown in Listing 1 was developed. It enables the model to distinguish between the generator and storage behavior by adjusting the lower active current limit based on the parameter K_e, allowing absorption of active power when $K_e > 0$. Furthermore, it ensures consistent handling of reactive power limits during high-voltage events by forcing $I_{q\min}$ to match a negative $I_{q\max}$, improving the model response under extreme conditions.

Listing 1. Enhanced Current Limit Logic in the `REEC_D` Model

```
model CLL_REECD "Current limit logic
    for REECD"
...
equation

Iqmax = if pqflag == false then min(
    VDLq_out, Imax) else min(VDLq_out,
    sqrt(Imax^2 - Ipcmd^2));
Iqmin = if Iqmax < 0 then Iqmax else -
    Iqmax;
Ipmax = if pqflag == false then min(
    VDLp_out, sqrt(Imax^2 - Iqcmd^2))
    else min(VDLp_out, Imax);
Ipmin = -Ke*Ipmax;
```

- **Local Current Compensation Block :** In this new electrical controller model there is a dedicated block to compensate for current measurement delays using the expression

$$|V_t - (R_c + jX_c)I_t|\,.$$

This helps stabilize the inverter response by correcting for dynamic errors due to measurement lags or system impedance, which is especially important for grid-tied converters.

- **The Baseload Flag :** The baseload flag feature allows control over how the power output of the inverter behaves. With the flag set to 0, the model operates normally. When set to 1, the output is capped at the initial power level P_{gen0}, allowing only reductions. If set to 2, the output is fixed to P_{gen0}.

3 Leveraging Modelica's Object Oriented Features for Open-Source Generic RES Models

This section discusses the benefits of using Modelica to implement the generic RES models in OpenIPSL, particularly through its support for the development of model architectures that provide modularity and object-oriented design. One of the key features leveraged in this work is the use of `replaceable` classes, which enable the substitution of flexible components and system reconfiguration. Figure 4 shows the structure of the `OpenIPSL.Electrical.Renewables` package, where control components such as the plant controller, electrical controller, and generator/converter are organized into distinct subpackages, supporting a clean and extensible modeling architecture for renewable plant systems.

3.1 Replaceable Template-Based Modeling of the WECC Modules

A key advantage of using Modelica in this work is the use of `replaceable` classes. This feature enhances flexibility and modularity in system design. Figure 4 illustrates the PV plant, enclosed in the red dotted box, where the modules `REPC_A`, `REEC_D`, and `REGC_B` are visually represented as gray highlighted blocks, indicating their implementation as `replaceable` classes in Modelica. The `RenewableElectricalController` block is highlighted in a green box to emphasize this concept, although the replaceable structure applies equally to the other modules. This approach is further illustrated in the Listing 2, where the `RenewableElectricalController` is declared as `replaceable` in the text layer, allowing flexible reconfiguration of renewable plant components in accordance with WECC modeling guidelines. This flexibility is especially valuable when evaluating multiple converter configurations or when integrating new control strategies. It allows for rapid model customization, making it well suited for system-level studies where converter behavior may vary based on application or test conditions. The modeling approach adopted here builds on principles demonstrated in our previous work on generation units using OpenIPSL and Modelica's object-oriented architecture (Fachini, Bhattacharjee, et al. 2023). While that work focused on synchronous generation units, the current model extends the concept to detailed representation of power converter units, further validating the usefulness of `replaceable` templates for scalable and reusable component modeling.

Listing 2. `Replaceable` Declaration of the `RenewableElectricalController REEC_D` in the `PV Plant` Model

```
model PV "Framework for a photovoltaic
    plant including controllers"
....
replaceable OpenIPSL.Electrical.Renewables.
    PSSE.RenewableElectricalController.
    BaseClasses.BaseREECD
```

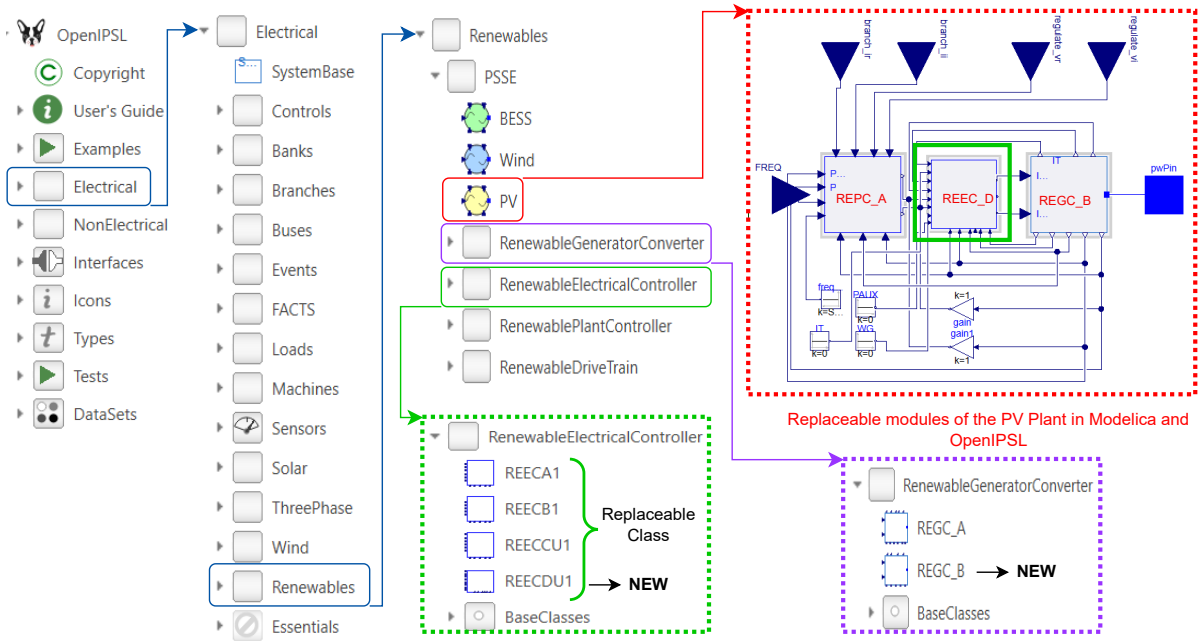


Figure 4. The `OpenIPSL.Electrical.Renewables` Package Structure

```
    RenewableController(
    pfflag=pfflag,
    vflag=vflag,
....
end PV
```

3.2 Utilizing the Switch Component in Modelica for Current Limit Logic

The implementation of the `RateFlag` in the `REGC_B` model marks a key improvement over the `REGC_A` by introducing logic that allows the rate limit to apply either to active-current (*Ip*) or to active-power (*P*), depending on the value of the flag. When `RateFlag = 0`, the logic enforces a rate limit on the current ramp (*Ip*), and when `RateFlag = 1`, it enforces it on active power. This distinction reflects practical considerations, as some vendors implement rate limits in terms of power rather than current. The flexibility of Modelica enables this logic to be cleanly embedded using graphical components such as the `Logical.Switch` block, which conditionally routes the signal based on the Boolean value of the `RateFlag`. As shown in Figure 2, the section marked with red dotted lines labeled "Active Current Limiting Logic" implements the `RateFlag` functionality. The use of a `Switch` block in Modelica enables seamless routing between control paths based on the flag value, without repeating the code in the text layer.

4 Results

4.1 Simulation Model Configuration

The components built in Sections 2 and 3 are arranged and connected as shown in Figure 4, forming the internal

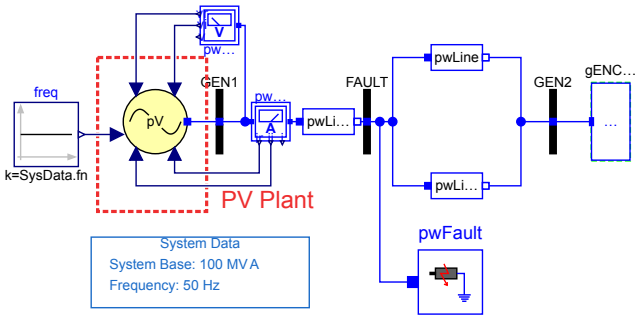


Figure 5. PV Plant in an SMIB Test Setup in OpenIPSL

structure of the PV plant. Then this PV plant model as highlighted within the red dotted block in Figure 5 is tested by integrating it into a single machine infinite bus (SMIB) test system. A similar model can be set for other models, such as BESS or Wind Plant, simply by making changes to existing templates in OpenIPSL, `*.Renewables.BESS` or `*.Renewables.Wind`, and setting control flags according to the WECC guidelines.

The setup in Figure 4 introduces a three-phase fault from ground to ground through the `pwFault` component of OpenIPSL allowing for evaluation of their dynamic response under grid disturbances. The system includes three transmission lines (`pwLine`, `pwLine1`, and `pwLine2`) with defined impedances. The bus `GEN2` is connected to an infinite bus model from OpenIPSL that represents the connection to the grid.

4.2 Comparing `REEC_B` vs `REEC_D`

Next, the voltage response and fault dynamics are analyzed to validate the behavior of the model under grid disturbances when using the existing `REEC_B` vs. the newly implemented `REEC_D`. The results are verified through

DOI
10.3384/ecp218233

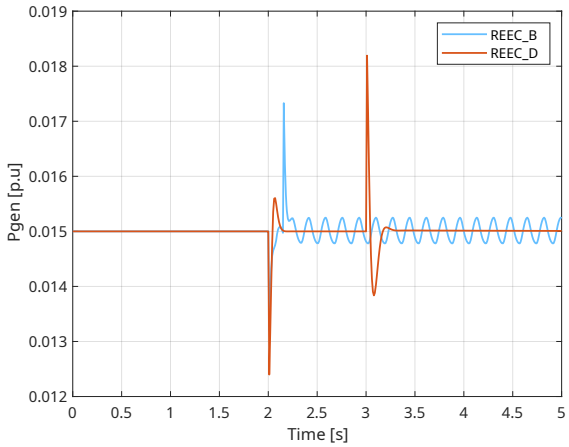


Figure 6. Active Power generated by the PV Plant in OpenIPSL, comparing the response of the new `REEC_D` model vs the previous `REEC_B` model.

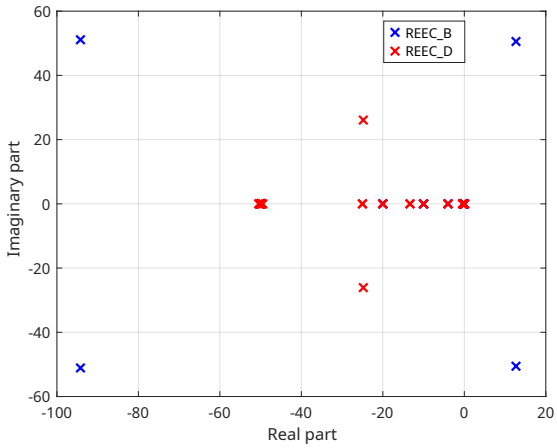


Figure 7. Poles of the PV plant comparing the REEC_B and REEC_D controllers

linear analysis of the two model variants.

Figure 6 shows the active power output P_{gen} of the `REGC_A` converter model along with the newly implemented controller `REEC_D` (in red), along with the response of the `REGC_A` with the previous controller `REEC_B` (in blue). The simulation scenario subjects the power system to a fault at the `FAULT` bus (see Figure 5), which occurred around 2 seconds and lasted for 1 second. Observing the red trace, which corresponds to the use of the new controller model `REEC_D`, during the application of the fault, there is a sharp deviation in P_{gen}followed by a transient response as the system stabilizes. After the event, the power gradually returns to its steady state, indicating the system's ability to recover from the fault. It should be noted that the blue line that corresponds to the response when simulating the use of the `REEC_B` controller results in a growing oscillatory response during the fault, along with a critically stable response after the fault. This shows one of the main benefits for simulation accuracy that the new model offers.

Figure 7 illustrates the linear analysis of the PV plant test model as shown in Figure 4. The poles of the model with both the electrical controllers, `REEC_D` and the previous version `REEC_B`, are compared. It can be observed that the `REEC_B` model, exhibits an unstable pole, which is not present in the newly implemented `REEC_D` model. This is also reflected in the Table 1 where the `REEC_D` model shows poles with lower frequency (5.73 Hz) and moderate damping (0.6889), indicating slower, well-damped oscillations. In contrast, the `REEC_B` model has a pair of poles with a higher frequency (8.3 Hz) and negative damping (-0.2433), which explains the oscillations as seen in Figure 6. Observe that although this is an unstable pole, the states that affect are internal to the plant, that is, the plant controller and the renewable energy controller, as indicated in the contribution to states column in Table 1. Consequently, this results in the oscillatory response in Figure 6, and because in the test model it does not interact with other

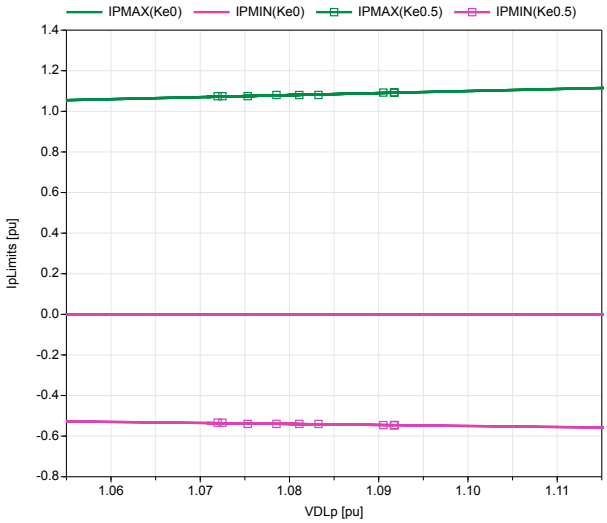


Figure 8. Active Current Limits for $K_e = 0$ and 0.5 based on the expanded VDL_p table.

dynamic components, the power system will not become unstable. This is the effect of modeling the "infinite bus", which absorbs the oscillations injected by the PV and does not interact with it. This bounded response when interfacing a power electronic device with a strong network has recently been observed for the case of PV plants (Wang et al. 2022) and data centers (Mishra, Vanfretti, Delaree, Purcell, et al. 2025), and therefore care should be taken when choosing when to use `REEC_B` or `REEC_D`, and/or, how they are parameterized in order to capture this behavior if appearing in a particular plant.

4.3 Illustrating New Model Features of `REEC_D`

Figure 8 illustrates the verification of the enhanced current limit logic based on the expanded voltage-dependent limit table implemented in the `REEC_D` model, as defined in Listing 1. The plot shows the active current limits (`Ipmax` and `Ipmin`) for two configurations: $K_e = 0$ and $K_e = 0.5$, plotted against the voltage-dependent limit out-

Conjugate Pair number	Eigen Value	Freq. [Hz]	ζ	Contribution to States
Model: REEC_D				
19/20	$-24.795 \pm j26.09$	5.7284	0.6889	z[19/20] → `PlantController.leadLag.TF.x_scaled[1]` (73 %) z[19/20] → `PlantController.simpleLag2.state` (8.7%)
Model: REEC_B				
18/19 & 16/17	$12.68 \pm j50.55$	8.2959	-0.2433	z[18/19] → `PlantController.leadLag.TF.x_scaled[1]` (41%) z[16/17] → `RenewableController.integrator2.y` (25.5%)

Table 1. Complex conjugate pairs of eigenvalues and their contributions for REEC_D and REEC_B models.

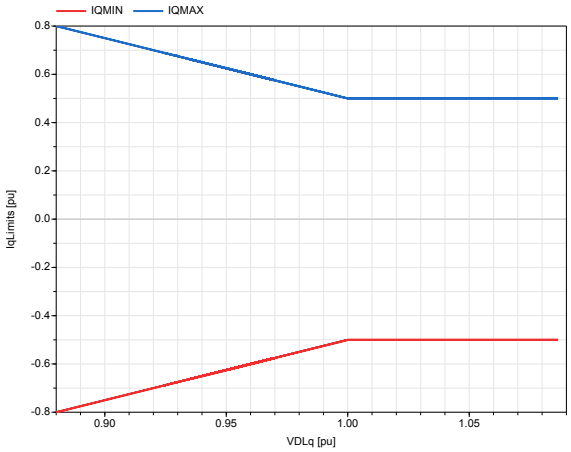


Figure 9. Reactive Current Limits for $K_e = 0$ based on the expanded VDL_q table.

put, VDL_p. The results are consistent with the expected behavior presented in the WECC report introducing the model (Pourbeik 2018), where the maximum current (`Ipmax`) remains unchanged in both cases, reflecting its independence from the K_e parameter.But the minimum active current (`Ipmin`) varies with K_e, remaining at zero for $K_e = 0$, which corresponds to a generator that allows discharging only, with no charging capability. In contrast, for $K_e = 0.5$, `Ipmin` is around -0.5 while `Ipmax` remains around 1.0, reflecting a storage device and indicating that the it can charge at half the maximum discharging rate. This is clearly reflected in the plot, where the `Ipmin` curve for $K_e = 0.5$ reaches half the magnitude of the corresponding `Ipmax`.

Similarly, Figure 9 compares the implemented reactive current limits (`Iqmax` and `Iqmin`) in the model against the expected behavior defined by the VDL_q table. The simulated curves closely follow the voltage-dependent shape of the reference table which confirms that the model correctly applies the VDL_q logic to constrain the reactive current based on terminal voltage. The observation made on both Figures 8 and 9 is verified by the results reported in the WECC documentation (Pourbeik 2018).

5 Conclusions

This work demonstrates the effective use of Modelica and the OpenIPSL library to implement the second-generation WECC generic renewable energy system models, specifically the generator/converter `REGC_B` and the electrical controller `REEC_D` components. Leveraging replaceable template-based modeling in Modelica allowed us to build flexible and modular models that can adapt to different simulation needs without changing the base structure. This approach supports faster testing and better customization, especially when working with different converter setups. The use of the `RateFlag` in the `REGC_B` model added a practical and important improvement by allowing the model to switch between limiting based on current or power. This logic was built using Modelica's `Switch` component, which simplifies the design by clearly separating control paths and makes the model easier to trace and modify through its visual interface. Overall, the combination of Modelica and OpenIPSL offers a strong and reusable way to model inverter-based resources and supports more accurate and scalable studies of power system dynamics.

In future work, we will develop new power grid models that would allow us to better illustrate the behavior of the generator/converter `REGC_B`, with a particular focus on presenting its results, as it is specifically designed to handle weak grid conditions. We will also examine the electrical controller `REEC_D` under different SCRs and compare them against the previous models in order to illustrate the benefits mentioned in the `WECC` reports (Pourbeik 2018).

To access the models in this paper before they are integrated into OpenIPSL, the reader can find them in the following GitHub repository: `https://github.com/ALSETLab/OpenIPSL_IMC2025WECC`

Acknowledgements

This paper is in part, based upon work supported by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) under the Industrial Efficiency and Decarbonization Office, Award Number DE-EE0009139, and in part by the National Science Foundation Award No. 2231677.

References

Adib, Aswad et al. (2018). "On stability of voltage source inverters in weak grids". In: *IEEE Access* 6, pp. 4427–4439.

Baudette, Maxime et al. (2018). "OpenIPSL: Open-instance power system library—update 1.5 to "iTesla power systems library (iPSL): A modelica library for phasor time-domain simulations"". In: *SoftwareX* 7, pp. 34–36.

Bhattacharjee, Srijita, Luigi Vanfretti, and Fernando Fachini (2024). "Building Power System Models for Stability and Control Design Analysis using Modelica and the OpenIPSL". In: *Modelica Conferences*, pp. 63–71.

Castro, Marcelo de et al. (2023). "Version [OpenIPSL 2.0. 0]-[iTesla Power Systems Library (iPSL): A Modelica library for phasor time-domain simulations]". In: *SoftwareX* 21, p. 101277.

Chang, Hao and Luigi Vanfretti (2024). "Integrating the IEEE/-CIGRE DLL Modeling Standard to Use "Real Code" Models for Power System Analysis in Modelica Tools". In: *Modelica Conferences*, pp. 72–79.

Fachini, Fernando, Srijita Bhattacharjee, et al. (2023). "Exploiting Modelica and the OpenIPSL for University Campus Microgrid Model Development". In: *Modelica Conferences*, pp. 285–292.

Fachini, Fernando, Hao Chang, et al. (2025). "Customized open source renewable energy models validated through PHIL lab experiments". In: *Renewable Energy* 244, p. 122627.

Fachini, Fernando, Luigi Vanfretti, et al. (2021). "Modeling and Validation of Renewable Energy Sources in the OpenIPSL Modelica Library". In: *IECON 2021–47th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, pp. 1–6.

Fortmann, Jens et al. (2015). "Wind Plant Models in IEC 61400-27-2 and WECC - latest developments in international standards on wind turbine and wind plant modeling". English. In: 14th International Workshop on Large Scale Integration of Wind Power into Power Systems as well as on Transmission Networks for Offshore Wind Power Plants, Conference date = 20-10-2015 Through 22-10-2015.

Franke, Martin, Adrien Guironnet, and Carmen Cardozo (2024). "Comparing IEC and WECC generic dynamic models for type 4 wind turbines". In: *Electric Power Systems Research* 235, p. 110806.

Fritzson, Peter (2014). *Principles of object-oriented modeling and simulation with Modelica 3.3: a cyber-physical approach*. John Wiley & Sons.

Fritzson, Peter and Vadim Engelson (1998). "Modelica-a unified object-oriented language for system modeling and simulation". In: *ECOOP*. Vol. 98. Citeseer, pp. 67–90.

Mishra, Chetan, Luigi Vanfretti, Jaime Delaree, and Kevin D. Jones (2024). "Analyzing a Non-Sinusoidal Response from a Real-World Solar PV". In: *IEEE Transactions on Power Systems* 39.2, pp. 4771–4774. DOI: 10.1109/TPWRS.2024.3350377.

Mishra, Chetan, Luigi Vanfretti, Jaime Delaree, T.J. Purcell, et al. (2025). "Understanding the inception of 14.7Hz oscillations emerging from a data center". In: *Sustainable Energy, Grids and Networks* 43, p. 101735. ISSN: 2352-4677. DOI: https://doi.org/10.1016/j.segan.2025.101735. URL: https://www.sciencedirect.com/science/article/pii/S2352467725001171.

Nuschke, Maria et al. (2021). "Implementation and Validation of the Generic WECC Photovoltaics and Wind Turbine Generator Models in Modelica". In: *Modelica Conferences*, pp. 633–642.

Ouafi, M. et al. (2023). "Parallelization of EMT simulations for integration of inverter-based resources". In: *Electric Power Systems Research* 223, p. 109641. ISSN: 0378-7796. DOI: https://doi.org/10.1016/j.epsr.2023.109641. URL: https://www.sciencedirect.com/science/article/pii/S0378779623005308.

Pourbeik, Pouyan (2018). *Proposal for New Plant Controller and Electrical Controller*. Tech. rep. Revised multiple times through February 2023. WECC Renewable Energy Modeling Working Group (REMWG). URL: https://www.wecc.org/sites/default/files/documents/progress_report/2025/Memo_RES_Modeling_Updates_For_New_Plant_and_Electrical_Controls_011524.pdf.

Pudasaini, Bikal et al. (2025). "Dynamic Performance Analysis of an Inverter-Based PV Plant during Sunrises and Sunsets through Synchrophasors". In: *2025 IEEE Power & Energy Society General Meeting (PESGM)*. Austin, Texas, USA.

Ramasubramanian, Deepak et al. (2016). "Converter model for representing converter interfaced generation in large scale grid simulations". In: *IEEE Transactions on Power Systems* 32.1, pp. 765–773.

Ramirez-Gonzalez, M. et al. (2024). "Real-Time Simulation of the Initial Dynamic Model of CE in ePHASORsim: Issues and Troubleshooting". In: *2024 IEEE 65th International Scientific Conference on Power and Electrical Engineering of Riga Technical University (RTUCON)*, pp. 1–6. DOI: 10.1109/RTUCON62997.2024.10830830.

Sanchez-Gasca, Juan J (2015). "Generic wind turbine generator models for WECC-a second status report". In: *2015 IEEE Power & Energy Society General Meeting*. IEEE, pp. 1–5.

Vanfretti, Luigi et al. (2016). "iTesla Power Systems Library (iPSL): A Modelica library for phasor time-domain simulations". In: *SoftwareX* 5, pp. 84–88.

Wang, Chen et al. (2022). "Identifying Oscillations Injected by Inverter-Based Solar Energy Sources". In: *2022 IEEE Power Energy Society General Meeting (PESGM)*, pp. 1–5. DOI: 10.1109/PESGM48719.2022.9916830.

Winkler, Dietmar (2017). "Electrical power system modelling in modelica–comparing open-source library options". In: *Proceedings of the 58th Conference on Simulation and Modelling (SIMS 58) Reykjavik, Iceland, September 25th–27th, 2017*. 138. Linköping University Electronic Press, pp. 263–270.

DOI
10.3384/ecp218233

Analytical Treatment of Hollow Toroid Flux Tubes

Herbert Schmidt

Department of Electrical Engineering, Bochum University of Applied Sciences, Germany
herbert.schmidt@hs-bochum.de

Abstract

Stray flux tubes around cylindrical poles are commonly modelled starting from the results for planar flux tubes using the circumference of the cylinder as depth. While this is a tried and tested approach, we here discuss analytical expressions using the actual axisymmetric geometry of a fraction of a hollow torus and compare their results to those of the accepted approach.
Keywords: magnetics, stray flux, reluctance force

1 Introduction

In analytically calculating the permeance G_m (or equivalently the reluctance R_m) of a given stray flux tube, we start from Hopkinson's law for a prismatic element:

$$\frac{1}{R_m} = G_m = \frac{\Phi}{V_m} = \mu_0 \frac{A}{l} \tag{1}$$

where Φ is the flux perpendicular to a surface area A and V_m is the magnetic tension along its length l. This simple equation explicitely requires a prismatic flux tube, i.e., constant cross-section and constant length of flux lines within this element (cf. Fig. 1a).

If one of these prerequisites is not met, the flux tube may be subdivided indefinitely and the effective permeance then is computed as an integral over these elements.[1] This works well, e.g., for a hollow cylinder (or any fraction thereof) with radial or circumferential flux. A typical teaching problem might be a half hollow cylinder with circumferential flux (cf. Fig. 1b), resulting in (Roters 1941, P. 132, Eq. 8b):

$$G_m = \frac{\mu_0 l}{\pi} \ln \frac{r_o}{r_i} = \frac{\mu_0 l}{\pi} \ln \left(1 + \frac{2t}{g}\right) \tag{2}$$

where r_o is the outer radius, $r_i = g/2$ is the inner radius, $t = r_o - r_i$ their difference, and l is the length (depth) of the hollow cylinder. Here, the cross section $A = tl$ is constant, but the length of the flux lines varies from πr_i to πr_o.

The same method that yields this analytical result becomes slightly more cumbersome, if both the cross section and the length vary within the flux tube. It still is valid, though, and in this paper we will show the analytical result for the flux in similar flux tubes wrapped within or without cylindrical poles, i.e., for half or quarter hollow torus geometries, where both the cross section per flux line and the length of each flux line are variable. We will cover five qualitatively different situations (cf. Fig. 2):

(a) inner half hollow torus

(b) lower (or upper) half hollow torus

(c) outer half hollow torus

(d) inner quarter hollow torus

(e) outer quarter hollow torus

[1] Note, that this is allowable only for stray flux tubes, i.e., $\mu_r = 1$ (or at least constant), since variable cross section means variable operating point and hence variable permeability within the element.

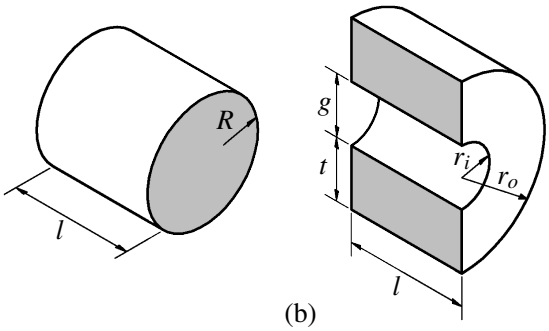


Figure 1. Flux tube geometries. (a) generic prismatic flux tube with $A = \pi R^2$. (b) Half hollow cylinder with circumferential flux. In both figures, flux enters and exits the flux tube through areas marked in gray.

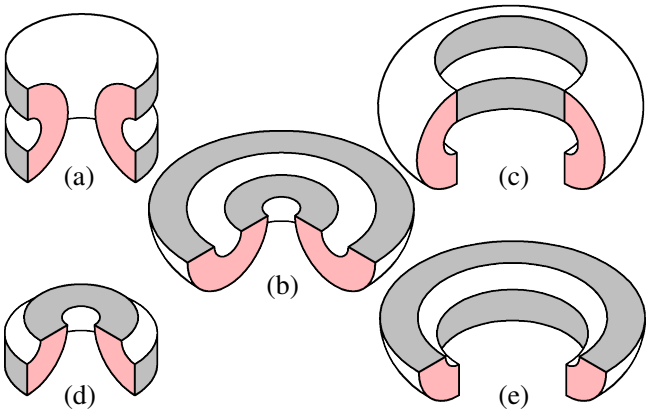


Figure 2. Flux tube geometries. (a) inner half hollow torus. (b) lower half hollow torus. (c) outer half hollow torus. (d) inner quarter hollow torus. (e) outer quarter hollow torus. In all figures, flux enters and exits the flux tube through areas marked in gray. Red are areas opened by 3/4 cut-aways. The actual flux tubes extend over a full rotation.

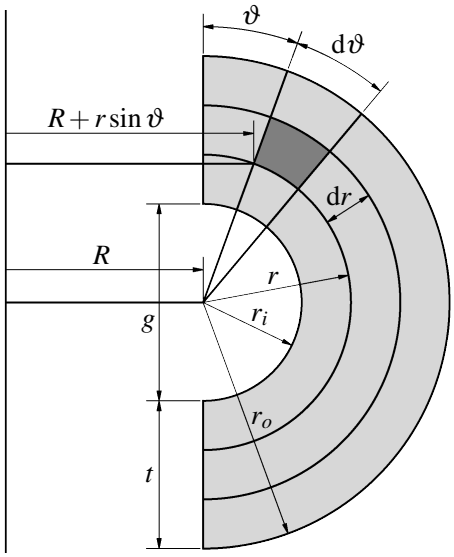


Figure 3. Geometry used in integration.

All these may be addressed by first focusing on the outer and inner half hollow torus. The quarter variants have half the reluctance of the half variants, while the lower (or upper) half hollow torus is the sum of an inner and an outer quarter hollow torus.

Mind that these flux tubes look like half hollow cylinders in their cross section, but they are none, because they are wrapped around cylindrical poles. It is therefore not permissible to simply scale the result to arbitrary polar angles. Scaling from half to quarter hollow torus is the only acceptable fraction, since the flux tube is perfectly mirror symmetrical with respect to this cutting plane.

2 Analytical Treatment

We use a subdivision as depicted in Figure 3 and start by calculating the permeance of an infinitesimal flux tube at radius r and polar angle ϑ:

$$G_m(r,\vartheta) = \mu_0 \frac{2\pi(R \pm r\sin\vartheta)\,\mathrm{d}r}{r\,\mathrm{d}\vartheta} \tag{3}$$

The positive sign refers to the outer half hollow torus geometry shown in Figure 3, the negative sign refers to the corresponding inner half hollow torus.

We first derive the total permeance of a slice of polar width $\mathrm{d}\vartheta$:

$$G_m(\vartheta)\,\mathrm{d}\vartheta = \int_{r_i}^{r_o} 2\pi\mu_0 \left(\frac{R}{r} \pm \sin\vartheta\right) \mathrm{d}r \tag{4}$$

$$= 2\pi\mu_0 \left(R\ln\frac{r_o}{r_i} \pm (r_o - r_i)\sin\vartheta\right) \tag{5}$$

We then calculate the total reluctance of the full flux tube:

$$R_m = \frac{1}{2\pi\mu_0 t} \int_0^{\pi} \frac{\mathrm{d}\vartheta}{\frac{R}{t}\ln\frac{r_o}{r_i} \pm \sin\vartheta} \tag{6}$$

With $\eta = \frac{R}{t}\ln\frac{r_o}{r_i}$ we find two cases for the primitive (Bronštein et al. 1993, P. 763, Eq. 306):

$$\int \frac{\mathrm{d}\vartheta}{\eta \pm \sin\vartheta} = \begin{cases} \frac{2}{\sqrt{\eta^2-1}} \arctan\frac{\eta\tan\frac{\vartheta}{2}\pm 1}{\sqrt{\eta^2-1}} & \text{for: } \eta > 1 \\ \frac{1}{\sqrt{1-\eta^2}} \ln\frac{\eta\tan\frac{\vartheta}{2}\pm 1-\sqrt{1-\eta^2}}{\eta\tan\frac{\vartheta}{2}\pm 1+\sqrt{1-\eta^2}} & \text{for: } \eta < 1 \end{cases} \tag{7}$$

We deviate from Bronštein et al. (1993) by stating the cases without the use of squares. Comparing η to 1 is equivalent to comparing η^2 to 1 since all radii are positive and $r_o > r_i$, so η is always positive.

2.1 Inner Half Hollow Torus (Negative Sign)

We first observe that only the case $\eta > 1$ exists for the inner half hollow torus. To see this, we consider the case $\eta = 1$:

$$\frac{R}{t}\ln\frac{r_o}{r_i} = 1 \tag{8}$$

$$\ln\frac{r_o/R}{r_i/R} = \frac{r_o}{R} - \frac{r_i}{R} \tag{9}$$

$$\ln\frac{r_o}{R} - \frac{r_o}{R} = \ln\frac{r_i}{R} - \frac{r_i}{R} \tag{10}$$

$\ln x - x$ is always negative, yet has a single high point at $x = 1$. For the inner half hollow torus we know the argument to be smaller than one, otherwise the flux tube could not exist without intersecting itself. Thus this function is strictly monotonic increasing for relevant arguments, and the only solution of Equation 10 hence is $r_i = r_o$. In this case, the flux tube ceases to exist and $R_m \to \infty$. This means, the only relevant case is $\eta > 1$.

Looking at the upper integration limit first, $\vartheta = \pi$. Then $\tan\frac{\vartheta}{2} \to +\infty$, while everything else in the argument of the arctan is finite, so the arctan returns $\frac{\pi}{2}$. Looking at the lower integration limit now, $\vartheta = 0$. Then $\tan\frac{\vartheta}{2} = 0$ and the argument of the arctan is negative, so we end up with:

$$R_m = \frac{1}{\pi\mu_0 t}\frac{1}{\sqrt{\eta^2-1}}\left(\frac{\pi}{2} + \arctan\frac{1}{\sqrt{\eta^2-1}}\right) \tag{11}$$

using the point symmetry of the arctan. With $G_{m0} = \pi\mu_0 t$ and using that the argument of the arctan is always positive, so we can use $\alpha_+ = \frac{\pi}{2} + \operatorname{arccot}\sqrt{\eta^2-1}$, we can write the permeance as:

$$G_m = G_{m0}\frac{\sqrt{\eta^2-1}}{\alpha_+} \tag{12}$$

All major results, like this permeance, are collected in Table 1 towards the end of the paper for quick reference.

We can finally verify the expected behaviour of the reluctance for $\eta^2 \to 1$ by considering $\eta^2 = 1 + \varepsilon^2$:

$$\lim_{\varepsilon\to 0}\frac{1}{G_{m0}}\frac{1}{\varepsilon}\left(\frac{\pi}{2} + \arctan\frac{1}{\varepsilon}\right) \to \infty \tag{13}$$

DOI 10.3384/ecp218243

2.2 Outer Half Hollow Torus (Positive Sign)

In turning to the outer half hollow torus now, we first note that in this case both $\eta > 1$ and $\eta < 1$ are physical. Consider first, $r_o = 2r_i = R$, i.e., $t = r_i = R/2$ and $\eta = 2\ln 2 > 1$. Consider next $r_o = 2r_i = 2R$, i.e., $t = r_i = R$ and $\eta = \ln 2 < 1$. The restriction for the inner half hollow torus exclusively came from the limited radial space towards the center – outwards there is unlimited space.

Looking at the case $\eta > 1$ first, the reasoning is much the same as before (here, $\alpha_- = \frac{\pi}{2} - \operatorname{arccot}\sqrt{\eta^2-1}$, allowing for the sign), leaving:

$$G_m = G_{m0}\frac{\sqrt{\eta^2-1}}{\alpha_-} \tag{14}$$

Looking at the case $\eta < 1$ now, we find at $\vartheta = \pi$ that both the tangens in the numerator and the denominator of the logarithm's argument diverge, i.e. the argument approaches unity and the logarithm vanishes. At $\vartheta = 0$, both tangens vanish, and we are left with:

$$R_m = -\frac{1}{2G_{m0}}\frac{1}{\sqrt{1-\eta^2}}\ln\frac{1-\sqrt{1-\eta^2}}{1+\sqrt{1-\eta^2}} \tag{15}$$

We use $\lambda = \ln\frac{1+\sqrt{1-\eta^2}}{1-\sqrt{1-\eta^2}}$ to write the permeance concisely:

$$G_m = G_{m0}\frac{2\sqrt{1-\eta^2}}{\lambda} \tag{16}$$

Note that the reciprocal in the logarithm's argument cancels the sign in Equation 15. Both terms may not be evaluated at $\eta = 1$, yet other than before they do not diverge but match up continuously. To see this, we start from the equation for $\eta > 1$, and use $1 = \eta^2 - \varepsilon^2$, arriving at:

$$\lim_{\varepsilon\to 0}\frac{1}{G_{m0}}\frac{1}{\varepsilon}\left(\frac{\pi}{2} - \operatorname{arccot}\varepsilon\right) = \frac{1}{G_{m0}} \tag{17}$$

Similarly, using $\eta^2 = 1 - \varepsilon^2$ for the equation for $\eta > 1$ yields:

$$\lim_{\varepsilon\to 0}\frac{-1}{2G_{m0}}\frac{1}{\varepsilon}\ln\frac{1-\varepsilon}{1+\varepsilon} = \frac{1}{G_{m0}} \tag{18}$$

as above. So all these cases give physical solutions, and they match continuously.

2.3 Inner and Outer Quarter Hollow Torus

The equations for the inner and outer quarter hollow torus immediately follow from the observation, that their reluctance will be half that of the corresponding half hollow torus, their permeance will be twice that of the corresponding half hollow torus.

2.4 Lower Half Hollow Torus

The lower half hollow torus then follows from these:

$$R_m = \frac{\alpha_-}{2G_{m0}\sqrt{\eta^2-1}} + \frac{\alpha_+}{2G_{m0}\sqrt{\eta^2-1}} \tag{19}$$

$$G_m = G_{m0}\frac{2\sqrt{\eta^2-1}}{\frac{\pi}{2} - \operatorname{arccot}\sqrt{\eta^2-1} + \frac{\pi}{2} + \operatorname{arccot}\sqrt{\eta^2-1}} \tag{20}$$

$$= G_{m0}\frac{\sqrt{\eta^2-1}}{\pi/2} \qquad \text{for: } \eta > 1 \tag{21}$$

We do not need to consider other cases, as the inner quarter torus only exists for $\eta > 1$.

3 Check with Finite Element Method

In order to evaluate the applicability of these formulae, we want to compare them with finite element method (FEM) calculations (Meeker 2025). This is necessary since on our way here we made two separate steps: we firstly *assumed* a flux pattern and we then secondly did the appropriate calculations to solve for this assumption. We did so far, however, not verify that the assumed flux patterns are correct in the first place. *Are* the flux lines exactly shaped like half circles? The prior results are only valid insofar as this is a reasonable approximation of reality.

To verify the flux pattern, we compare with FEM. The way the simulation is set up, only three areas are actually modelled (cf. Fig. 4): First, the flux tube in question, modelled as air ($\mu_r = 1$). Second, the area inside r_i, modelled as an insulator ($\mu_r = 10^{-6}$) containing a current linkage of, in this case, $\Theta = 1$ A. Third, a yoke serving as a magnetic short ($\mu_r = 10^{12}$). All outside areas are cut off by Dirichlet boundary conditions ($A = 0$). We then evaluate the integral over B_n in the center of the yoke, thus making sure that the flux lines at the point of integration have minimal curvature and are expected to yield a reliable value of Φ. The permeance is finally computed from:

$$G_m = \frac{\Phi}{\Theta} \tag{22}$$

Since all flux patterns scale, we effectively have to consider only two variables instead of three (R, r_i and r_o). For FEM we fix $R = 1$ mm for all calculations and use r_i/R as independent variable for our plots. We show different values of r_o/R as a family of curves, taking care to keep $r_i < r_o$ and $r_o < R$ where appropriate. Note that the quarter hollow torus elements (d) and (e) were not separately studied, as they are exactly half of (a) and (c).

Three exemplary results of such analysis are shown in Figure 4. Apart from the permeance, these pictures give us valuable clues with respect to the previously raised question: *are* the flux lines half-circle shaped? Even though we dictate the outer limit to be a half-circle, this is not exactly the case. In Figure 4a, e.g., it is quite evident that the distance of the outer-most flux line from the white surrounding area is *not* constant. Similarly, it is obvious, that

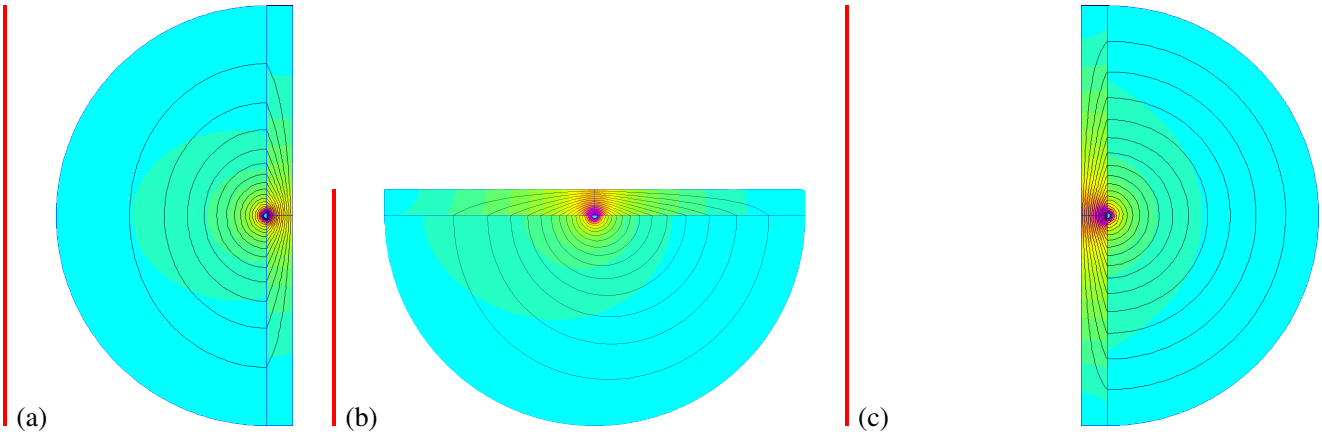


Figure 4. FEM results for $r_o/R = 0.8$ and $r_i/R = 0.01$. (a) inner half hollow torus. (b) lower half hollow torus. (c) outer half hollow torus. The left vertical red lines indicate the rotational axes.

the false colour map (showing the absolute of the flux density) does not exactly mimic the circular symmetry of the flux tube. The teal area is horizontally elongated in Figure 4a, while it is vertically elongated in Figure 4c. So we do well to remain cautious.

Next we study the values for G_m from FEM, from our formulae and from the formulae presently implemented in Modelica. The results are shown in Figure 5.

Let us consider the inner and outer half hollow torus first. Both these are compared to `Modelica.Magnetic.Fluxtube.Shapes.Leakage.HalfHollowCylinder`, which essentially assumes that the half hollow torus is a half hollow cylinder wrapped around a cylinder:

$$G_m = 2\mu_0 R \ln\left(1 + \frac{t}{r_i}\right) = G_{m0}\frac{\eta}{\pi/2} \tag{23}$$

The latter is written using the abbreviations introduced above. Comparing this to our expression for G_m ($\eta > 1$) requires $\sqrt{\eta^2 - 1} \to \eta$ in the numerator and $\operatorname{arccot}\sqrt{\eta^2 - 1} \to 0$ in the denominator. Both are true for $\eta \to \infty$, i.e., the presently used Equation 23 holds up for:

$$\frac{R}{t}\ln\frac{r_o}{r_i} \to \infty \tag{24}$$

In this limit, it similarly approaches our expressions for either half hollow torus (the sign of the arccot in the denominator is inconsequential, as in the considered limit the entire arccot is neglected). So our analytical result is not at odds with the present practice, but is a consistent extension to arbitrary parameter choices.

We intuitively expect Equation 23 to be fine for $R \gg r_i$ and $R \gg r_o$ (the former is noted in `Modelica.Magnetic.Fluxtube.Shapes.Leakage.HalfHollowCylinder` as a condition for cylindrical poles). What we find here is, that the actual condition is $R \gg (r_o - r_i)$ while at the same time r_o markedly larger than r_i (if the latter is not the case, the logarithm will be small). So the former condition cannot be fulfilled by making both r_o and r_i large while keeping their difference small.

In Figure 5a and 5c we find a reasonable fit for $r_o/R = 0.1$, however, for larger r_o/R we find considerable deviations even for $r_i/R = 0.01$. So if stating the above condition using a single simple equation, $R \gg r_o$ might be more useful than $R \gg r_i$ (since $r_i < r_o$, the former includes the latter anyway).

Considering the lower half hollow torus now, we find a quite reasonable fit with the equation presently used in Modelica (Roters 1941, P. 139, Eq. 22a),[2] which after substituting the appropriate symbols actually is identical to Equation 23.

This actually gives quite reasonable results with the exception of r_o/R close to unity, since the systematic errors of the inner and outer bit mostly cancel each other out. Note however, that in this case, our exact result is not much more complex, yet gives consistently good results for all parameter combinations.

For all three cases, our analytical result shows excellent fit over the studied parameter range. There actually are deviations from exactly circular flux lines, however, these do not result in notable differences in permeance. In order to quantify this statement, we in Figure 6 show the relative deviations at $r_o/R = 0.1$. For high r_o/R the shortcomings of the presently implemented approach are obvious. What we want to do here is to make sure that our analytical approach is at least as good as the presently used approach where there are no obvious problems with the latter.

From this comparison we find that even for $r_o/R = 0.1$ our approach is systematically better than the presently used formula. In the case of the inner and outer half hollow torus, it is better in terms of the relative deviation from the FEM results by at least one order of magnitude. In the

[2] There are two cases given in (Roters 1941) which actually are used to decide which bit of open pole surface does not count towards the useful radius sector, depending on whether the inner or outer useful radius difference is larger. We do not note these cases here as we explicity only use the appropriate radius sectors here, by starting from R and drawing two half circles with radii r_i and r_o centered there.

DOI
10.3384/ecp218243

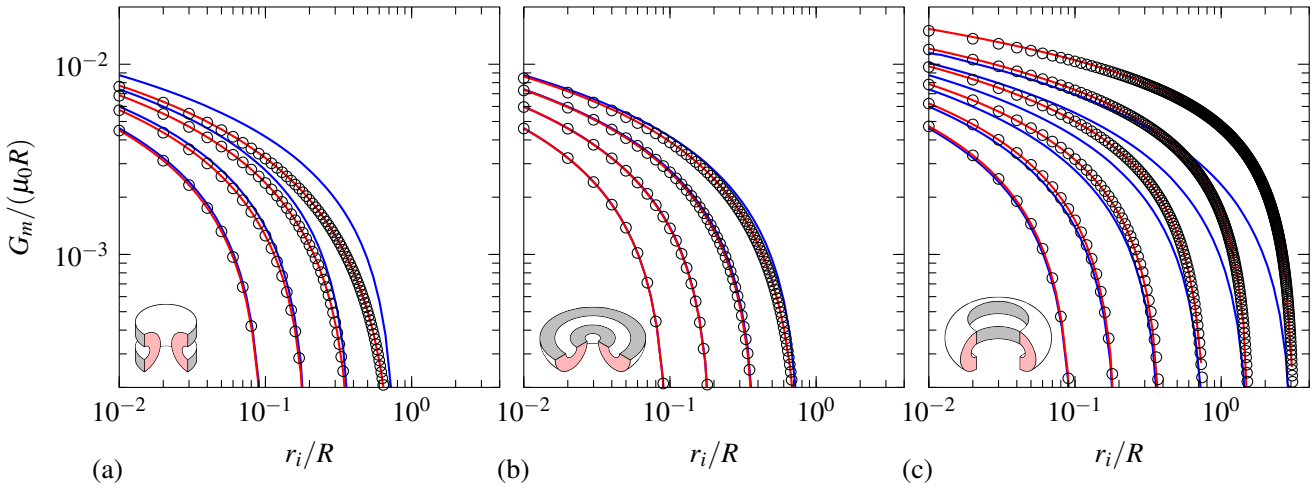


Figure 5. Results for the permeance. (a) inner half hollow torus. (b) lower half hollow torus. (c) outer half hollow torus. Each left to right (bottom to top): $r_o/R = 0.1, 0.2, 0.4, 0.8, 1.6, 3.2$ (the latter two only in (c)). Circles mark individual FEM results, red lines are analytical formulae presented here, blue lines are formulae presently used in Modelica.

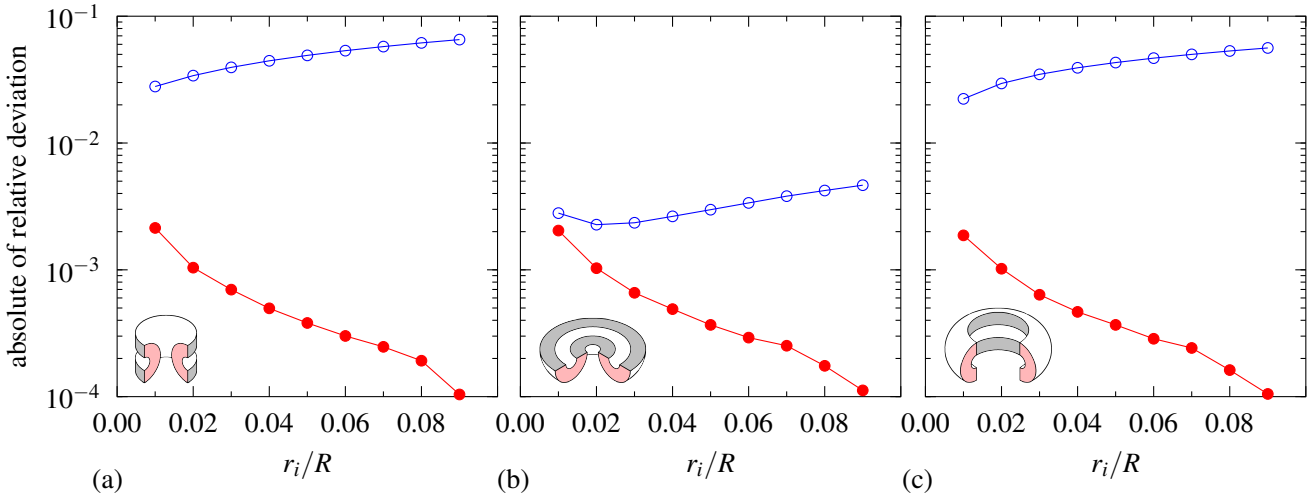


Figure 6. Relative deviations from FEM results for $r_o/R = 0.1$. (a) inner hollow hollow torus. (b) lower half hollow torus. (c) outer half hollow torus. Red filled symbols are analytical formulae presented here, blue open symbols are formulae presently used in Modelica.

case of the lower half hollow torus, the deviations partly cancel, especially for $r_i/R = 0.1$ the result is almost as good as the one presented here. Still, we can conclude that over the studied parameter range the formulae presented here give consistently better results than the status quo.

4 Equations for Force

Having analytical expressions for the permeance of flux tube geometries (a) through (e) (cf. Fig. 2) puts us in a position to also provide analytical expressions for forces generated by their deformation, e.g., by a moving armature. We use (Roters 1941, P. 197, Eq. 2a):

$$F = \frac{1}{2} V_m^2 \frac{\mathrm{d}G_m}{\mathrm{d}g} \tag{25}$$

which in Modelica is implemented in `Modelica.Magnetic.FluxTube.BaseClasses.Force`. To utilize this, we only need to supply an equation for the latter derivative, `dGmBydx`. We will, however, have to address a number of different cases, as for computing the derivatives it is quintessential to specify, which quantities are allowed to change with the air gap, g. We will walk through the relevant cases to illustrate what is meant by this.

The first relevant case is $r_o = \text{const}$. This case is appropriate if there is a natural limit to the radial extent of the flux tube. For an inner hollow torus this might simply be $r_o = R$. For an outer hollow torus this might be due to another part of the magnetic circuit, which would attract flux lines outwards from a given radius. A typical example might be the red flux tube in Figure 7a. At a point, where $\frac{\pi}{2} r_o$ equals the radius difference of the shown yoke parts, flux lines would short out to the outer part of the yoke

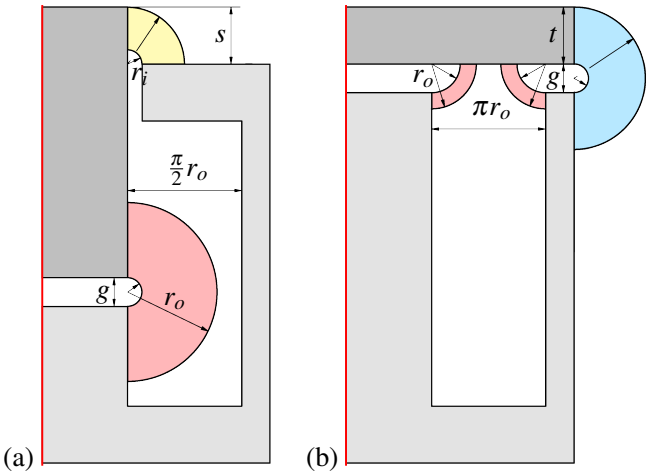


Figure 7. Application cases for force calculation. (a) plunger type solenoid. (b) flat face type solenoid. Light gray is the yoke, dark gray the armature. The left vertical red lines indicate the rotational axes. Red flux tubes are of type $r_o = \text{const.}$, blue of type $t = \text{const.}$, yellow of type $r_i = \text{const.}$

rather than arcing over this outer half hollow torus. Analogous cases will occur for inner half hollow tori, if there are equivalent hollow cylinder parts facing each other. A similar example for an inner and outer quarter hollow torus might be seen in the red flux tubes in Figure 7b, where again the radius difference within the yoke gives an upper limit to what outer radius such a toroid flux tube could have.

The second relevant case is $t = \text{const.}$ This is the case presently implemented in `Modelica.Magnetic.FluxTubes.Shapes.Force.LeakageAroundPoles`. This case is appropriate, if the stray flux tube is limited by a given axial width t of either one or both of the facing elements. As an example, consider the blue flux tube in Figure 7b. While the armature moves, $r_i = g/2$ will change, t will stay constant and $r_o = r_i + t$ will change accordingly.

Flux tubes of this type again may occur in the shape of inner or outer, half or quarter hollow tori. In discussing the relevant derivatives below, though, we will not consider the quarter hollow tori separately, as nothing qualitatively new happens when doing that. The force will simply be quadruple what we find in a half hollow torus, since for the quarter hollow torus twice the magnetic tension per length is applied (i.e., twice the magnetic field strength), while twice the distortion is effected on a quarter hollow torus for given stroke, s, as compared to a half hollow torus. Since for the derivatives, we only need the change of stroke, ds, it is allowable to identify $r_i = s$ for $t = \text{const.}$ and $r_o = \text{const.}$, and $r_o = s$ for $r_i = \text{const.}$

The latter is the third relevant case, and this case will mostly occur for inner or outer quarter hollow tori.[3] The technical situation might be a cylindrical plunger in a cylindrical hole, so an outer quarter hollow toroid flux tube is restricted in r_o and t, while at the same time r_i (being the fixed radius difference between plunger and hole) is constant. This situation is visualized by the yellow flux tube in Figure 7a. The same type of flux tube may occur on the inside, e.g., if the face of the plunger is lowerd into a hole in the yoke.

Note, that since in axisymmetric geometries motion is expected along the rotational axis, lower half hollow toroid flux tubes will generally not generate force (while maintaining this general geometry). Looking at the case of two coaxial hollow cylinders moving relative to each other, e.g., the relevant flux tube would be expected to be significantly distorted away from the half hollow torus geometry considered here. We will therefore not give equations for these.

We will now proceed to look at the required derivatives. All the relevant results are collected in Table 1 for quick reference (formulae for quarter tori are only shown for $r_i = \text{const.}$ to keep the table concise; the equations for $r_o = \text{const.}$ and $t = \text{const.}$ may easily be derived from those shown, as pointed out above).

4.1 Inner Half Hollow Torus ($r_o = \text{const.}$)

Starting from Equation 12, the required derivative is of the structure:

$$\frac{\mathrm{d}G_m}{\mathrm{d}g} = \left(\frac{G_m}{G_{m0}} \frac{\mathrm{d}G_{m0}}{\mathrm{d}r_i} + G_{m0} \frac{\mathrm{d}G_m/G_{m0}}{\mathrm{d}\sqrt{\eta^2-1}} \frac{\mathrm{d}\sqrt{\eta^2-1}}{\mathrm{d}\eta} \frac{\mathrm{d}\eta}{\mathrm{d}r_i} \right) \frac{\mathrm{d}r_i}{\mathrm{d}g} \tag{26}$$

wherein we need the following derivatives:

$$\frac{\mathrm{d}G_{m0}}{\mathrm{d}r_i} = -\frac{G_{m0}}{t} \tag{27}$$

$$\frac{\mathrm{d}G_m/G_{m0}}{\mathrm{d}\sqrt{\eta^2-1}} = \frac{\left(\frac{\pi}{2} + \operatorname{arccot}\sqrt{\eta^2-1}\right) + \frac{\sqrt{\eta^2-1}}{\eta^2}}{\left(\frac{\pi}{2} + \operatorname{arccot}\sqrt{\eta^2-1}\right)^2} \tag{28}$$

$$\frac{\mathrm{d}\sqrt{\eta^2-1}}{\mathrm{d}\eta} = \frac{\eta}{\sqrt{\eta^2-1}} \tag{29}$$

$$\frac{\mathrm{d}\eta}{\mathrm{d}r_i} = \frac{1}{t}\left(\eta - \frac{R}{r_i}\right) \tag{30}$$

$$\frac{\mathrm{d}r_i}{\mathrm{d}g} = \frac{1}{2} \tag{31}$$

This might look somewhat intimidating, however, there are a lot of recurring terms that are needed to calculate the permeance, anyway, so this can actually be coded quite efficiently. In doing that, it becomes necessary to ascertain $r_o > r_i$. Once this is no longer valid, this flux tube ceases to exist, resulting in $G_m = 0$ and $\frac{\mathrm{d}G_m}{\mathrm{d}g} = 0$. The same is true for the outer half hollow torus. This check is unnecessary if $t = \text{const.}$ is used, as this way $r_o - r_i = t$ for $t > 0$ always results in an existing flux tube.

[3] It is possible to invent scenarios, where this might occur for half hollow tori. These scenarios, however, appear to be rather artificial. For this reason, we do not present them here. If needed, the required formulae may easily be produced starting from what is presented here.

DOI 10.3384/ecp218243

4.2 Inner Half Hollow Torus ($t = \text{const.}$)

The general procedure is much the same as above:

$$\frac{\mathrm{d}G_m}{\mathrm{d}g} = G_{m0}\frac{\mathrm{d}G_m/G_{m0}}{\mathrm{d}\sqrt{\eta^2-1}}\frac{\mathrm{d}\sqrt{\eta^2-1}}{\mathrm{d}\eta}\frac{\mathrm{d}\eta}{\mathrm{d}r_i}\frac{\mathrm{d}r_i}{\mathrm{d}g} \tag{32}$$

The first term in Equation 26 may be omitted, as $G_{m0} = \pi\mu_0 t$ is constant in this case. η we can therefore rewrite as:

$$\eta = \frac{R}{t}\ln\left(1+\frac{t}{r_i}\right) \tag{33}$$

As far as the derivatives themselves are concerned, only Equation 30 needs to be adjusted accordingly:

$$\frac{\mathrm{d}\eta}{\mathrm{d}r_i} = -\frac{R}{r_i(r_i+t)} \tag{34}$$

4.3 Outer Half Hollow Torus ($r_o = \text{const.}$)

In this case, we need to keep in mind to separate the cases $\eta > 1$, $\eta = 1$ and $\eta < 1$ (this was not necessary for the inner half hollow torus). Starting with $\eta > 1$ and using Equation 14 this time, the equations look much the same as before for the inner half hollow torus and we can keep using Equation 26. We only need to adjust Equation 28 to:

$$\frac{\mathrm{d}G_m/G_{m0}}{\mathrm{d}\sqrt{\eta^2-1}} = \frac{\left(\frac{\pi}{2}-\operatorname{arccot}\sqrt{\eta^2-1}\right)-\frac{\sqrt{\eta^2-1}}{\eta^2}}{\left(\frac{\pi}{2}-\operatorname{arccot}\sqrt{\eta^2-1}\right)^2} \tag{35}$$

Since the permeance is defined using cases, we will have to check for continuity between these. In order to facilitate this, we will give the full relevant equation:

$$\frac{\mathrm{d}G_m}{\mathrm{d}g} = \left(\frac{G_m}{G_{m0}}\frac{\mathrm{d}G_{m0}}{\mathrm{d}r_i} + G_{m0}\frac{\mathrm{d}G_m/G_{m0}}{\mathrm{d}\sqrt{\eta^2-1}}\frac{\mathrm{d}\sqrt{\eta^2-1}}{\mathrm{d}\eta}\frac{\mathrm{d}\eta}{\mathrm{d}r_i}\right)\frac{\mathrm{d}r_i}{\mathrm{d}g} \tag{36}$$

$$= \frac{G_{m0}}{2t\alpha_-}\left(\left(\frac{\eta}{\sqrt{\eta^2-1}}-\frac{1}{\eta^2\alpha_-}\right)\left(\eta-\frac{R}{r_i}\right)-\sqrt{\eta^2-1}\right) \tag{37}$$

For $\eta < 1$ the following derivatives need to be added to our pool:

$$\frac{\mathrm{d}G_m/G_{m0}}{\mathrm{d}\sqrt{1-\eta^2}} = -2\frac{\frac{2\sqrt{1-\eta^2}}{\eta^2}-\ln\frac{1+\sqrt{1-\eta^2}}{1-\sqrt{1-\eta^2}}}{\ln^2\frac{1+\sqrt{1-\eta^2}}{1-\sqrt{1-\eta^2}}} \tag{38}$$

$$\frac{\mathrm{d}\sqrt{1-\eta^2}}{\mathrm{d}\eta} = -\frac{\eta}{\sqrt{1-\eta^2}} \tag{39}$$

Again, we give the full relevant equation:

$$\frac{\mathrm{d}G_m}{\mathrm{d}g} = \left(\frac{G_m}{G_{m0}}\frac{\mathrm{d}G_{m0}}{\mathrm{d}r_i} + G_{m0}\frac{\mathrm{d}G_m/G_{m0}}{\mathrm{d}\sqrt{1-\eta^2}}\frac{\mathrm{d}\sqrt{1-\eta^2}}{\mathrm{d}\eta}\frac{\mathrm{d}\eta}{\mathrm{d}r_i}\right)\frac{\mathrm{d}r_i}{\mathrm{d}g} \tag{40}$$

$$= \frac{G_{m0}2}{2t\lambda}\left(\left(\frac{2}{\eta\lambda}-\frac{\eta}{\sqrt{1-\eta^2}}\right)\left(\eta-\frac{R}{r_i}\right)-\sqrt{1-\eta^2}\right) \tag{41}$$

We now turn to consider the limit $\eta \to 1$, $\eta > 1$, by Taylor expansion using $\eta^2 = 1+\varepsilon^2 \Rightarrow \sqrt{\eta^2-1} = \varepsilon$ or $\eta = 1+\frac{\varepsilon^2}{2}$. First, we consider:

$$\alpha = \frac{\pi}{2} - \operatorname{arccot}\varepsilon \approx \varepsilon - \frac{\varepsilon^3}{3} = \varepsilon\left(1-\frac{\varepsilon^2}{3}\right) \tag{42}$$

Since the factor in front of the brackets is the same for $\eta > 1$ and $\eta < 1$, we only consider the term in the brackets and find for $\varepsilon \to 0$:

$$-\frac{1}{3}\left(1+2\frac{R}{r_i}\right) \tag{43}$$

We now turn to consider the limit $\eta \to 1$, $\eta < 1$, again by Taylor expansion using $\eta^2 = 1-\varepsilon^2 \Rightarrow \sqrt{1-\eta^2} = \varepsilon$ or $\eta = 1-\frac{\varepsilon^2}{2}$. First we consider:

$$\lambda = \ln\frac{1+\varepsilon}{1-\varepsilon} = \ln(1+\varepsilon)-\ln(1-\varepsilon) \tag{44}$$

$$\approx 2\varepsilon + 2\frac{\varepsilon^3}{3} = 2\varepsilon\left(1+\frac{\varepsilon^2}{3}\right) \tag{45}$$

We now again only consider the term in the brackets of Equation 41 and finally arrive at the same expression:

$$-\frac{1}{3}\left(1+2\frac{R}{r_i}\right) \tag{46}$$

The limits therefore coincide at $\eta = 1$ and we can note in this case:

$$\frac{\mathrm{d}G_m}{\mathrm{d}g} = -\frac{G_{m0}}{2t}\frac{1}{3}\left(1+2\frac{R}{r_i}\right) \tag{47}$$

This flux tube (like all others) only exists, if $r_o > r_i$. Since in this case, we keep r_o constant while changing $g = 2r_i$, it becomes important to handle the case $r_o \leq r_i$ separately, resulting in $G_m = 0$ and $\frac{\mathrm{d}G_m}{\mathrm{d}g} = 0$.

4.4 Outer Half Hollow Torus ($t = \text{const.}$)

For the general structure of the solution we again use Equation 32. Starting with $\eta > 1$ and using Equation 14 we actually did compute all relevant derivatives before. Since we need to consider the limit $\eta = 1$ in a minute, we note the full expression of this term:

$$\frac{\mathrm{d}G_m}{\mathrm{d}g} = -\frac{G_{m0}R}{r_i r_o}\frac{\eta}{2\sqrt{\eta^2-1}}\frac{\left(\frac{\pi}{2}-\operatorname{arccot}\sqrt{\eta^2-1}\right)-\frac{\sqrt{\eta^2-1}}{\eta^2}}{\left(\frac{\pi}{2}-\operatorname{arccot}\sqrt{\eta^2-1}\right)^2} \tag{48}$$

For $\eta < 1$ we start from 16, do not need any additional derivatives either, and we again note the full expression:

$$\frac{\mathrm{d}G_m}{\mathrm{d}g} = -\frac{G_{m0}R}{r_i r_o}\frac{\frac{2}{\eta}-\frac{\eta}{\sqrt{1-\eta^2}}\ln\frac{1+\sqrt{1-\eta^2}}{1-\sqrt{1-\eta^2}}}{\ln^2\frac{1+\sqrt{1-\eta^2}}{1-\sqrt{1-\eta^2}}} \tag{49}$$

In both cases, the first term is identical. In order to check for continuity, we check the remaining terms. First, $\eta^2 = 1+\varepsilon^2 \Rightarrow \sqrt{\eta^2-1} = \varepsilon$ or $\eta = 1+\frac{\varepsilon^2}{2}$ as above:

$$\lim_{\varepsilon\to 0} \frac{\left(1+\frac{\varepsilon^2}{2}\right)}{2\varepsilon} \frac{\frac{\pi}{2} - \operatorname{arccot}\varepsilon - \frac{\varepsilon}{1+\varepsilon^2}}{\left(\frac{\pi}{2} - \operatorname{arccot}\varepsilon\right)^2} = \frac{1}{3} \tag{50}$$

Now for $\eta^2 = 1-\varepsilon^2 \Rightarrow \sqrt{1-\eta^2} = \varepsilon$ or $\eta = 1-\frac{\varepsilon^2}{2}$, again using the limit for λ stated earlier we find:

$$\lim_{\varepsilon\to 0} \frac{\frac{2}{1-\frac{\varepsilon^2}{2}} - \frac{1-\frac{\varepsilon^2}{2}}{\varepsilon} \ln\frac{1+\varepsilon}{1-\varepsilon}}{\ln^2\frac{1+\varepsilon}{1-\varepsilon}} = \frac{1}{3} \tag{51}$$

As expected, the curve is continuous and for $\eta = 1$ we can note:

$$\frac{\mathrm{d}G_m}{\mathrm{d}g} = -\frac{G_{m0}R}{3r_i r_o} \tag{52}$$

4.5 Inner Quarter Hollow Torus (r_i = const.)

Since the gap, g, is not meaningful for quarter hollow tori, we instead use the stroke, s, as discussed above (cf. Fig. 7a). A constant offset will not change the result, and we may simply use $r_o = s$ and $t = s - r_i$. We need to consider:

$$\frac{\mathrm{d}G_m}{\mathrm{d}s} = \frac{G_m}{G_{m0}}\frac{\mathrm{d}G_{m0}}{\mathrm{d}s} + G_{m0}\frac{\mathrm{d}G_m/G_{m0}}{\mathrm{d}\sqrt{\eta^2-1}}\frac{\mathrm{d}\sqrt{\eta^2-1}}{\mathrm{d}\eta}\frac{\mathrm{d}\eta}{\mathrm{d}s} \tag{53}$$

with:

$$\eta = \frac{R}{s-r_i}\ln\frac{s}{r_i} \tag{54}$$

Most of the derivatives we have noted before. We only need:

$$\frac{\mathrm{d}G_{m0}}{\mathrm{d}s} = \frac{G_{m0}}{t} \tag{55}$$

$$\frac{\mathrm{d}\eta}{\mathrm{d}s} = \frac{1}{t}\left(\frac{R}{r_o} - \eta\right) \tag{56}$$

We thus arrive at:

$$\frac{\mathrm{d}G_m}{\mathrm{d}s} = \frac{2G_{m0}}{t\alpha_+}\left(\sqrt{\eta^2-1} + \left(\frac{\eta}{\sqrt{\eta^2-1}} + \frac{1}{\eta\alpha_+}\right)\left(\frac{R}{r_o} - \eta\right)\right) \tag{57}$$

Note that this term is positive, other than the terms considered before. This is due to the fact that by moving the plunger into the hole, the permeance in this case increases, so this flux tube (by itself) actually acts to push the plunger out of the yoke (it of course is more than compensated by the radial flux contribution pulling the plunger into the yoke - the latter term is well-described already, though, and therefore not included here).

This term only exists for $r_o < R$. Since we use $r_o = s$ while changing s, it becomes important to handle the case $s > R$ separately (the permeance remains that of $r_o = R$, the force, however vanishes, since the permeance does not change anymore). Furthermore, this term only exists for $s > r_i$ (resulting in $G_m = 0$ and $\frac{\mathrm{d}G_m}{\mathrm{d}s} = 0$).

4.6 Outer Quarter Hollow Torus (r_i = const.)

In this case we again need to consider cases depending on η. We start with $\eta > 1$, can keep using Equation 53 and do not need any new derivatives. The desired result is:

$$\frac{\mathrm{d}G_m}{\mathrm{d}s} = \frac{G_m}{G_{m0}}\frac{\mathrm{d}G_{m0}}{\mathrm{d}s} + G_{m0}\frac{\mathrm{d}G_m/G_{m0}}{\mathrm{d}\sqrt{\eta^2-1}}\frac{\mathrm{d}\sqrt{\eta^2-1}}{\mathrm{d}\eta}\frac{\mathrm{d}\eta}{\mathrm{d}s} \tag{58}$$

$$= \frac{2G_{m0}}{t\alpha_-}\left(\sqrt{\eta^2-1} + \left(\frac{\eta}{\sqrt{\eta^2-1}} - \frac{1}{\eta\alpha_-}\right)\left(\frac{R}{r_o} - \eta\right)\right) \tag{59}$$

For $\eta < 1$, no new derivatives are needed for arriving at:

$$\frac{\mathrm{d}G_m}{\mathrm{d}s} = \frac{G_m}{G_{m0}}\frac{\mathrm{d}G_{m0}}{\mathrm{d}s} + G_{m0}\frac{\mathrm{d}G_m/G_{m0}}{\mathrm{d}\sqrt{1-\eta^2}}\frac{\mathrm{d}\sqrt{1-\eta^2}}{\mathrm{d}\eta}\frac{\mathrm{d}\eta}{\mathrm{d}s} \tag{60}$$

$$= \frac{2G_{m0}}{t}\frac{2}{\lambda}\left(\sqrt{1-\eta^2} + \left(\frac{2}{\eta\lambda} - \frac{\eta}{\sqrt{1-\eta^2}}\right)\left(\frac{R}{r_o} - \eta\right)\right) \tag{61}$$

As before we need to consider the case $\eta = 1$ by testing limits for both cases and coincidingly find:

$$\frac{\mathrm{d}G_m}{\mathrm{d}s} = \frac{2G_{m0}}{t}\left(1 + \frac{2}{3}\left(\frac{R}{r_o} - 1\right)\right) \tag{62}$$

5 Numerical Implementation

All of these formulae have been implemented in Modelica (Schmidt 2025). This process is mostly straightforward, only a few remarks might be in place.

Firstly, some flux tubes allow only for certain parameter combinations, and while the variable geometry changes, they may essentially cease to exist. This might be fixed by setting $G_m = 0$ and $\frac{\mathrm{d}G_m}{\mathrm{d}x} = 0$ in those cases, however, since in the base class `Force` a reluctance is produced by calculating the inverse of G_m, the latter needs to be set to an arbitrary small number rather than zero (we chose $G_m = 10^{-15}$ H). It is in the nature of absolute values, that none will be small as compared to every other value, thus there is an inherent (if practically small) risk in doing that.

Secondly, while mathematically the three given cases $\eta \lessgtr 1$ cover all eventualities, numerically one needs to keep some distance from $\eta = 1$. Otherwise, Modelica is needlessly forced to evaluate terms that analytically safely converge, yet numerically might introduce significant errors. Experimentally it is found that by reserving $\eta \in [0.999999, 1.000001]$ to the solution strictly correct for $\eta = 1$ only, such problems can be avoided. Since all three solutions merge continuosly, no harm is done by this.

Thirdly, what does the added accuracy and versatility cost in terms of computational time? We put this to the test using our `OuterHalfHollowTorusConstantt` and the presently implemented `LeakageAroundPoles`.

We will start by looking at the accuracy of the force calculation. While our class is exact within the assumption of circular flux lines, the presently implemented class

DOI
10.3384/ecp218243

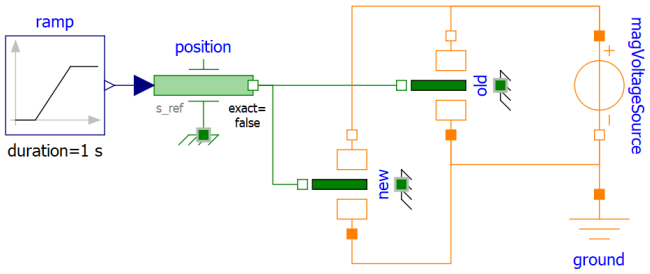


Figure 8. OMEdit test model used to compare the "new" model to the established "old" model of an outer half hollow torus.

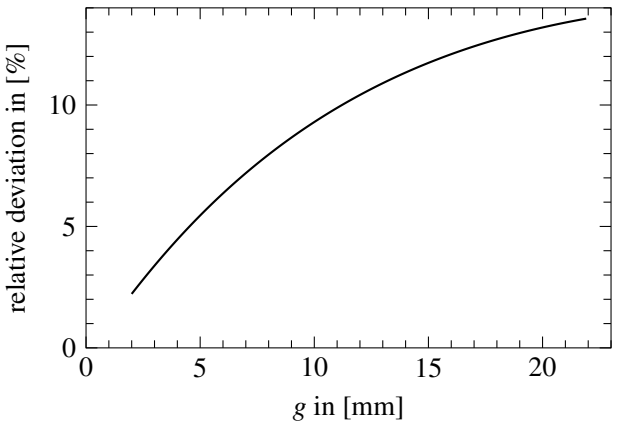


Figure 9. Relative deviation of force calculated from "old" compared to "new" model.

assumes bending a straight quarter hollow cylinder into a quarter hollow torus. This restricts the meaningful parameter choices. An additional, practical problem with `LeakageAroundPoles` is, which circumference to use? The class calls for the mean circumference to use as width, w, however, even if one accepts the arithmetic mean to be applicable, this mean changes as $r_o = r_i + t$ increases with increasing gap, $g = 2r_i$. Most likely, users will either put $2\pi R$, i.e., an obviously constant, yet systematically low value, or $2\pi(R + t/2)$, which might be expected to fit best for low values of the gap. Actually, the former gives quantitatively better results in our case, and we therefore went with $w = 2\pi R$. Other parameters were $t = R = 10$ mm.

We set up a model (cf. Fig. 8) using a prescribed positional ramp (20 mm stroke, 2 mm offset over 1 s) and a minimal magnetic loop including 1 A of magnetic tension to produce force vs. stroke curves. Figure 9 shows the relative deviation of these two curves. Note that the torus model went through all three cases $\eta \lessgtr 1$ within this ramp, without any trace of it in the result (as it should be).

`LeakageAroundPoles` systematically neglects that the width of the flux tube is a function of the gap. Including that would reflect into the derivative needed to calculate force and thereby change the class itself. This existing class deviates noticably from the more exact result. However, the significant relative deviation for large gaps is to be taken with a grain of salt, as absolute values of force in this region of stroke are very small.

Finally looking at the computational times listed in the transformational debugger (in OMEdit), we find the following break down of major contributions:

- 25.6 % for the position
- 13.3 % for the old force calculation
- 61.1 % for the new force calculation

At first glance, we find that our more elaborate model needs more than quadruple the computational time of the existing, simpler model. To put that into perspective: 12.2 % of the computational time is used up in calculating $\ln \frac{r_o}{r_i}$ alone. This illustrates that the absolute increase in computational time is not quite as significant as it may appear from looking at the relative increase. In absolute terms one may put it like this: the new force calculation is comparable in computational time to two position presets.

Whether this is acceptable may obviously be decided on a case-to-case basis. In most geometries considered in this paper, there is no "presently used model" anyway, so mostly such flux tubes would have previously been neglected or falsly been modelled using the one existing class, even though that would not really have been applicable (e.g., if r_o was constant rather than t).

6 Conclusion

In this paper we have derived exact expressions for the permeance of half or quarter hollow toroid flux tubes (cf. Tab. 1). These expressions are an extension of the much simpler expression extensively used previously, lifting the limitation to a certain range of allowable parameters. Our derivation started from the assumption that the flux lines follow a general circular pattern. We verified our result by studying the actual flux patterns using FEM analysis. Quantitatively we find our expressions for the permeance to be in agreement with the FEM results within the expected limitations of the numerical method. We furthermore derived expressions necessary for calculating force, taking into account three relevant use cases (constant outer radius, constant radius difference, and constant inner radius of the torus, respectively). Corresponding Modelica models have been made available electronically.

Acknowledgements

The author would like to thank Silvia Hacia and Jörg Frochte for fruitful discussions, as well as Christof Kaufmann for technical support.

References

Bronštein, Il'ja N. et al. (1993). *Taschenbuch der Mathematik*. 1st ed. Verlag Harri Deutsch. ISBN: 3-8171-2001-X.

Meeker, David C. (2025). *Finite Element Method Magnetics, Version 4.2 (21April2019 Build)*. URL: https://www.femm.info/ (visited on 2025-04-23).

Roters, Herbert C. (1941). *Electromagnetic Devices*. 1st ed. John Wiley & Sons. ISBN: 978-0471739203.

Schmidt, Herbert (2025). URL: https://github.com/HerbertSchmidt75/Toroid/ (visited on 2025-07-28).

Table 1. Essential formulae in simulating hollow toroid flux tubes.

Inner half hollow torus  for $\eta > 1$

$$G_m = G_{m0}\frac{\sqrt{\eta^2-1}}{\alpha_+}$$

for $r_o = \text{const.}$: $$\frac{\mathrm{d}G_m}{\mathrm{d}g} = -\frac{G_{m0}}{2t\alpha_+}\left(\sqrt{\eta^2-1} - \left(\frac{\eta}{\sqrt{\eta^2-1}} + \frac{1}{\eta\alpha_+}\right)\left(\eta - \frac{R}{r_i}\right)\right)$$

for $t = \text{const.}$: $$\frac{\mathrm{d}G_m}{\mathrm{d}g} = -\frac{G_{m_0}}{2r_o\alpha_+}\left(\frac{\eta}{\sqrt{\eta^2-1}} + \frac{1}{\eta\alpha_+}\right)\frac{R}{r_i}$$

Inner quarter hollow torus  for $\eta > 1$

for $r_i = \text{const.}$: $$\frac{\mathrm{d}G_m}{\mathrm{d}s} = \frac{2G_{m0}}{t\alpha_+}\left(\sqrt{\eta^2-1} + \left(\frac{\eta}{\sqrt{\eta^2-1}} + \frac{1}{\eta\alpha_+}\right)\left(\frac{R}{r_o} - \eta\right)\right)$$

Symbols

$$g = 2r_i$$

$$t = r_o - r_i$$

$$G_{m0} = \pi\mu_0 t$$

$$\eta = \frac{R}{t}\ln\frac{r_o}{r_i}$$

$$\alpha_\pm = \frac{\pi}{2} \pm \operatorname{arccot}\sqrt{\eta^2-1}$$

$$\lambda = \ln\frac{1+\sqrt{1-\eta^2}}{1-\sqrt{1-\eta^2}}$$

Lower half hollow torus  for $\eta > 1$

$$G_m = G_{m0}\frac{\sqrt{\eta^2-1}}{\pi/2}$$

Outer half hollow torus 

$$G_m = \begin{cases} G_{m0}\frac{\sqrt{\eta^2-1}}{\alpha_-} & \text{for: } \eta > 1 \\ G_{m0} & \text{for: } \eta = 1 \\ G_{m0}\frac{2\sqrt{1-\eta^2}}{\lambda} & \text{for: } \eta < 1 \end{cases}$$

for $r_o = \text{const.}$: $$\frac{\mathrm{d}G_m}{\mathrm{d}g} = \begin{cases} -\frac{G_{m0}}{2t}\frac{1}{\alpha_-}\left(\sqrt{\eta^2-1} - \left(\frac{\eta}{\sqrt{\eta^2-1}} - \frac{1}{\eta\alpha_-}\right)\left(\eta - \frac{R}{r_i}\right)\right) & \text{for: } \eta > 1 \\ -\frac{G_{m0}}{2t}\frac{1}{3}\left(1 + 2\frac{R}{r_i}\right) & \text{for: } \eta = 1 \\ -\frac{G_{m0}}{2t}\frac{2}{\lambda}\left(\sqrt{1-\eta^2} - \left(\frac{2}{\eta\lambda} - \frac{\eta}{\sqrt{1-\eta^2}}\right)\left(\eta - \frac{R}{r_i}\right)\right) & \text{for: } \eta < 1 \end{cases}$$

for $t = \text{const.}$: $$\frac{\mathrm{d}G_m}{\mathrm{d}g} = \begin{cases} -\frac{G_{m0}R}{r_i r_o}\frac{1}{2\alpha_-}\left(\frac{\eta}{\sqrt{\eta^2-1}} - \frac{1}{\eta\alpha_-}\right) & \text{for: } \eta > 1 \\ -\frac{G_{m0}R}{r_i r_o}\frac{1}{3} & \text{for: } \eta = 1 \\ -\frac{G_{m0}R}{r_i r_o}\frac{1}{\lambda}\left(\frac{2}{\eta\lambda} - \frac{\eta}{\sqrt{1-\eta^2}}\right) & \text{for: } \eta < 1 \end{cases}$$

Outer quarter hollow torus 

for $r_i = \text{const.}$: $$\frac{\mathrm{d}G_m}{\mathrm{d}s} = \begin{cases} \frac{2G_{m0}}{t}\frac{1}{\alpha_-}\left(\sqrt{\eta^2-1} + \left(\frac{\eta}{\sqrt{\eta^2-1}} - \frac{1}{\eta\alpha_-}\right)\left(\frac{R}{r_o} - \eta\right)\right) & \text{for: } \eta > 1 \\ \frac{2G_{m0}}{t}\left(1 + \frac{2}{3}\left(\frac{R}{r_o} - 1\right)\right) & \text{for: } \eta = 1 \\ \frac{2G_{m0}}{t}\frac{2}{\lambda}\left(\sqrt{1-\eta^2} + \left(\frac{2}{\eta\lambda} - \frac{\eta}{\sqrt{1-\eta^2}}\right)\left(\frac{R}{r_o} - \eta\right)\right) & \text{for: } \eta < 1 \end{cases}$$

DOI 10.3384/ecp218243

Master controller concept for power flexible energy systems

Thomas Egsgaard Kallesen[1] Søren Waagø Christiansen[1] Rene Just Nielsen[1]

[1]Added Values P/S, Denmark, {tep, swc, rjn}@AddedValues.eu

Abstract

A generic model of a master controller concept is presented to demonstrate the advantages of aggregating multiple flexible electrical power units. The master controller consists of various submodels with different features. A Power Balancing Controller ensures that the operation does not cause any imbalance with the traded power. A waterfall method that ensures that any ancillary service activations are handled using the appropriate units to obtain cheapest operation. Simulations of the concept include examples of step responses on automatic Frequency Restoration Reserve (aFRR) activations, as well as actual aFRR data from the Danish TSO. The results suggest that if the system consists of slow units, the aggregation of units can increase the capacity offered to faster types of ancillary services. Additionally, by allowing certain units to shut down automatically more capacity can be offered as the units are no longer bound by minimum loads.

Keywords: master controller, district heating, energy systems, ancillary services

1 Introduction

The expected continued expansion of and reliance on non-dispatchable renewable energy sources in the production of electricity has led to greater volatility in the power markets (International Energy Agency 2020). That is, production does not necessarily follow consumption as fewer dispatchable energy sources are available in electricity grids. In order to mitigate this increased volatility, weather forecasting has become an important tool used to predict the production capacity, which is used to set the prices on the day-ahead market.
However, uncertainties in weather forecasts will always cause deviations between planned production and consumption. Here, the importance of ancillary services comes in. They are standardized across the connected grids to ensure that enough capacity is connected to compensate for the uncertainties with the increasing share of renewable energy. The forecast deviation from actual production will only increase in the future. Hence, there is a need for more flexible grid-connected capacity that can participate in ancillary services (Energinet 2024b).

Multiple ancillary services are available on the Danish DK1 grid, which is part of the European Network of Transmission System Operators for Electricity – ENTSO-E (ENTSO-E 2018). DK1 is chosen as the reference in this paper, as it is the market of the authors' workplace. The ancillary services (AS) are distinguished by their response time requirements and the energy delivery that are inversely correlated (Energinet 2025b):

- **Frequency Containment Reserve (FCR)** requires response time from zero to activated load in 30 seconds, but is typically quickly returned to baseline load again resulting in average activations as ratio of capacity of 0.05 %. The load is proportional to grid frequency deviation.
- **Automatic Frequency Restoration Reserve (aFRR)** requires a response time from zero to activated load in 5 minutes. It has an average activation ratio below 12 % compared to the reserved capacity. The transmission system operator (TSO) uses a closed-loop controller to suppress the power imbalance, which makes it automatic, as the name implies.
- **Manual Frequency Restoration Reserve (mFRR)** requires a response time from zero to activated load in 12.5 minutes—typically through the balance responsible party (BRP).

These ancillary services can be performed both by electricity *consuming* or *producing* units as long as they can adjust the power at the required rate. There is even an economic incentive to participate, but this is not covered in this paper. However, often enough, electric units do not comply with any of the above requirements, and certainly not all of them (Danish Energy Agency 2025). Besides, many units have some minimum power limit they can operate at. Thus, some unexploited capacity is available in many power or power-to-heat (PtH) district heating plants that could be optimized with respect to utilization and speed. This can be done by combining the attributes of different units in the plant. For example, a fast but expensive unit can temporarily help a slower but cheaper unit increase its speed so that it can participate in faster ancillary services, such as aFRR. Furthermore, the power consumption or production in a plant can be distributed, so it allows units to shut down and start up dynamically. This gives additional regulation capacity to the plant while maintaining a continuous trajectory of the power.

Most of the literature on the aggregation of grid-connected units focusses on coordinating large populations of small household devices to provide ancillary

services. Within this field, (Iria, Soares, and Matos 2019) addresses the optimisation of day-ahead market bidding for aggregated resources. In terms of real-time control approaches, (Meyn et al. 2015) investigates the aggregation of large load populations using optimal control techniques, while (Bella et al. 2021) applies mixed-integer linear programming to the problem of load distribution. Similarly, (Iria and Soares 2019) introduces a fast model predictive control method, though it is still computationally heavy. A common feature of these studies is their focus on aggregating large numbers of small devices or prosumers to provide ancillary services. In contrast, (Tan and Zhang 2017) proposes a state machine approach for the combined control of a battery and wind power plant; however, this method is highly specific to that particular application.

The present work instead targets industrial-scale plants and aims to develop a general-purpose algorithm suitable for real-time control systems. The proposed approach emphasizes low computational requirements and operational transparency, ensuring that it remains accessible and comprehensible to plant operators, thus facilitating troubleshooting. By aggregating several power producing or consuming units with different technical characteristics, the strategy coordinates their setpoints to gain larger overall regulation capacity and faster response times. This is enabled by utilizing the fast response times of the more expensive units to aid the cheaper but slower units.
The final result is a structured Modelica implementation of the control concept, exemplified on a test system combining a heat pump, an electric boiler, and a battery. This implementation demonstrates both the internal architecture of the controller and the methodology applied to develop and validate it.

2 Master controller — working principle

The so-called *master controller* (MC) connects the electrical consumption or production units in the plant portfolio with the power grid, as illustrated in Figure 1.

The balance responsible party (BRP) is a third-party company responsible for power bidding and trading on behalf of the plant owner and for handling ancillary service requests from the transmission system operator (TSO). The BRP distributes AS activation signals from the transmission system operator directly to the master controller of its customers. A load scheduling system generates optimized base load setpoints for the individual plants, based on day-ahead prices, operating characteristics, plant availability, weather forecasts, and heat demand forecasts, among other factors.

Figure 2 illustrates the internal control concept of the master controller which can be divided into several parts introduced below. The parts are explained in the same order in which the setpoints are processed throughout the

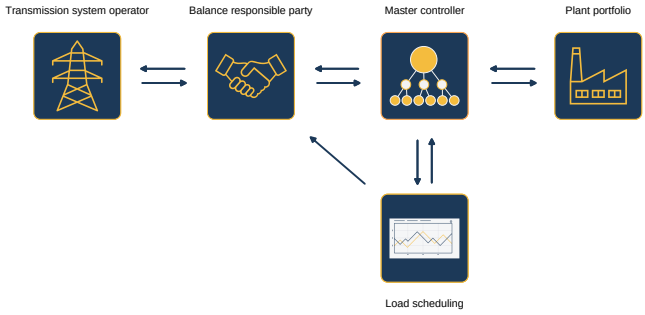


Figure 1. Simple overview of the master controller principles.

MC:

- **Load Schedule (LS)** handles the setpoints to each unit received from the load scheduling system. The setpoints are ramped according to the mFRR requirements and provide the operating point (baseline) of the system.
- **Balance Responsible Party (BRP)** handles ancillary service activations to the plant. The BRP setpoints are not individual for each production or consumption unit as the strategy requires the system to be considered as one aggregated unit. Each of the AS — FCR, aFRR and mFRR — are ramped according to their respective requirements, added together, and forwarded to the power balance controller.
- **Power Balance Controller (PBC)** ensures that the power consumption is equal to the traded power including AS. This is necessary since non-dispatchable consumption from pumps, lights, or similar ("house load") can be hard to predict. The PBC uses a closed-loop controller to steadily reach a zero deviation between the actual and traded power.
- **Waterfall method** distributes the AS and balance controller output to the units in a prioritized order. Higher-priority units respond to activations before lower-priority units.
- **Output signal handling** Each unit receives their load plan and the respective power correction from the waterfall method. Additional unit-related control such as output magnitude and ramp limitation, and unit start/stop is also handled here.

3 Model

The model is built of components from the Modelica Standard Library (MSL) (Modelica Association 2020) and blocks from an in-house developed library. It is divided into sub-models that correspond to each of the

DOI
10.3384/ecp218253

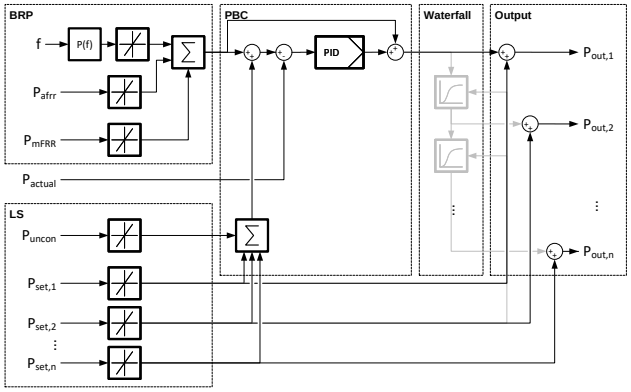


Figure 2. Overview of the master controller principles.

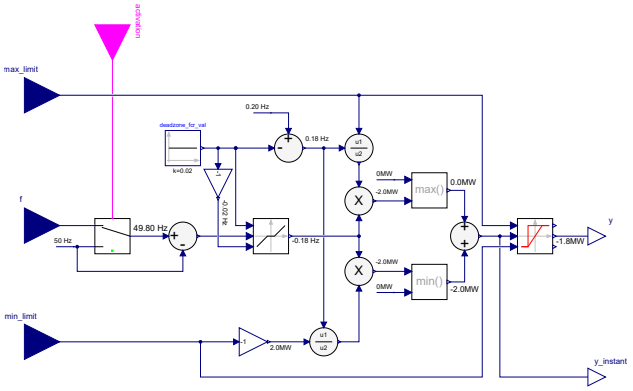


Figure 3. Diagram of FCR input handling

framed sections in Figure 2.

3.1 Input handling

Input handling comprises the BRP and LS functionality and the master controller inputs consist of load schedules and ancillary services signals. The load schedules are ramped to comply with the mFRR response times, and the resulting signal is routed to each corresponding unit to define their operating points. The ancillary services, which are later distributed to the units by the waterfall method, are ramped to meet their individual timing requirements. Thus, the aFRR input signal is ramped so that it can reach reserved capacity in less than 5 minutes. FCR is ramped so that the reserved capacity can be reached in 30 seconds. This differentiation in ramping, depending on the source of the activation, ensures that the reacting units and their components do not react faster than necessary.

Most setpoint sources are simply treated with a ramp. The exception is FCR, which is derived from the grid frequency deviation. Figure 3 shows the FCR calculation model. It converts the frequency measurement deviation from 50 Hz, into a proportional power response based on reserved capacity in the system. A rate limiter placed at the output performs the final setpoint shaping.

The input of the load schedule and the mFRR setpoint is ramped differently than the other ancillary services. The ramp times are set to 10 minutes to ensure compliance with the formal 12.5 minute requirements. mFRR responses that deviate from the 10 minute ramping are expressed as bills to the participant, depending on the current imbalance price. Therefore, these inputs are handled with the model shown in Figure 4, which is a modified rate limiter that adjusts the slope to ensure that the input value is reached after a prescribed time. It works by continuously calculating the slope required to reach the input from the current output and then inserting it into the rate limiter model. An *analogue memory* model in the

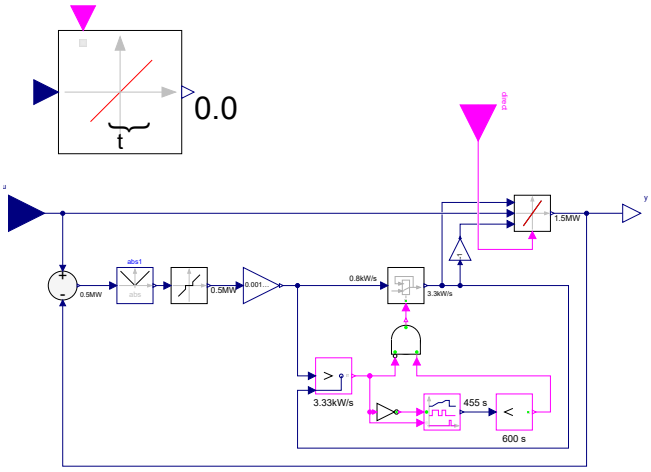


Figure 4. Modelica icon and diagram view of the constant timed ramp model including its icon.

centre of the figure saves the slope until a larger slope is derived from the calculation, at which point the updated maximum slope is replaced. With this method, the model works with any arbitrary input.

3.2 Power balance controller

The Power Balance Controller (PBC) compares the actual (measured) total power consumption with the combined setpoint of AS and the sum of LS setpoints. It uses an integrator followed by a PID controller to adjust the combined setpoint, so that the electric energy deviation becomes zero within a 15 minute horizon when the integrator is reset. See Figure 5. The 15 minutes cycle is introduced in the model because the BRP issues the electric power bill to the plant owners on the basis of 15 minutes energy accumulation.

Typically, it is difficult to predict the intermittent house load and, therefore, the PBC can help to precisely reach the setpoint, removing any possible economic consequences when deviating from the traded power. Some units have to actually change their consumption, but that is

distributed by the waterfall method (subsection 3.3). The controller output is added to the AS feed-forward setpoint as shown in Figure 2.

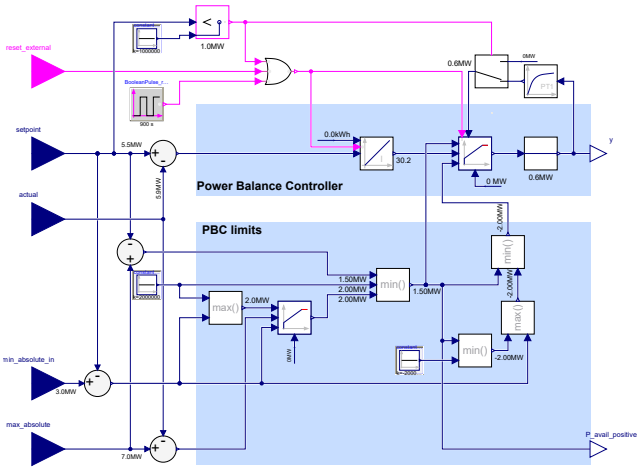


Figure 5. Modelica graphical model of the power balance controller.

In Figure 6, the system used later on in Section 4 is subjected to a step change in the ancillary services input. This is done both with and without the power balancing controller to compare the responses. Looking at the response without PBC, it is clear that there is a non-zero steady-state error. This is simply because, in this case, the system is essentially an open-loop system that does not correct the (non-dispatchable) house load of the system.

The steady-state error is cancelled when the PBC is activated. However, it introduces a relatively large overshoot that settles at the setpoint after 5–10 minutes. The overshoot is intentional and ensures that the accumulated energy deviation from the day-ahead energy purchase is zero over a 15 minute horizon. This means that steady state must be reached in 15 minutes, and that the overshoot actually compensates for energy that was not covered in the beginning of the response. This is illustrated by the two equally sized hatched areas. Normally, the overshoot is not going to be this large. It is purely due to the step input, which is not a realistic input and is chosen here to show the concept.

3.2.1 Power limitation

The power balance controller is also used to limit total power consumption or production. This is useful if faults occur on a transformer or at the electricity provider. When the power limit is reduced (input "max_absolute" in Figure 5) or the power consumption exceeds its limit, the power limitation proactively reduces the PBC output (via its upper limit input signal), potentially driving it to a negative value. This ensures that the sum of all

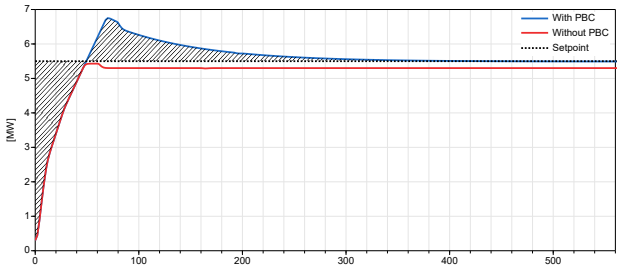


Figure 6. Demonstration of the power balance controller.

setpoints for the units is immediately reduced to comply with the transformer power limits. The power reduction is distributed to the units in the same way as the output from PBC.

If the setpoint is inaccurate compared to the actual power consumed or produced, then the above proactive (feed-forward) power limitation may be insufficient. Thus, an additional feedback controller is used to ensure that the actual power does not exceed the power limit. The minimum of the proactive power limitation and the controller limiter is the final maximum power used in the PBC.

3.3 Waterfall method

The *waterfall method* distributes one setpoint correction among several units according to a denoted priority associated with each unit. Generally, it means that one (total) setpoint is delivered to the unit with priority 1. The deviation between the setpoint and the *expected* dynamic response from the first unit is sent to the unit with priority 2. Similarly, the expected control deviation from unit 2 is forwarded to the unit with priority 3. This pattern is repeated until the setpoint is zero or the system runs out of available units.

The waterfall method is demonstrated with a system of three units: a slow heat pump as the first priority. A faster electric boiler as second priority, and a fast, but low-energy battery as third priority. The step responses from the output of the PBC sub-model to the individual units are shown in Figure 7. This corresponds to a 1 MW step change in ancillary services.

The heat pump receives the entire correction but has a large deviation at the beginning of the transition. The estimated deviation is passed on to the electric boiler which covers more of the transients. Lastly, the expected deviation of the electric boiler (shown as a short pulse) is passed as a setpoint to the battery The combined responses of all three units cover most of the delivery as shown with the output P_{total}.

This waterfall method consists of mainly two parts to

DOI 10.3384/ecp218253

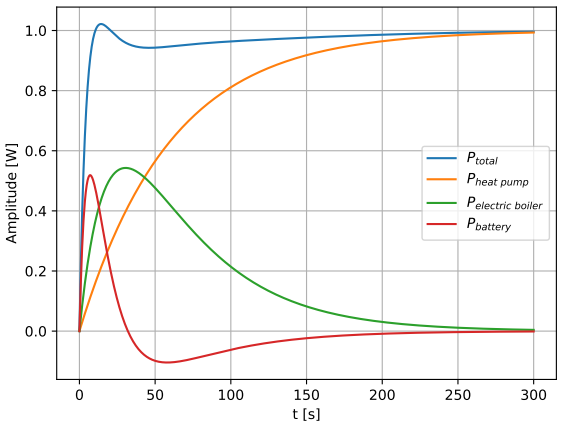


Figure 7. Step response of linear system with three units and two first order unit estimations.

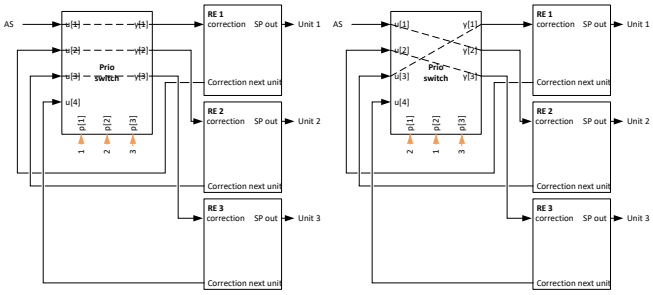


Figure 8. Different configurations of the priority switch

operate:

- **Priority switch** routes setpoints from the unit with first priority to the unit with the next priority and so on. It allows dynamic switching of priorities between units.
- **Response estimation** is the function that is repeated for each unit to estimate the power residual that *cannot* be covered by the unit. This residual is forwarded to the next unit in the order.

3.3.1 Priority switch

The priority switch is a function that connects each output to one of the inputs, such that the setpoint is handled by each unit in the order specified by the list of priorities. The priority switch is required in the case that different priorities are needed. otherwise, simple routing between each Response Estimation is sufficient. Figure 8 illustrates the layout of the waterfall method comprising the three mentioned units and shows the functionality with two different prioritization lists. In the figure to the left, the priority list is $\{heat\,pump = 1, boiler = 2, battery = 3\}$. This is the simplest configuration as it connects the outputs with the corresponding indexed input, indicated by the dashed lines inside the "prio switch" block. The priority list can be any incremented list of integers starting from 1 $(1 : n_{units})$. Thus, another valid case is $\{heat\,pump = 2, boiler = 1, battery = 3\}$ which is the case shown in the figure to the right.

Reversing The priority switch features a Boolean input that enables automatic reversal of the priority order during operation. This is useful because a participant may not want the highest priority unit to react as the first unit in both up- and downward directions. Using heat pump and electric boiler as an example, the heat pump produces much cheaper heat. Therefore, it is desirable to use the heat pump as priority 1 when increasing its load. But when the combined load is decreased, it is beneficial to reduce the load of the electric boiler before the heat pump. Therefore, when ancillary services move from a positive setpoint to a negative one, the priority order can be reversed to ensure the cheapest unit is always running. Additionally, the model gives the option to include only the first n units in the reversing. This is again useful if a system uses a battery to only adjust for the last corrections as the last priority. Then it is undesired if it suddenly acts as the first priority due to the reversing action.

3.3.2 Response estimation

It is commonly known that real measured responses introduce delays and more complex higher-order dynamics. While implementing a feedback control scheme for the priority switch inputs using real measured responses would be quite simple, introducing these delays in the control of the units would result in unnecessary activation of all prioritized units, which is undesirable. This is the case, because the delays in the measurements of the first priority unit will cause a small amount of the signal to be forwarded to the subsequent unit and so on, even if that unit is not necessary to meet the response requirements.

This phenomenon is also shown in the top graph of Figure 9. The figure shows a comparison between using real measured responses as feedback instead of estimations (top graph) and using an approach based on a rate limit block (bottom graph). The figure uses the example featuring a heat pump, an electrode boiler, and a battery energy storage system. In this test, the electric boiler is parametrized to meet the FCR requirements, eliminating the need for a faster battery in this specific case. However, it is clear that on the top graph, which uses real measurements, the battery still contributes, even though it is unnecessary.

For that reason, the dynamic response of the units is estimated with rate limit blocks as they react without any delay. Although the implementation is not as simple as using measured responses, using rate limit blocks prevents the lowest priority unit from being activated every time. This is shown in the bottom graph of Figure 9.

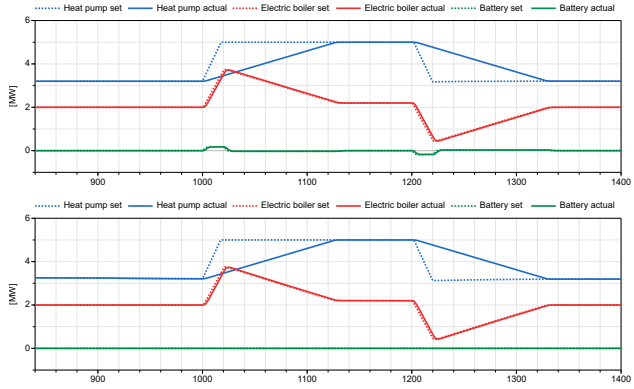


Figure 9. Comparison of waterfall method using real feedback (top figure) or estimated feedback (bottom figure).

The response estimation (RE) model utilizes information on ramping and limiting capabilities of the corresponding unit to estimate the response of the unit. The setpoint that is fed back to the priority switch to be routed to the next unit is the input setpoint subtracted by the estimated response of the unit. This can be formulated with Equation 1:

$$r_{corr,i+1} = (r_i + r_{corr,i})(1 - \hat{G}_{p,i}) \quad (1)$$

Where r_i is reference (load schedule) for unit i. $r_{corr,i}$ is the correction setpoint from the priority switch and $\hat{G}_{p,i}$ represents the estimation model of the unit. The combined setpoint that is sent to the unit is $u_i = r_i + r_{corr,i}$. A block diagram of the example with three units (without the priority switch) is shown in Figure 10.

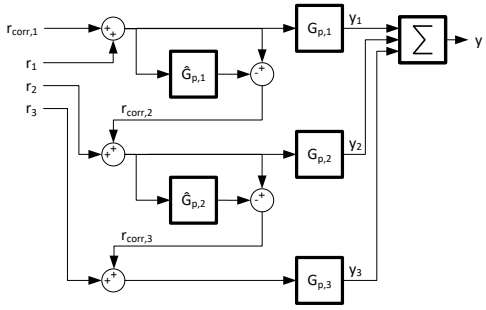


Figure 10. Block diagram of the waterfall principle

In the simulation, $\hat{G}_{p,i}$ is simply modelled with a power rate limiter and absolute limits of the unit. The rate limiter is sufficient to model the unit, as the master controller ramps the setpoint before it is sent to the unit. However, the unit typically also ramps the input itself if the input ramp is faster than the capability of the unit. Therefore, it is also important for this strategy to receive up-to-date information on the ramping capability of each unit as it is used as input in the response estimation.

If the actual ramping slopes differ from that used in the unit response estimations, the responses can become less accurate.

3.4 Automatic shutdown

A function can be implemented to automatically shutdown a unit in the output signal handling module mentioned in Section 2. This function can cause the unit to shut down when ancillary services combined with the planned setpoint reach the minimum limit of the unit. When this occurs, the waterfall method ensures that other units with lower priority temporarily replace the lost power during the ramp, resulting in almost continuous control.

3.5 Ancillary service validation

A validation function is modelled to ensure that activated ancillary services are within the response envelope required by the transmission system operator (TSO). The model calculates the upper and lower bounds required in the response and uses the constant-timed ramp model in Figure 4.

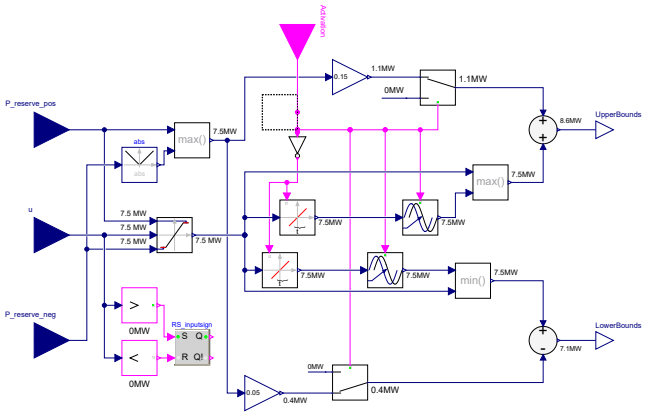


Figure 11. Modelica graphical model of the ancillary services validation.

This validation model (Figure 11) is based on the pre-qualification requirements in DK1, described in (Energinet 2025c). The model is applied to the setpoint components (load schedule, aFRR, mFRR) that cover the relevant ancillary services, after which the upper and lower bounds of each setpoint component are added together and plotted with the actual total response of the system for comparison. An example of bounds applied on a square-wave input setpoint is shown in Figure 12.

The envelope shows that the *increasing* source signal (the square wave) is allowed to rise immediately and must increase at least as fast as the blue ascending slope of the lower boundary. In the opposite direction, the source signal can decrease immediately and must decrease at

DOI 10.3384/ecp218253

least as fast as the blue descending slope. Similarly, the envelope allows for a certain steady-state deviation from the setpoint, indicated by the vertical margins. The validation model is especially useful when using realistic aFRR data that are updated continuously.

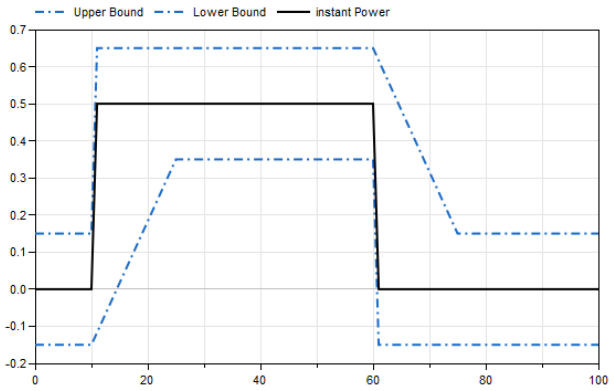


Figure 12. Modelica graphical model of the ancillary services validation envelope.

4 Simulation results

To demonstrate the functionality of the master controller, the concept is exemplified with three aggregated units in a PtH district heating system: a slow but inexpensive heat pump, a faster, but more expensive electric boiler and a battery with limited energy capacity. Participating in the AS markets requires some flexibility, so the example here assumes that a sufficient large heat storage tank can adjust for heat production variations.

The characteristics of the three units are shown in Table 1.

Table 1. Key model parameters.

Quantity	*Value*	*Unit*
Heat pump rate limit	$\pm$ 0.0125	MW/s
Electric boiler rate limit	$\pm$ 0.05	MW/s
Battery rate limit	$\pm$ 0.1	MW/s
Heat pump max limit	5	MW
Electric boiler max limit	5	MW
Battery max limit	1	MW
Heat pump min limit	2	MW
Electric boiler min limit	0.4	MW
Battery min limit	-1	MW
Heat pump priority	1	-
Electric boiler priority	2	-
Battery priority	3	-

Heat pump response times vary considerably depending on, e.g., compressor type. (MAN 2024) shows a heat pump that can participate in part in FCR and (Danish Energy Agency 2025) shows that heat pumps can typically ramp with 5 % in 30 seconds. Thus, to show the concept working on slower types of units, the rate limit of the heat pump in the table is selected to correspond to 400 seconds. (PARAT 2020) is an example of a fast ramping electric boiler with full FCR capabilities. Instead of 30 seconds, the selected rate limit in the table corresponds to 100 seconds, to show that the performance of the equipment does not need to be maxed out. Based on the notion that batteries regulate fast, 10 seconds is selected for the battery. Thus, the rates have been selected so that the heat pump is too slow for aFRR activations, whereas the electric boiler is not.

Other notable features in the simulations are

- The heat pump and electric boiler are configured to swap priority when $r_{corr,1} < 0$.
- A simple State-Of-Charge (SOC) proportional controller is added to the battery setpoint to ensure the battery is within capacity limits of 1 MWh.
- The heat pump can automatically shut down as described in Section 3.4.

In the following sections, the system is simulated mainly with aFRR responses, since its requirements are more strict than mFRR. FCR is even more strict, but in DK1 the activated bids are smaller and less profitable than aFRR, and thus from an economic perspective not as interesting.

4.1 Up- and down regulation

This simulation adds an aFRR input to the control scheme. The combined load schedule is around 5 MW, and the aFRR signal contributes first with an additional 5 MW in upward direction followed by 4 MW in downward direction. This is close to the maximum capacity possible with the chosen set-up given the minimum limits of the units. See Figure 13.

The combined electric power consumption is shown on the top and stays within the allowed envelope during the simulation. The second plot shows how the response is distributed between units. The heat pump reacts first, followed by the electric boiler when the heat pump reaches its limit. The battery reacts when the electric boiler reaches its limit. During the load reduction, starting after 24 minutes, the heat pump and electric boiler swap priority, causing the electric boiler to back down first, followed by the heat pump and then the battery. The responses from different units are delivered without any notable bumps in the combined response.

Note that the battery only contributes when the other units reach the limits and not every time there is a correction to the overall setpoint. This shows the advantage of

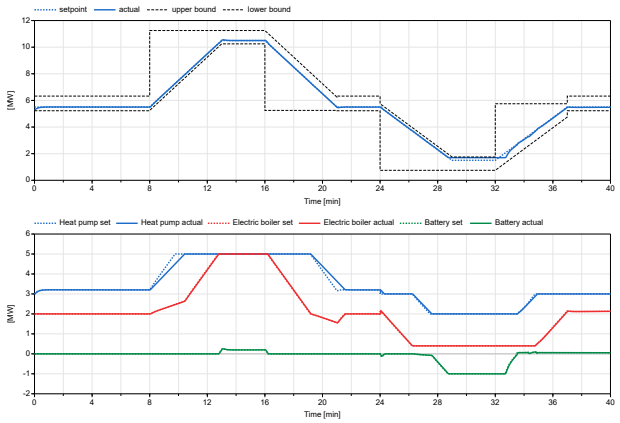


Figure 13. Simulation of an aFRR response on the example system.

using rate limiters to model expected responses.

4.2 Automatic startup and shut-down

The simulation test in Figure 14 shows the option of letting the heat pump automatically stop when the setpoint is below minimum. Similarly, it starts when the setpoint is above the minimum again. Thus, allowing the system to regulate in the downward direction with 5.5 MW, contrary to the 4 MW in Figure 13. In addition, the shutdown dynamics have been modelled to be faster than normal rates, whereas the startup rate is assumed to be equal to the continuous rate limit.

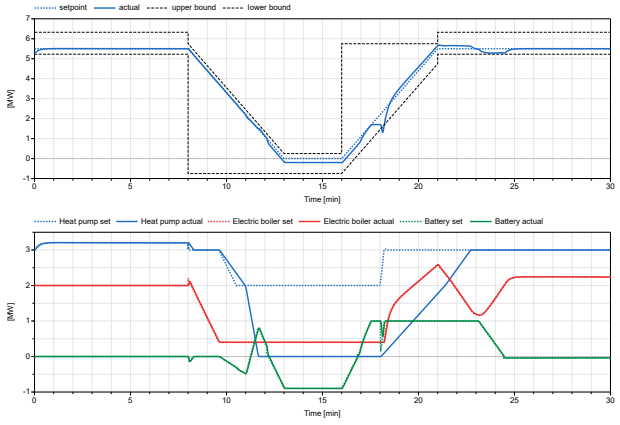


Figure 14. Simulation of auto-stop response on the example system.

Only minor bumps in power consumption are seen in combined consumption. However, after approximately 18 minutes, the total consumption decreases as the battery reaches its maximum power limit. With the current concept, the electric boiler cannot contribute from the waterfall method, since at this moment the priority is 1 for the electric boiler and 2 for the heat pump, due to the reversing. In this case, the PBC is utilized to increase the setpoint of the electric boiler before the total consumption deviates for a prolonged period of time.

4.3 Realistic aFRR activations

The entire system is tested with actual DK1 aFRR activations gathered from the Danish TSO data base (Energinet 2025a). The data covers ranges between ± 150 MW, but is then scaled with 0.035 to match the showcased capability (around 5 MW in each direction).

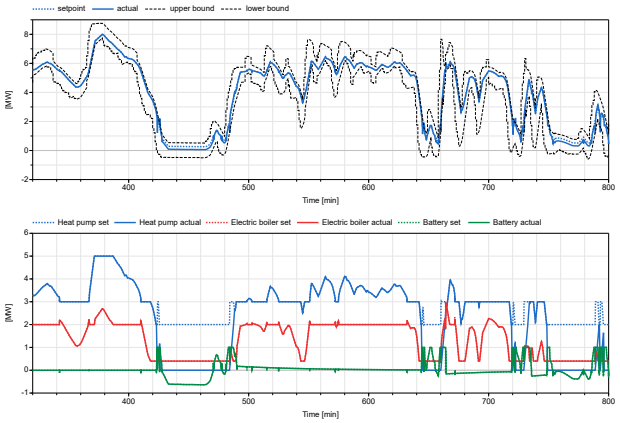


Figure 15. Simulation of an realistic aFRR response on the example system.

The simulation is shown in Figure 15 which shows several elements introduced in the paper. The waterfall method coordinates the units as expected even when using this continuous setpoint. Evaluation and visualization of the upper and lower boundaries of aFRR activations show that the response is, in fact, within the requirements. Occasionally stopping and starting the heat pump is not handled perfectly as there are some quick spikes in the consumption, but they stay within the boundaries. During the large load reductions, the automatic stopping of the heat pump happens fairly often. This could indicate potential issues with real heat pumps if they are not built for frequent on and off cycles. Lastly, the SOC controller for the battery is seen to apply small and slowly changing setpoints. This is compensated by some of the other units by the PBC.

5 Discussion

This work presents the concept of aggregating the electric power units of a PtH district heating plant. It shows that the master controller concept offers several advantages in terms of providing flexible operation to the grid through ancillary services. Since participation in ancillary service can be economically beneficial, it should be taken advantage of as much as possible. This work shows that the master controller concept provides the following potential advantages.

DOI
10.3384/ecp218253

- **Faster responses:** The power capacity in dynamically slow units can now participate in fast ancillary services.
- **Automatic shutdown:** Units with a relatively high minimum power level are no longer a limiting factor when aggregating, as long as there are other units that can replace its power level when it is off. If some units have a minimum power level of 50% (which is not uncommon (Danish Energy Agency 2025)), then the capacity in ancillary services can be practically doubled.
- **Power balancing:** Uncertainties in actual power consumption can be adjusted with the power balance controller to produce or consume the exact amount of power that has been traded, reducing the imbalancing bill.

It is important to note that this work is still quite theoretical, with multiple tests performed on estimated data instead of real-world data. However, the work presents a general structure of the master controller concept from which it is possible to create more tailored solutions based on the setup and needs of individual PtH plants.

The work highlights a potential economic benefit to be gained from implementing the different parts of the master controller. If a slow unit, such as a heat pump, cannot ramp fully within the 5-minute aFRR window, adding a 1 MW battery to help with transients increases the system's regulation capacity by 1 MW. The battery alone cannot be pre-qualified to deliver 1 MW due to its limited energy content (Energinet 2025c), so both units benefit from aggregation. The prices vary and evolve a lot, but at the time of writing, the price of the aFRR up-regulation capacity averaged around 70 €/MW/h throughout the year (Energinet 2024a). This can be directly translated into additional income when the battery is added to the portfolio. If it is assumed that the battery can only help half the time during 6 months of cold weather, then it is still possible to earn 151,000 € throughout the year. Additional income can be earned in the energy activation market, but it requires a more extensive analysis. An interesting next step would be to further explore this potential economic benefit with specific real-world PtH district heating plant setups and also adding actual imbalance costs, AS prices, etc.

The theoretical nature of the work means that some real-world complications are not taken into account. As mentioned, this work emphasises that some units can help other units decrease response time, but this is exclusively relevant if the response time is slower than the AS requirements to begin with. If all units are technically able to meet all requirements on their own, this concept could become less beneficial.

Another point made in this work is that it is possible to utilize a larger capacity by automatically shutting down units instead of being restricted by a minimum load on all units. Here, it has already been mentioned that this requires other units to help in meeting all requirements. However, the work does not account for any complications related to possible shutdown/startup restrictions. Such restrictions could potentially be due to, i.e., wear and tear, minimum downtimes during shutdown, or other unit obligations tied to shutdown/startup of the unit. As an example, it is quite common for some of the units used in the simulations (e.g., heat pumps) to have minimal downtime between operational cycles. In addition, there could be a limit on the number of shutdowns allowed within a day or other period of time. Any such restrictions are not taken into account in this work, but could cause the automatic shutdown function to be less useful.

6 Conclusion

This paper presents a master controller concept that aggregates electricity producing and consuming units of a power-to-heat district heating plant. The concept has shown potential to offer improved participation in ancillary services. Aggregating the units enhances the ability of some units to meet certain AS delivery time requirements. Using the described Waterfall Method, the faster units can assist the slower ones, ensuring that the overall response complies with the requirements. Thus, it is possible to increase the total capacity that can be offered to AS. Additionally, by utilizing the ability to completely shut down units, the aggregated system can also increase the total capacity offered to AS.

Along with these improvements associated with AS, the master controller concept can also increase performance on other parameters. The power balance controller component assists the system in maintaining zero deviation between purchased and used electricity. This can help prevent the undesirable costs associated with imbalances. The Waterfall Method in conjunction with the priority switch also enables the system to use the correct units for different AS activations. This entails that the cheapest units are utilized the most and the more expensive units are used last.

The potential benefits provided by the concept are shown using a simple example with a heat pump, an electric boiler, and a battery energy storage system. This example indicates that the concept offers a framework in connection with operating a power-to-heat district heating plant.

References

Bella, Alessio La et al. (2021). "A mixed-integer distributed approach to prosumers aggregation for providing balancing ser-

vices". In: *Electrical Power and Energy Systems* 133. DOI: 10.1016/j.ijepes.2021.107228.

Danish Energy Agency (2025). *Technology Data - Generation of Electricity and District Heating*. Tech. rep. URL: https://ens.dk/en/analyses-and-statistics/technology-data-generation-electricity-and-district-heating.

Energinet (2024a). *Årlig statistik for reserver og modhandel 2024*. Tech. rep. URL: https://energinet.dk/media/fi1iozqw/arlig-statistik-for-reserver-2024.pdf.

Energinet (2024b). *SCENARIERAPPORT 2024 – 2040 Forventninger til fremtidens Systemydelser*. Tech. rep. URL: https://energinet.dk/media/aj1frc1q/151124da_forventninger-til-fremtidens-systemydelser-2024-2040.pdf.

Energinet (2025a). *Energi Data Service*. Tech. rep. URL: https://www.energidataservice.dk/tso-electricity/PowerSystemRightNow.

Energinet (2025b). *Overview of Ancillary Services*. Tech. rep. URL: https://energinet.dk/el/systemydelser/introduktion-til-systemydelser/oversigt-over-systemydelser/.

Energinet (2025c). *Prequalification of Units and aggregated portfolio*. Tech. rep. URL: https://en.energinet.dk/media/gkijvg05/prequalification-of-units-and-aggregated-portfolios-february-2025.pdf.

ENTSO-E (2018). *Electricity Balancing in Europe*. Tech. rep. URL: https://eepublicdownloads.entsoe.eu/clean-documents/Network%20codes%20documents/NC%20EB/entso-e_balancing_in%20_europe_report_Nov2018_web.pdf.

International Energy Agency (2020). *Power Systems in Transition*. Tech. rep. URL: https://www.iea.org/reports/power-systems-in-transition.

Iria, José and Filipe Soares (2019). "Real-time provision of multiple electricity market products by an aggregator of prosumers". In: *Applied Energy* 255. DOI: 10.1016/j.apenergy.2019.113792.

Iria, José, Filipe Soares, and Manuel Matos (2019). "Optimal bidding strategy for an aggregator of prosumers in energy and secondary reserve markets". In: *Applied Energy* 238, pp. 1361–1372. DOI: 10.1016/j.apenergy.2019.01.191.

MAN (2024). *MAN heat pumps Transcritical compression cycle (TCC)*. Tech. rep. URL: https://www.man-es.com/docs/default-source/document-sync/man-heat-pump-hpu328c592b5d584f9eba47f4b6bed3574e.pdf?sfvrsn=9bd416fb_14.

Meyn, Sean P. et al. (2015). "Ancillary Service to the Grid Using Intelligent Deferrable Loads". In: *IEEE Transactions on Automatic Control* 60.11. DOI: 10.1109/TAC.2015.2414772.

Modelica Association (2020). *Modelica Standard Library*. Tech. rep. URL: https://github.com/modelica/ModelicaStandardLibrary.

PARAT (2020). *High Voltage Electrode boiler for Steam and Hot water*. Tech. rep. URL: https://www.parat.no/en/products/industry/parat-ieh-high-voltage-electrode-boiler.

Tan, Jin and Yingchen Zhang (2017). "Coordinated Control Strategy of a Battery Energy Storage System to Support a Wind Power Plant Providing Multi-Timescale Frequency Ancillary Services". In: *IEEE Transactions on Sustainable Energy* 8.3. DOI: 10.1109/TSTE.2017.2663334.

Modelling, Simulation and Validation of thermal propagation for 3D discretized battery cells in Modelica

Finn van Ginneken[1], Alexander Busch[2], Andreas Varchmin[1], Wilhelm Tegethoff[1,2], Juergen Koehler[2]

[1]TLK-Thermo GmbH
Rebenring 31, 38106 Braunschweig
f.van-ginneken@tlk-thermo.com

[2]Institut für Thermodynamik, TU-Braunschweig
Hans-Sommer-Straße 5, 38106 Braunschweig
alexander.busch1@tu-braunschweig.de

Abstract

This publication presents models for dynamic, 3-dimensional simulation of the thermal runaway, thermal propagation and the thermo-electric behavior of battery cells. An optional and replaceable electrical equivalent circuit diagram describes the electrical behavior of the battery cell. Coupled with the 3-dimensional, thermal equivalent circuit diagram, consisting of a resistance-capacitance network, the thermo-electric behavior is represented. A special feature of the battery cell model is the optional, replaceable and physical model of the thermal runaway based on Arrhenius equations and the associated mass loss. This means that either the thermo-electric or the thermal runaway behavior or both can be simulated in combination if required. The anisotropic thermal conductivity of the battery cell and the modelling of relevant heat conduction paths, such as the path across the housing of the battery cell, is made possible by 3D discretization of the battery cell.

Keywords: Lithium-Ion-Battery, thermo-electric Battery model, Arrhenius, Thermal Runway, Dymola, Modelica

1 Introduction

Lithium-ion batteries are used in electrochemical energy storage systems due to properties such as high cycle stability, energy density and capacity at comparatively low costs. However, thermal runaway (TR) can occur in lithium-ion batteries (Lenz et al. 2023; Walker et al. 2022; Groß und Golubkov 2021). This is a chain reaction in which exothermic decomposition reactions lead to exponential self-heating of the battery cell. This fault results in the destruction of the battery cell and, in worst case, can lead to the spread of the fault in the entire system, known as thermal propagation (TP). In addition to the release of heat, there is also a significant release of gas and particles during TR, which leads to a loss of mass in the battery cell. Suitably designed battery systems, particularly their thermal management, can limit the occurrence of faults and thus prevent a TR or at least a TP. The design can be carried out either by means of expensive and time-consuming experiments or by using suitable models (Walker et al. 2022; Groß und Golubkov 2021).

Two modelling approaches for TR are widely used in literature. The first approach is based on empirical heat release models, which can be further subdivided into time-dependent and temperature-dependent models (Hoelle et al. 2023). The advantage of empirical heat releases is the low parameterization and calculation effort (Groß und Golubkov 2021). An overview of other empirical models can be found in Hoelle et al..

The second modelling approach is Arrhenius equation-based models. These models are based on the publication by Hatchard et al. 2001. This model, originally based on two Arrhenius equations, has been extended by various authors for different cell formats and compositions (Kriston et al. 2019; Ren et al. 2018; Wang et al. 2022; MacNeil und Dahn 2001; Lenz 2024). Both modelling approaches (empirical, Arrhenius-based) have advantages and disadvantages. According to Groß und Golubkov the chosen empirical approach has the advantage that no fitting of the Differential Scanning Calorimetry (DSC) curves is necessary and the calculation effort is relatively low even for large battery packs. A decisive disadvantage of empirical modelling is the neglect of the chemical reactions and thus the lack of insight into the reaction cascade. In addition, the TR heat is only released above a certain temperature T_{TR} (Hoelle et al. 2023). As a result, the heat released in the battery cell before the temperature T_{TR} is not considered in the design of thermal management systems.

In addition to the heat release during TR, the modelling of normal operating states is necessary for the design of battery systems and in particular for thermal management (Epp et al. 2022; Epp et al. 2023). Therefore, models for the thermo-electric simulation of pouch and prismatic battery cells are also present. Modelica was chosen as modelling language due to its physical, object-oriented and equation-based characteristics. Several models and libraries are already available for modelling the electrical behavior of a battery cell in Modelica. Einhorn et al. 2013 compared three electrical equivalent circuit models in terms of parameterization effort and accuracy. However, no thermal effects were considered. Gerl et al. 2014 and Uddin und Picarelli 2014 focus on the thermo-electric

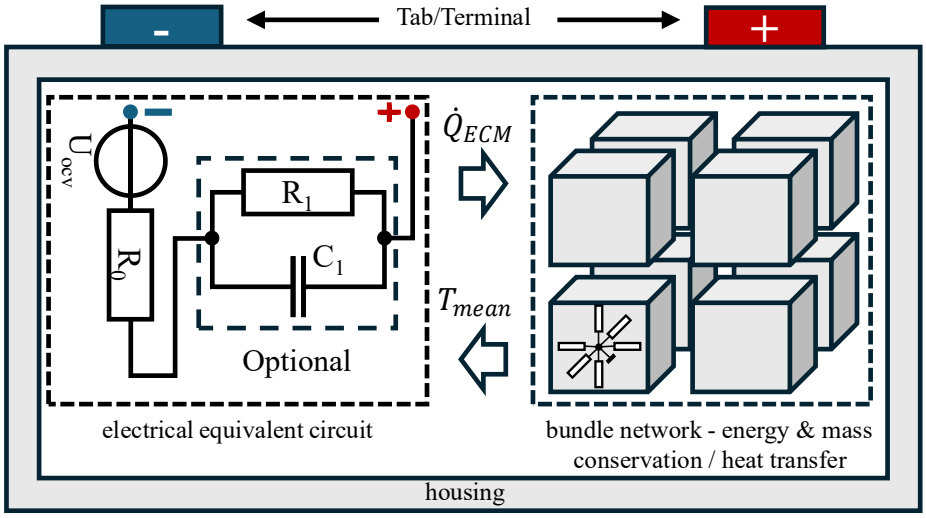


Figure 1. Schematic representation of a prismatic battery cell with the sub models ECM, bundle network, terminal/tab and the coupling of ECM and bundle network

behavior in the respective Modelica library, but do not consider the anisotropic thermal conductivity or the TR. Wendland et al. 2018 address the anisotropic thermal conductivity but does not offer the possibility to consider the TR for individual battery cells. In addition, a point mass can comprise more than one battery cell. The Modelica library by Groß und Golubkov 2021 enables an empirical simulation of the TR, but does not take into account either the anisotropic thermal conductivity or a possible temperature distribution in the battery cell. The library presented by Groß und Golubkov is, to the authors' knowledge, currently the only available library for simulating TR in Modelica.

The models presented in this publication are used for thermo-electric simulation including a TR based on the Arrhenius equation with optional 3D discretization of the battery cell. This allows the anisotropic thermal conductivity and other relevant thermal conduction paths of the battery cell to be considered. In addition, this discretization combines many advantages of high-dimensional flow simulations (CFD) and simplified point mass models.

2 Modelling

The models are combined in the Modelica package named *TIL3_AddOn_Battery*. The package is partly based on the *TIL Suite* and *TILMedia Suite* libraries from TLK-Thermo. In the following, elementary models are presented. Not all models of the package are covered in this publication.

2.1 The Battery Cell

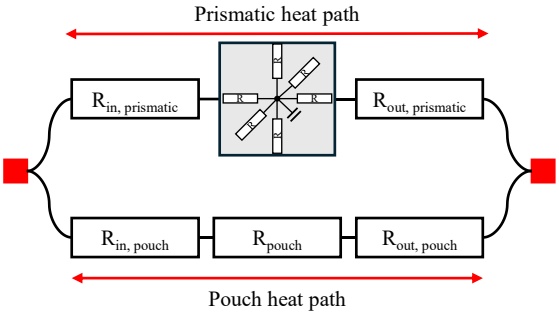
The battery cell model consists of various components (see Figure 1), which are described with sub models. The core of the battery cell is the so-called bundle network, which describes the active material, separators and current collectors in discretized form. No distinction is made between the individual active material layers, separators and current collector foils, so that the bundle network is only parameterized as one material. The bundle network is a replaceable model with 6 heat port matrices consisting of heat ports to enable thermal connection to the housing. The basic 3D control volumes, so-called bundles, are located within the bundle network (see section 2.2). These bundles are instantiated as a 3D matrix. The control volumes are thermally coupled in 3 dimensions in order to address the anisotropic thermal conductivity in the battery cell (Coman et al. 2017; Kim et al. 2021). The discretization is freely selectable.

The bundle network is surrounded by a housing (cf. Figure 1). Apart from the thickness, its discretization is based on the discretization of the bundle network. The housing is not discretized in thickness. A special feature of the housing is the non-structural change of the cell type (pouch-shaped, prismatic). As well as the consideration of the internal heat conduction paths for prismatic housing. For pouch-shaped battery cells, the housing consists of pouch foil. Due to the low foil thickness and to avoid the resulting small-time constants, the foil is modelled as a discretized thermal resistor. For the prismatic battery cell, the heat capacity is considered due to the significantly thicker housing (see Figure 2). By adjusting the internal and external thermal resistance, the heat conduction path can be virtually eliminated depending on the housing. In addition, these resistances are used to model inner/outer foils and to consider reduced thermal conductivity due to the safety valve and/or the terminals for the prismatic battery cell.

The electrical equivalent circuit model (ECM) (see section 2.4 and Figure 1) is globally (not discrete, one ECM per battery cell) integrated in the battery cell and electrically contacted via the two pins. The ECM was not implemented locally due to additional states. For the thermo-electric coupling, the volumetric mean value of the temperatures in the bundle state is used as input. The

DOI 10.3384/ecp218263

heat from the ECM is distributed volumetrically to all bundles. The heat distribution is a simplification; in the event of a thermal runaway, local effects such as short circuits can occur due to a breakdown of the separator (Kim et al. 2021); these effects cannot be modelled by the ECM.


Prismatic heat path
R
R
R

(see section 2.3), the ECM heat $\dot{Q}_{ECM}$ (see section 2.4) and the heat across the balance boundaries. For the enthalpy flow, it is assumed that the ejected mass carries the same specific enthalpy as the solid has at the time ($h_{s,out} = h_s$).

$$\begin{aligned}\frac{dU_s}{dt} = &\sum_{i=1}^{6} \dot{Q}_i + \dot{Q}_{ECM} \\ &+ m_s \dot{q}_{TR} + m_{s,initial} \dot{Y}_{rel,out} h_{s,out} \\ &- p \frac{dV_s}{dt}\end{aligned} \tag{6}$$

Equation 6 represents the energy balance according to the first law of thermodynamics, neglecting potential and kinetic energy. The relationship between the solid enthalpy H_s and the internal energy $U_s = H_s - pV_s$ leads to the following total differential.

$$\frac{dU_s}{dt} = \frac{dH_s}{dt} - p\frac{dV_s}{dt} - V_s\frac{dp}{dt} \tag{7}$$

Where p describes the pressure. Due to the pressure-independent properties of the solid, only $dU_s = dH_s - pdV_s$ remains. By inserting the relationship (cf. equation 7) into equation 6, the work of volume change pdV_s is reduced and the following total differential of the solid enthalpy remains on the left-hand side.

$$\frac{dH_s}{dt} = m_s\frac{dh_s}{dt} + h_s\frac{dm_s}{dt} \tag{8}$$

With the mass balance and the assumption for the enthalpy flow carried along, equation 6 can be simplified to the following implemented equation.

$$m_s\frac{dh_s}{dt} = \sum_{i=1}^{6} \dot{Q}_i + \dot{Q}_{ECM} + m_s\dot{q}_{TR} \tag{9}$$

2.3 The Thermal Runaway Model

Another central element of the battery cell is the calculation of the TR. For this purpose, a base class has been implemented to enable the replacement of different TR models. This means that new TR models can be easily integrated. The base class implementation targets Arrhenius equation based TR models, but can be easily adapted to empirical models, such as the one by Groß und Golubkov.

One of the most important inputs in the TR model is temperature. The decisive output is the mass-specific heat flow. This is determined by the change in the degree of conversion of the reactions. In literature, the degree of conversion is often referred to as dimensionless concentrations. The output and the implementation are illustrated below using the publication by Ren et al..

In the mentioned publication, DSC measurements are carried out for various battery cell components (cathode, anode, electrolyte). Using these measurements at different heating rates and the Kissinger or Flynn-Wall-Ozawa method, the kinetic triplet for specific DSC-peaks is derived (Ren et al. 2018). The decisive kinetic triplet (equation 10) consists of frequency parameter A_x, activation energy $E_{A,x}$ and reaction model f.

$$\begin{aligned}\kappa_x &= A_x \exp\left(-\frac{E_{A,x}}{RT(t)}\right) \\ \frac{d\alpha_x}{dt} &= -\kappa_x f \\ f &= \alpha_x^{n_x} \\ &\quad \cdot (1-\alpha_x)^{m_x} \\ &\quad \cdot (-\ln(1-\alpha_x))^{p_x} \\ \alpha_x(t=0) &= 1, \qquad \alpha_x \in [0,1]\end{aligned} \tag{10}$$

κ_x represents the respective conversion rate described by the Arrhenius equation. α_x is the degree of conversion of the approximated x-th DSC peak. f represents the kinetics-dependent reaction model, which describes the thermal decomposition of the solid. The exponents are used to represent different reaction models, such as the autocatalytic reaction ($n_x = 1, m_x = 1$) (Al-Khamis et al. 2010; MacNeil und Dahn 2001).

For the TR model of Ren et al., a power law reaction model with variable reaction order n_x is assumed for the reaction model ($p_x = 0, m_x = 0$). Six dominant DSC peaks are found in the publication and an ordinary differential equation (ODE) is implemented accordingly. The correct implementation can be verified by specifying a heating rate (via the temperature input) and then comparing the modelled with the measured DSC peak. The superposition of the modelled DSC peaks enables the determination of the resulting, mass-specific total heat flow $\dot{q}_{TR}$ (see equation 11). Figure 3 shows the simulated mass-specific total heat flow of the using equation 11 (for the DSC-simulation the reactive mass is set to one) compared to the DSC-measurement of the anode-electrolyte sample by Ren et al.

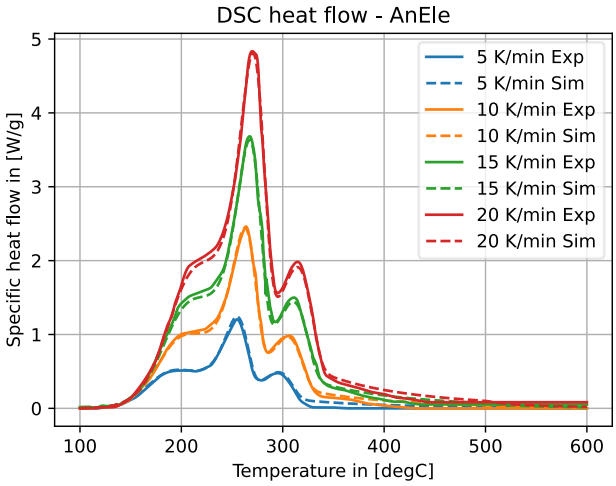


Figure 3. Comparison between modelled and measured specific heat flows for different heating rates

DOI 10.3384/ecp218263

The reaction enthalpy $\Delta h_{R,x}$ in equation 11 is the integral under the specific DSC-peak and n describes the number of peaks considered. $\varphi_{reactive}$ describes the reactive mass in the battery cell and is 55.3 % according to Ren et al..

$$\dot{q}_{TR} = \varphi_{reactive} \sum_{x=1}^{n} \Delta h_{R,x} \frac{d\alpha_x}{dt} \tag{11}$$

The relative change $\dot{\gamma}_{rel,out}$ must be defined for the ejected mass (see section 2.2). Therefore, the change of a specific degree of conversion or an average value from several degree of conversions can be used. A relative, dynamic mass loss is thus determined using the equation 12. φ_{loss} represents the relative mass loss of the battery cell.

$$\dot{\gamma}_{rel,out} = \frac{\varphi_{loss}}{n} \sum_{x=1}^{n} \frac{d\alpha_x}{dt} \tag{12}$$

By inserting the conversion degrees into equation 12, not the derivative, the already relative, ejected mass is described, so that equation 5 can be simplified to the following equation. This avoids the porosity state for each Li-Ion-bundle.

$$\rho_s V \phi = m_{s,initial} \cdot \left(1 - \frac{\varphi_{loss}}{n} \sum_{x=1}^{n} \alpha_x\right) \tag{13}$$

From a numerical perspective, it is useful to limit the conversion rate κ_x. The influence on the heat release should be as low as possible. The limitation via a time constant is not expedient in this case, as this is dependent on the temperature (see equation 14). For this reason, the numerical limitation is carried out using the characteristic time τ. For equation 14 a linear DGL was assumed for simplifications. The characteristic time is limited to $\tau_{min} = 10\,ms$. This time has proven to be a good compromise between numerical solvability and influence on the heat release.

$$\begin{gathered} \frac{d\alpha_x}{dt} = -\kappa_x \alpha_x \\ a_x = a_{x,t=0} e^{-\frac{t}{\tau}} = a_{x,t=0} e^{-\kappa_x t} \\ \tau = \frac{1}{\kappa_x} \\ \frac{d\alpha_x}{dt} = \begin{cases} -\kappa_x \alpha_x, & \kappa_x < \frac{1}{\tau_{min}} \\ -\frac{1}{\tau_{min}} \alpha_x, & \kappa_x \geq \frac{1}{\tau_{min}} \end{cases} \end{gathered} \tag{14}$$

2.4 The Electrical Equivalent Circuit Model

A replaceable ECM was chosen for the battery cell. The reasons for choosing an ECM are accurate description of the electrical behaviour and the released heat with low computational effort (Auch et al. 2023; Einhorn et al. 2013) as well as the direct possibility of parameter identification in Modelica (Einhorn et al. 2013; Uddin und Picarelli 2014). The currently implemented ECM are based on the publications by Einhorn et al. and Gerl et al.. They include the Rint (Ri), Thevenin (Th) and Dual Polarization Model (DP) (cf. Figure 4). These models have advantages and disadvantages depending on the accuracy and performance requirements (Einhorn et al. 2013). All ECM's use the same base class, which consists of several components such as a voltage source, resistors and current interrupt devices (CID). The CID decouples the ECM from the electrical pins at high temperatures, e.g. triggered by the TR. The voltage source U_{OCV} describes the open circuit voltage (OCV). The resistors describe several effects. R_0 represents the ohmic internal resistance of the battery cell. The terminal voltage U_T must be calculated differently depending on the model selected (see Figure 4).

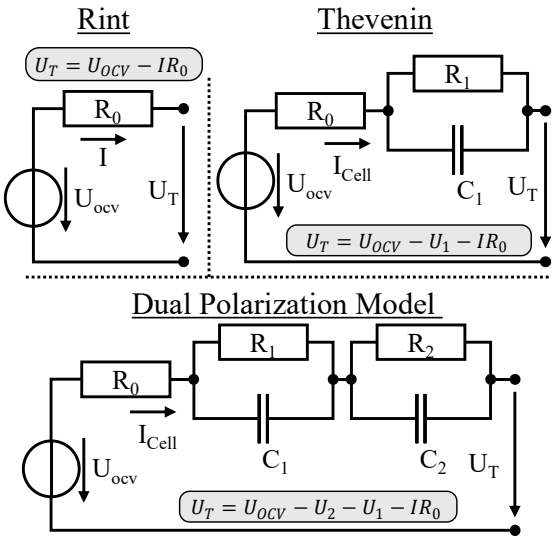


Figure 4. Schematic representation of the ECMs

The electrical data for the OCV, resistances and capacitances can be constant, equation-based or table-based. Whereby the look-up table enables 2D interpolation to consider the SoC and temperature dependence of the data (cf. Einhorn et al. 2013; Uddin und Picarelli 2014). The SoC is determined by Coulomb counting. According to Bernardi et al. 1985 the released heat is described by the irreversible $\dot{Q}_{Loss}$ and reversible $\dot{Q}_{Rev}$ heat. The irreversible part includes ohmic losses, charge transfer, overpotentials and the limitation of mass transfer. The reversible part includes entropy changes and is described by the partial derivative of the OCV with respect to temperature $\frac{\partial U_{OCV}}{\partial T}$ (ϵ small temperature delta) (Auch et al. 2023; Bernardi et al. 1985).

$$\begin{aligned} \dot{Q}_{ECM} &= \dot{Q}_{Loss} + \dot{Q}_{Rev} \\ \dot{Q}_{Loss} &= \mathrm{I}(\mathrm{U_{OCV}} - U_T) \\ \dot{Q}_{Rev} &= IT\frac{\partial \mathrm{U_{OCV}}}{\partial \mathrm{T}} \\ \frac{\partial \mathrm{U_{OCV}}}{\partial \mathrm{T}} &= \frac{\mathrm{U_{OCV}}(T+\epsilon) - \mathrm{U_{OCV}}(T-\epsilon)}{2\epsilon} \end{aligned} \tag{15}$$

2.5 States and Sparse Solvers

The number of differential states changes depending on the discretization. For each thermal and Li-Ion-bundle there is exactly one differential state, the temperature. The porosity can be replaced by an algebraic expression (see section 2.3). Thus, the number of states for the bundle network and housing depends on the discretization. In addition, there is one state each for the tabs/terminals. The Dymola sparse solver (Dassault Systèmes AB 2025) can help speed up the simulation. For example, a TR simulation of a 125-fold discretized battery cell without sparse solver takes ~36 s, with only ~15 s (on a laptop with an 11th Gen Intel Core i7-11800H - 2.30GHz and 16 GB RAM). Thus, the simulation time for this case can be reduced by approx. 42 % by using a sparse solver. A 1000-fold discretization of a battery cell is possible, but the simulation time increases to 2700 s and the DAE system 133408 scalar equations. Higher discretization's were not tested. A comparison with an CFD simulation has not yet been conducted.

By using a global TR model (not discretized one per battery cell) the number of states can be greatly reduced, which can be helpful when simulating entire battery systems. The model shows that Modelica and Modelica tools should continue to improve dealing with high discretization, since the translation time increases exponentially with the number of states, even though the bundles all consist of the same code. This challenge has been recognized and is being addressed in the European OpenSCALING 2023 project.

3 Experiment

The battery cell model described in Section 2.1 can be used to model normal operating behavior with charging and discharging processes as well as dynamic load profiles (Epp et al. 2022; Epp et al. 2023). In addition to simulating the normal operating behavior, it is also possible to carry out safety analyses for the event of damage to TP. These safety considerations can be used for the development of thermally safer battery systems. Furthermore, applications in model-based early detection of the TP are conceivable.

In this section, the *TIL3_AddOn_Battery* is checked for plausibility by comparing it with an experimental propagation test. The test by Schöberl et al. 2024 is used for comparison. This plausibility check concentrates on the thermal propagation. A simulation of the electrical behavior coupled with the thermal behavior is not carried out.

3.1 Propagation Experiment Setup

In the experimental propagation test by Schöberl et al. two battery cell types are examined with regard to their thermal propagation behavior. The battery cells investigated are nickel-manganese-cobalt (NMC)-811 and lithium iron phosphate (LFP). The measurement results for NMC-811 are considered below.

In the experiment, five cells were mounted between two steel plates with two additional thermal insulation layers. The cells are not electrically connected to each other. The battery cells were tensioned with a SoC of 30 %. The battery cells are wrapped in a thin insulating film. Vermiculite plates were used as thermal insulation layers. A heater was placed between one of the insulating layers and the neighboring battery cell to trigger the TR in the first cell. The TR in the first battery cell provided enough heat to trigger the TR in the other cells. By measuring the temperature between the cells, between the first cells it was possible to trace the propagation process

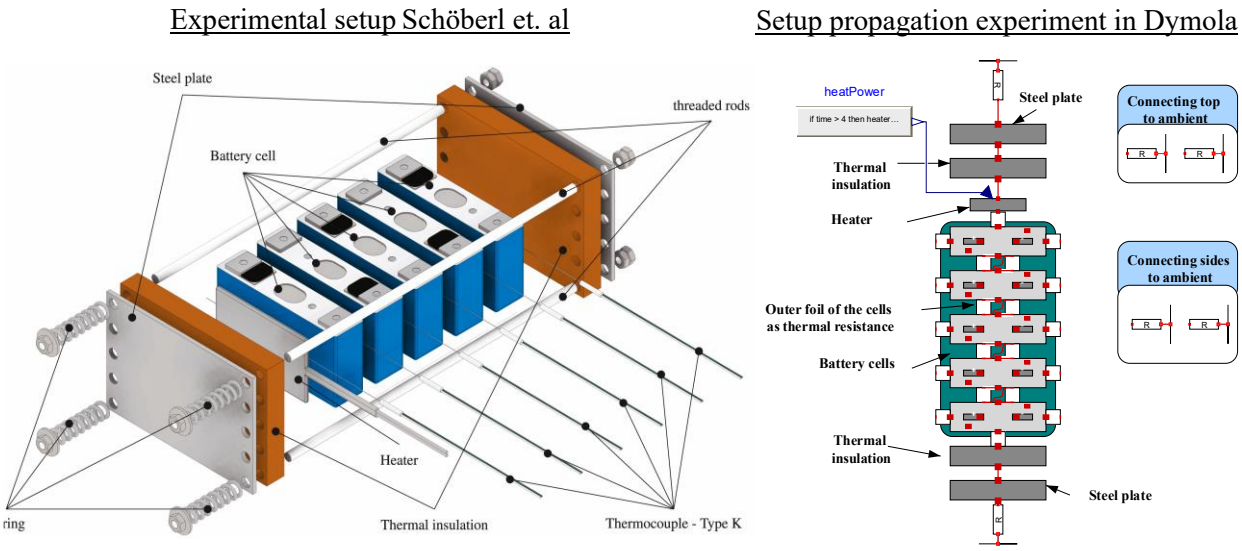


Figure 5. Propagation experiment by Schöberl et al. and the transfer of the setup to Modelica diagram

DOI 10.3384/ecp218263

in the experiment. The structure described is shown in Figure 5.

The steel plates, the thermal insulation layers and the heater are modelled as layers using the thermal bundles (see section 2.2). With the help of the *TILMedia Suite*, a material can be assigned to the layers via the replaceable *SolidType* record. As in the experiment, the dimensions of the layers are adapted to the cell size with an additional allowance, which is estimated using the illustrations from Schöberl et al.. In addition to the thermal insulation layers visible in Figure 5, a further insulation layer is placed on the undersides of the cells. The power for the heater is provided via input. The specified heat flow per area is selected as in the publication. Schöberl et al. specifies that the heater is switched off manually if the TR has occurred in the first cell. In the simulation setup, the heater is switched off at a defined temperature T_{off}.

The model described in section 2.1 is used for battery cells. The foils with which the battery cells are wrapped are modelled in the simulation experiment using thermal resistors.

The TR model (see section 2.3) of the battery cell used for the simulation is parameterized using Ren et al.. The parameters specified in the publication were determined for cell chemistry of NMC-111 and a different battery cell type. The model therefore needs to be adapted. The relative mass loss φ_{loss} and the factor for the relative mass of the cell that can decompose in the TR reaction $\varphi_{reactive}$ are adjusted. The estimation of the reactive mass $\varphi_{reactive}$ was based on the temperature levels available from the measurements. The mass loss is described in Schöberl et al. for all five cells with an average of $\varphi_{loss} =$ 45,8 %. For the individual cells, the relative mass loss is stored in the TR model.

The heat ports of the battery cells and the heater, which are not visibly connected to another heat port in Figure 5 are connected to the environment via a convection resistor. Table 1 shows the details of the individual models with their respective properties for the simulation.

Table 1. Summary of the relevant values for the propagation simulation

Symbol	Description	Value	Literature
General			
T_{Start}	Initialization & ambient temperature	20 °C	Assumption
T_{off}	Heater switch-off temperature	375 °C	Assumption
$\dot{q}_{HT}$	Heat output per heating surface	6.2 $\frac{W}{cm^2}$	Schöberl et al. 2024
α_{Air}	Heat transfer coefficient for convection resistance	12 $\frac{W}{m^2K}$	Assumption
Steel Plate (TILMedia-Stainless Steel)			
$n_{x,St}$,$n_{y,St}$,$n_{z,St}$	Discretization in x-, y- and z-direction	1, 1, 1	-
δ_{St}	Thickness	5 mm	Assumption
$c_{p,St}$	Specific heat capacity	450 $\frac{J}{kgK}$	TIL-Media
λ_{St}	Thermal conductivity	14.6 $\frac{W}{mK}$	TIL-Media
ρ_{St}	Density	7900 $\frac{kg}{m^3}$	TIL-Media
Thermal insulation layer (Vermiculite)			
$n_{x,TI}$,$n_{y,TI}$,$n_{z,TI}$	Discretization in x-, y- and z-direction	1,1,1	-
δ_{TI}	Thickness	25 mm	Schöberl et al. 2024
$c_{p,TI}$	Specific heat capacity	960 $\frac{J}{kgK}$	*Datasheet
λ_{TI}	Thermal conductivity	0.071 $\frac{W}{mK}$	*Datasheet
ρ_{TI}	Density	176 $\frac{kg}{m^3}$	*Datasheet
Heater (TILMedia-Aluminium)			
$n_{x,Ht}$,$n_{y,Ht}$,$n_{z,Ht}$	Discretization in x-, y- and z-direction	1,1,1	-
δ_{Ht}	Thickness	0.5 mm	Assumption
$c_{p,Ht}$	Specific heat capacity	920 $\frac{J}{kgK}$	TILMedia
λ_{Ht}	Thermal conductivity	215 $\frac{W}{mK}$	TILMedia
ρ_{Ht}	Density	2700 $\frac{kg}{m^3}$	TILMedia
Battery Cell NMC-811 / 173 x 42 x 85 mm (Housing material: TILMedia-Aluminium)			
$n_{x,Bz}$,$n_{y,Bz}$,$n_{z,Bz}$	Discretization in x-, y- and z-direction	3,15,1	-
m_{Bz}	Average mass	1423.9 g	Schöberl et al. 2024
$c_{p,Bz}$	Specific heat capacity	800 $\frac{J}{kgK}$	Assumption
$\lambda_{Bz,\parallel}$	Thermal conductivity parallel to the active material layers	23.1 $\frac{W}{mK}$	Hoelle et al. 2023
$\lambda_{Bz,\perp}$	Thermal conductivity across the active material layers	1.034 $\frac{W}{mK}$	Hoelle et al. 2023
$\lambda_{Bz,Foil}$	Thermal conductivity of outer foil	0.25 $\frac{W}{mK}$	Assumption
$\delta_{Bz,Foil}$	Thickness outer foil	0.05 mm	Assumption
$\varphi_{reactive}$	Reactive mass TR model	38 %	Assumption
φ_{loss}	Average mass loss	45.8 %	Schöberl et al. 2024

*Datasheet: Vermiculite Association

3.2 Simulation Results

For the plausibility check of the models, the temperatures between the battery cells are examined as in the experiment. The temperatures at the respective heat ports of the cell models can be used for this purpose. The simulation results compared to the measurement data are shown in Figure 6. In the simulation, all five cells enter the TR, and a TP could be shown.

The simulation and the experiment show a good agreement for the initial temperature rise of the TR for the first three battery cells. This shows that heat conduction is well represented for the first three battery cells and that TR heat is released at a temperature comparable to the experiment. However, some deviating temperature curves are recognizable. A comparison between T1 measurements and T1 simulation shows a slower heating phase for the simulation. The reasons for this may lie in the lack of information on the internal cell structure or in

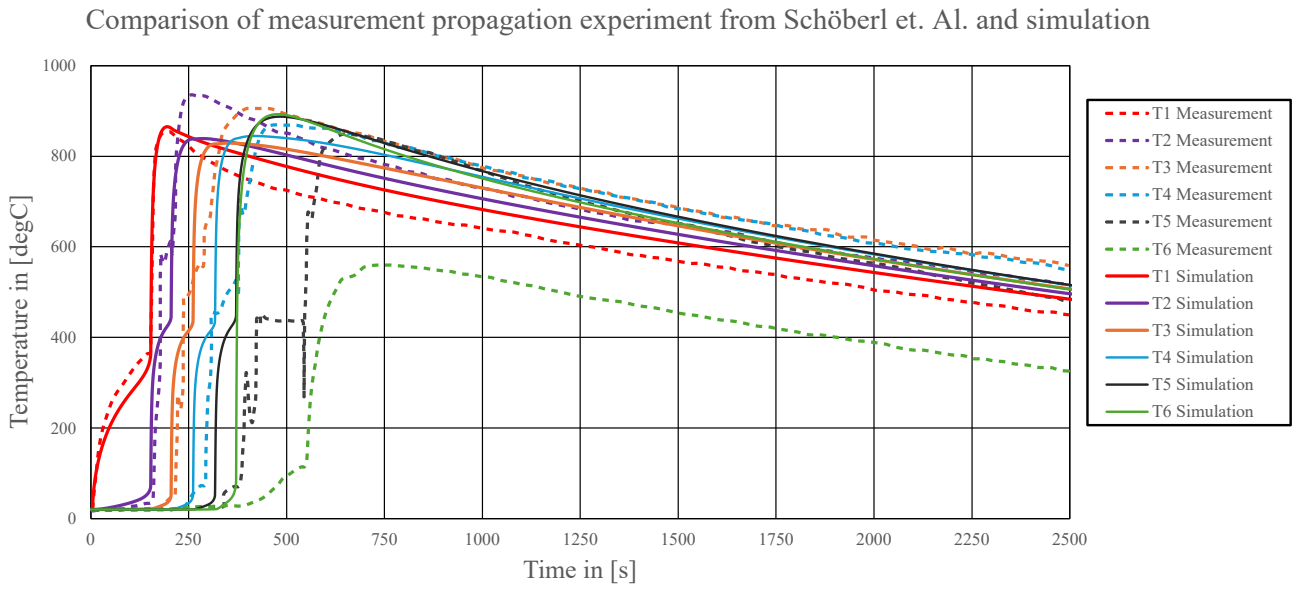


Figure 6. Comparison measurement (Schöberl et al. 2024) and simulation (T1-Heater-Cell 1; T2- Cell 1-Cell 2; T3- Cell 2-Cell 3; T4- Cell 3-Cell 4; T5- Cell 4-Cell 5; T6- Cell 5-Insulation)

deviating thermal properties due to assumptions made during modelling. Furthermore, the temperature for the measurement after the TR in cell 1 drops faster than in the simulation. In Schöberl et al. the mass loss could only be estimated retrospectively for the first cells. The mass loss of the first three cells was averaged due to irreversible fusion. If cell 1 has emitted more mass in the experiment than assumed, this can lead to the recognizable deviating cooling behavior. In the cooling phase of the other cells after the TP, similar temperature curves are observed between simulation and measurement.

The behavior of cell 5 cannot be matched with the simulation. The temperature rise in the TR stagnates here. One reason could be the decomposition reactions which are subject to a certain scatter (Lenz et al. 2023). Another reason could be the loss of mechanical integrity due to the mass loss and a resulting thermal contact resistance between the battery cells where the temperature is measured (Schöberl et al. 2024). This could also be the reason for the different temperature curve between simulation and measurement in cell 6. In addition, the temperature of cell 6 is significantly influenced by cell 5.

The simulation is based on several assumptions (cf. Table 1). The mentioned deviations in temperature curves can arise due to deviating thermal properties of the assumed material parameters. To make the simulation more reliable, the materials used should be thermally characterized in advance for a battery system to be investigated. In conclusion, it can be said that a TP can be simulated with the battery library and that these simulations can be used to carry out safety assessments for battery systems.

4 Conclusion

This publication presents a discretized 3D battery cell model in Modelica. The focus of the modelling is the physical simulation of the TR based on the Arrhenius equations and the modelling of the anisotropic thermal conductivity. This allows to consider relevant heat conduction paths in the battery cell using 3D discretization. The thermo-electric behavior is modelled by a coupled ECM. The model was developed for the user-friendly and fast simulation of battery cells and systems. To the authors' knowledge, the 3-dimensional thermal calculation with a coupled physical TR model is a novel approach in Modelica for the simulation of battery systems. The comparison with an experiment from the literature has shown good agreement with the simulation. The calculation of the TP was possible within approx. 83 s (experimental time approx. 47 min) and thus represents only a fraction of the required simulation time compared to complex 3D CFD simulations.

Future enhancements of battery cell model are gas generation, outgassing and particle ejection, which have a significant influence on the TR. As well as further components and exemplary models for simulation of entire battery systems.

Acknowledgements

This work has been supported by German Federal Ministry of Education and Research (BMBF, Grant No.: 01IS24045A).

References

Al-Khamis, Kh.M.; Al-Othman, Z. A.; Mahfouz, R. M. (2010): Kinetic Studies of the Non-Isothermal Decomposition of Unirradiated and Γ-Irradiated Gallium Acetylacetonate. In: *Progress in Reaction Kinetics and Mechanism* 35 (2), S. 187–207. DOI: 10.3184/146867810X12686717520239.

Auch, Marcus; Kuthada, Timo; Giese, Sascha; Wagner, Andreas (2023): Influence of Lithium-Ion-Battery Equivalent Circuit Model Parameter Dependencies and Architectures on the Predicted Heat Generation in Real-Life Drive Cycles. In: *Batteries* 9 (5), S. 274. DOI: 10.3390/batteries9050274.

Bernardi, D.; Pawlikowski, E.; Newman, J. (1985): A General Energy Balance for Battery Systems. In: *J. Electrochem. Soc.* 132 (1), S. 5–12. DOI: 10.1149/1.2113792.

Coman, Paul T.; Darcy, Eric C.; Veje, Christian T.; White, Ralph E. (2017): Modelling Li-Ion Cell Thermal Runaway Triggered by an Internal Short Circuit Device Using an Efficiency Factor and Arrhenius Formulations. In: *J. Electrochem. Soc.* 164 (4), A587-A593. DOI: 10.1149/2.0341704jes.

Dassault Systèmes AB (Hg.) (2025): Dymola - Full User Manual (2025x). All Chapters from User Manual 1A – User Manual 2C Included). Ideon Gateway, Scheelevägen 27 – Floor 9, SE-223 63 Lund, Sweden.

Einhorn, Markus; Conte, Fiorentino Valerio; Kral, Christian; Fleig, Jürgen (2013): Comparison, Selection, and Parameterization of Electrical Battery Models for Automotive Applications. In: *IEEE Trans. Power Electron.* 28 (3), S. 1429–1437. DOI: 10.1109/TPEL.2012.2210564.

Epp, Alexander; Rai, Sunny; van Ginneken, Finn; Varchmin, Andreas; Koehler, Juergen; Sauer, Dirk Uwe (2023): Simulative Investigation of Optimal Multiparameterized Cooling Plate Topologies for Different Battery System Configurations. In: *Energy Tech* 11 (9), Artikel 2300405. DOI: 10.1002/ente.202300405.

Epp, Alexander; Wendland, Robert; Behrendt, Jan; Gerlach, Robin; Sauer, Dirk Uwe (2022): Holistic battery system design optimization for electric vehicles using a multiphysically coupled lithium-ion battery design tool. In: *Journal of Energy Storage* 52, S. 104854. DOI: 10.1016/j.est.2022.104854.

Gerl, Johannes; Janczyk, Leonard; Krüger, Imke; Modrow, Nils (2014): A Modelica Based Lithium Ion Battery Model. In: Proceedings of the 10th International Modelica Conference, March 10-12, 2014, Lund, Sweden. the 10th International Modelica Conference, March 10-12, 2014, Lund, Sweden, March 10-12, 2014: Linköping University Electronic Press (Linköping Electronic Conference Proceedings), S. 335–341.

Groß, Christian; Golubkov, Andrej W. (2021): A Modelica library for Thermal-Runaway Propagation in Lithium-Ion Batteries. In: Proceedings of 14th Modelica Conference 2021, Linköping, Sweden, September 20-24, 2021. 14th Modelica Conference 2021, 2021-09-20: Linköping University Electronic Press (Linköping Electronic Conference Proceedings), S. 215–224.

Hatchard, T. D.; MacNeil, D. D.; Basu, A.; Dahn, J. R. (2001): Thermal Model of Cylindrical and Prismatic Lithium-Ion Cells. In: *J. Electrochem. Soc.* 148 (7), A755. DOI: 10.1149/1.1377592.

Hoelle, S.; Zimmermann, S.; Hinrichsen, O. (2023): 3D Thermal Simulation of Thermal Runaway Propagation in Lithium-Ion Battery Cell Stack: Review and Comparison of Modeling Approaches. In: *J. Electrochem. Soc.* 170 (6), S. 60516. DOI: 10.1149/1945-7111/acd966.

Kim, Jinyong; Mallarapu, Anudeep; Finegan, Donal P.; Santhanagopalan, Shriram (2021): Modeling cell venting and gas-phase reactions in 18650 lithium ion batteries during thermal runaway. In: *Journal of Power Sources* 489, S. 229496. DOI: 10.1016/j.jpowsour.2021.229496.

Kriston, Akos; Adanouj, Ibtissam; Ruiz, Vanesa; Pfrang, Andreas (2019): Quantification and simulation of thermal decomposition reactions of Li-ion battery materials by simultaneous thermal analysis coupled with gas analysis. In: *Journal of Power Sources* 435, S. 226774. DOI: 10.1016/j.jpowsour.2019.226774.

Lenz, Carolin (2024): Analyse des Reaktionsverhaltens beim thermischen Durchgehen einer Batteriezelle. Universitätsbibliothek Braunschweig.

Lenz, Carolin; Hennig, Jakob; Tegethoff, Wilhelm; Schweiger, Hans-Georg; Koehler, Juergen (2023): Analysis of the Interaction and Variability of Thermal Decomposition Reactions of a Li-Ion Battery Cell. In: *J. Electrochem. Soc.* DOI: 10.1149/1945-7111/acd964.

MacNeil, D. D.; Dahn, J. R. (2001): Test of Reaction Kinetics Using Both Differential Scanning and Accelerating Rate Calorimetries As Applied to the Reaction of Li x CoO 2 in Non-aqueous Electrolyte. In: *J. Phys. Chem. A* 105 (18), S. 4430–4439. DOI: 10.1021/jp001187j.

OpenSCALING (2023). Online verfügbar unter https://openscaling.org/, zuletzt geprüft am 19.03.2025.

Ren, Dongsheng; Liu, Xiang; Feng, Xuning; Lu, Languang; Ouyang, Minggao; Li, Jianqiu; He, Xiangming (2018): Model-based thermal runaway prediction of lithium-ion batteries from kinetics analysis of cell components. In: *Applied Energy* 228, S. 633–644. DOI: 10.1016/j.apenergy.2018.06.126.

Schöberl, Jan; Ank, Manuel; Schreiber, Markus; Wassiliadis, Nikolaos; Lienkamp, Markus (2024): Thermal runaway propagation in automotive lithium-ion batteries with NMC-811 and LFP cathodes: Safety requirements and impact on system integration. In: *eTransportation* 19, S. 100305. DOI: 10.1016/j.etran.2023.100305.

TIL Suite. Version 2024.01: TLK-Thermo GmbH. Online verfügbar unter https://www.tlk-thermo.com/en/software/til-suite, zuletzt geprüft am 14.03.2025.

TILMedia Suite. Version 2024.01: TLK-Thermo GmbH. Online verfügbar unter https://www.tlk-thermo.com/en/software/tilmedia-suite, zuletzt geprüft am 14.03.2025.

Uddin, Kotub; Picarelli, Alessandro (2014): Phenomenological Li ion battery modelling in Dymola. In: Proceedings of the 10th International Modelica Conference, March 10-12, 2014, Lund, Sweden. the 10th International Modelica Conference, March 10-12, 2014, Lund, Sweden, March 10-12, 2014: Linköping University Electronic Press (Linköping Electronic Conference Proceedings), S. 327–334.

Walker, William Q.; Bayles, Gary A.; Johnson, Kenneth L.; Brown, Ryan P.; Petrushenko, David; Hughes, Peter J. et al. (2022): Evaluation of Large-Format Lithium-Ion Cell Thermal Runaway Response Triggered by Nail Penetration using Novel Fractional Thermal Runaway Calorimetry and Gas Collection Methodology. In: *J. Electrochem. Soc.* 169 (6), S. 60535. DOI: 10.1149/1945-7111/ac7897.

Wang, Yu; Ren, Dongsheng; Feng, Xuning; Wang, Li; Ouyang, Minggao (2022): Thermal runaway modeling of large format high-nickel/silicon-graphite lithium-ion batteries based on reaction sequence and kinetics. In: *Applied Energy* 306, S. 117943. DOI: 10.1016/j.apenergy.2021.117943.

Wendland, Robert; Padberg, Pascal; Tegethoff, Wilhelm; Koehler, Juergen (2018): Dynamic Optimization of Thermal Management and Cooling Plate Design for Battery Systems. 27th Aachen Colloquium Automobile and Engine Technology. Aachen.

Development and Experimental Validation of an Unglazed Photovoltaic-Thermal Collector Modelica Model that only needs Datasheet Parameters

Lone Meertens[1*] Jelger Jansen[1] Lieve Helsen[1,2]

[1]Faculty of Engineering Science, Department of Mechanical Engineering, KU Leuven, Belgium.
⋆Corresponding author : lone.meertens@kuleuven.be
[2]EnergyVille, Thor Park 8310, 3600 Genk, Belgium.

Abstract

This work introduces the first validated, user-friendly, and accurate open-source photovoltaic-thermal (PVT) collector model in Modelica, tailored for system-level simulation and optimization. Current state-of-the-art PVT collector Modelica models are largely limited to oversimplified, steady-state representations that fail to capture the dynamic thermal behavior inherent to real PVT systems. A comprehensive Modelica model is developed based on the ISO 9806 standard (test method for the quasi-dynamic thermal performance of solar thermal collectors), coupled with an electrical system model through an internal heat transfer coefficient. The model calibration relies exclusively on manufacturer datasheet parameters, thereby eliminating the need for parameter estimation from measurement data. The model is validated using experimental results from an unglazed PVT collector, demonstrating strong agreement for various (weather) conditions. The findings highlight that, while steady-state models may suffice for conventional solar thermal collectors (STCs), accurate PVT modeling necessitates a dynamic approach, particularly for the thermal aspects. The electrical output of the PVT collector is less sensitive to transient effects. In addition to the model formulation and validation, this work presents a user-friendly automated calibration method based on manufacturer data, and critically addresses the limitations and potential trade-offs of using exclusively datasheet-derived parameters, thereby providing a transparent tool for PVT system simulation, design, and optimization within the open-source IDEAS library.

Keywords: Photovoltaic-thermal (PVT) solar collector, Modelica, PVT model validation, PVT model calibration, manufacturer datasheets

1 Introduction

Physics-based system models are key for addressing the challenges of climate change and the transition towards decarbonized energy systems. These models play a key role in the effective deployment of renewable energy sources (RES) such as photovoltaic (PV), solar thermal collectors (STC), and photovoltaic-thermal (PVT) systems, enabling optimal system design and control. PVT collectors, also known as hybrid solar collectors, simultaneously convert solar radiation into electricity and heat. A key distinction exists between glazed and unglazed PVT collectors: glazed collectors often prioritize thermal efficiency, while unglazed collectors, due to their lower operating temperatures, can enhance electrical efficiency. This paper focuses on unglazed (so-called WISC[1]) PVT collectors, which consist of a PV module coupled with a heat exchanger and may include rear insulation to reduce heat loss. Variants without insulation perform particularly well under sub-ambient conditions, making them suitable as heat sources for heat pump systems. The unglazed collectors are typically classified according to their rear insulation and heat transfer characteristics (Lämmle et al. 2020). The favorable market conditions for unglazed PVT designs further demonstrate their relevance in various urban and industrial energy contexts (Kramer, Amrizal, et al. 2020). Modeling approaches in the PVT field range from highly accurate, detailed, component-level models to simplified methods designed for integrated system performance assessment within a broader energy context. Although Modelica is a powerful modeling language for developing interdisciplinary simulation models, no validated open-source Modelica PVT model currently exists that captures the full dynamic thermal behavior while accounting for the thermal-electrical coupling. This paper addresses this gap by presenting a novel Modelica PVT collector model, validated using experimental data from unglazed PVT collector. Moreover, the PVT model relies solely on publicly available manufacturer data, avoiding complex calibration procedures for which measurement data are needed.

2 Background

The IEA SHC Task 60 *"Application of PVT Systems"* (2018–2020) has played a central role in evaluating PVT technologies by focusing on system operation, collector testing, system-level simulation, and global performance assessment (IEA SHC 2018). One of its key objectives

[1]Wind and/or infrared sensitive collectors

was to identify scenarios in which PVT systems provide added value compared to separate PV and solar thermal installations. To facilitate the integration of PVT systems into simulation environments and ensure broader adoption, Task 60 stresses the need for standardized input parameters that guarantee the "processability" of technical data for tools like TRNSYS, Polysun, Modelica, and SCenoCalc (Kramer, Amrizal, et al. 2020). This standardization is crucial due to potential variations in testing procedures among manufacturers (market fragmentation), making consistent component-level characterization essential. Therefore, Task 60 recommended the application of international standards, specifically ISO 9806 (ISO 9806 2013) for the thermal aspect and IEC 60904 for the PV component. To incentivize the use of these standards, established certification frameworks such as the Solar Keymark (ESTIF/SKN n.d.) utilize and promote them, with targeted extensions to address the hybrid interactions between electrical and thermal components. This includes specifying the operational mode of electricity generation, such as maximum power point (MPP) tracking or open-circuit (OC), as required by ISO 9806 (2013) which has been identified by Hofmann et al. (2010) as highly influential on the thermal output.

Existing PVT collector models can generally be categorized into (1) those requiring detailed design data and parameter estimation, and/or (2) simplified models based on datasheet parameters. Stegmann et al. (2011) developed a datasheet-based simulation model for unglazed, liquid-cooled PVT collectors, relying solely on standard test procedures and OC-tested datasheet parameters within the TRNSYS environment. In contrast, Lämmle (2018) proposed both a collector-level model in Modelica capable of capturing the detailed optical, electrical, and thermal energy flows using detailed manufacturer data, and a system-level model in TRNSYS that assesses overall performance based on empirical efficiency curves. Other approaches focus on parameter identification of ISO 9806 parameters using measurement data: Jonas et al. (2019) presented a detailed implementation and validation procedure within TRNSYS, while Veynandt, Klanatsky, et al. (2023) extended this work using data-driven grey-box models in MATLAB. Additionally, Kang (2021) developed a Modelica-based PVT model for district heating applications, but they didn't provide any implementation details. These existing models present several limitations: some are not implemented in Modelica, others lack sufficient implementation details, and many rely on parameter estimation of ISO 9806 parameters, requiring extensive measurement data. The work presented here offers an alternative by providing a Modelica-based PVT collector model that combines a high-fidelity physical representation with the simplicity of datasheet-only inputs.

The electrical operation of a PVT collector differs from standard PV IEEE rating schemes due to the influence of heat transfer fluid (HTF) flow and the resulting temperature distribution across the PV cells (Herrando, Markides, and Hellgardt 2014; Lämmle 2018; Brottier 2019; Jonas et al. 2019). Similarly, the electricity generation mode (MPP/OC) impacts the thermal behavior of the PVT collector (Hofmann et al. 2010). The existing equations describing the thermal and electrical performance, used as the starting point in this paper, are presented in Section 2.1 and 2.2, respectively.

2.1 Thermal modeling

The thermal part of a PVT collector is similar to a STC. Similarly, a good efficiency requires both effective solar absorption and efficient heat transfer to the HTF (Zondag 2008). Different mathematical models are used to describe the thermal performance of liquid heating collectors, depending on the collector type (glazed or unglazed) and the test method (steady-state or quasi-dynamic). ISO 9806 defines a steady-state model for glazed liquid-heating solar collectors, widely adopted in both testing and simulation. Its useful thermal power output is calculated by Equation 1

$$\frac{\dot{Q}_{\text{th}}}{A_G} = \eta_{0,\text{hem}}\, G - a_1\,(\theta_m - \theta_a) - a_2\,(\theta_m - \theta_a)^2, \qquad (1)$$

where $\dot{Q}_{\text{th}}$ is the useful thermal power output (W), A_G is the collector area (m^2), $\eta_{0,\text{hem}}$ is the zero-loss (optical) efficiency for hemispherical irradiance (–), G is the global irradiance on the collector plane (W m^{-2}), θ_m and θ_a are the mean HTF and ambient air temperatures (K or °C). a_1 and a_2 are the linear and quadratic heat-loss coefficients ($\text{W m}^{-2}\,\text{K}^{-1}$ and $\text{W m}^{-2}\,\text{K}^{-2}$, respectively).

For unglazed collectors under steady-state conditions, wind-speed and long-wave effects must be included. One convenient formulation replaces G by a net solar irradiance G'', which integrates the long-wave irradiance:

$$\frac{\dot{Q}_{\text{th}}}{A_G} = G''\Big[\,\eta_{0,\text{hem}}\,(1-b_u)\;-\;(b_1 + b_2\,u)\,\frac{\theta_m - \theta_a}{G''}\Big], \qquad (2)$$

$$\text{with}\quad G'' \;=\; G \;+\; \frac{\varepsilon}{\alpha}\left(E_L - \sigma\,\theta_a^4\right) \qquad (3)$$

Here, b_u is the collector efficiency coefficient (wind dependence) (s m^{-1}), b_1 is the linear heat-loss coefficient ($\text{W m}^{-2}\,\text{K}^{-1}$), b_2 is the wind-speed dependence of the heat-loss coefficient ($\text{W s m}^{-3}\,\text{K}^{-1}$), u is the wind speed, E_L is the long-wave irradiance (W m^{-2}) , and σ is the Stefan–Boltzmann constant.

To capture both unglazed/glazed types and transient effects, the quasi-dynamic model adds a thermal capacitance, dependence of direct (G_b) and diffuse (G_d) radiation via incidence-angle modifiers (K_b, K_d), and explicit wind/infrared terms:

$$\begin{aligned}\frac{\dot{Q}_{th}}{A_G} &= \eta_{0,b} K_b(\theta_L,\theta_T)\,G_b + \eta_{0,b} K_d\,G_d - c_1\,(\theta_m - \theta_a)\\ &\quad - c_2\,(\theta_m - \theta_a)^2 - c_3\,u\,(\theta_m - \theta_a)\\ &\quad + c_4\left(E_L - \sigma\,\theta_a^4\right) - c_5\,\frac{d\theta_m}{dt} - c_6\,u\,G. \end{aligned} \qquad (4)$$

The quasi-dynamic model extends the steady-state form by including coefficients c_1 and c_2, linear and quadratic

DOI 10.3384/ecp218273

heat-loss terms analogous to a_1, a_2 in Equation 1, with units $\mathrm{W\,m^{-2}\,K^{-1}}$ and $\mathrm{W\,m^{-2}\,K^{-2}}$; c_3 and c_6, representing wind-speed dependence of heat losses and zero-loss efficiency, with units $\mathrm{J\,m^{-3}\,K^{-1}}$ and $\mathrm{s\,m^{-1}}$; c_4, the sky temperature dependence of the heat-loss coefficient (-); and c_5, the effective thermal capacitance, with unit $\mathrm{J\,m^{-2}\,K^{-1}}$.

Equations 1–4 were originally developed for STCs, but are commonly applied to describe the thermal behavior of PVT systems, as no dedicated PVT testing standard currently exists. Depending on the test method (unglazed) steady-state or quasi-dynamic, manufacturer datasheets list all coefficients needed to solve the corresponding equation that gives the collector's thermal output under specific operational and ambient conditions. Due to the strong coupling between thermal and electrical performance in PVT systems, the thermal coefficients should ideally be determined under MPP conditions. If testing is performed under OC conditions, the solar irradiance must be corrected to account for the electrical power that would be extracted under MPP operation, as outlined by Lämmle (2018).

2.2 Electrical modeling

In PVT collectors, the electrical performance is highly sensitive to the temperature of the PV cells, which is influenced by the thermal coupling between the PV layer and the HTF in the STC. A low thermal resistance between the PV cells and the HTF is essential to avoid excessive PV cell temperatures, which degrade both electrical and thermal efficiency (Zondag 2008). This section focuses on two key aspects: the calculation of electrical yield using performance ratios, and the modeling of the PV(T) cell temperature, which plays a central role in determining the electrical output.

2.2.1 Electrical yield

The general form of the electrical power output from a PVT collector is given by Equation 5

$$\frac{P_{\mathrm{el}}}{A_G} = \eta_{\mathrm{PV}} \cdot PR_{\mathrm{tot}} \cdot G, \tag{5}$$

where η_{PV} is the electrical efficiency at nominal conditions (–), PR_{tot} is the overall instantaneous performance ratio (–), and G is the global solar radiation on the PVT plane (W).

The typical loss mechanisms affecting the electrical performance of PV and PVT systems are detailed and summarized in Kramer, Amrizal, et al. (2020), with their corresponding performance ratios PR_{IAM}, PR_G, PR_T, PR_C and PR_I denoting incidence-angle modifier losses, irradiance-related losses, PV cell temperature losses, ohmic (cable) losses and inverter efficiency losses, respectively. In this work, only the temperature-related performance ratio PR_T is explicitly modeled. All other losses are grouped into a single lumped loss factor, as discussed in Section 3.1.3.

The temperature dependence of the electrical efficiency is calculated following the approach by Skoplaki and Palyvos (2009) as formulated in Equation 6

$$PR_T = 1 - \beta \cdot (T_{\mathrm{cell}} - T_{\mathrm{nom}}), \tag{6}$$

where β is the temperature coefficient of power for the PV cells ($\%\,\mathrm{K^{-1}}$), T_{cell} is the PV cell temperature, and T_{nom} is the nominal cell temperature under standard test conditions.

2.2.2 PV cell temperature

An essential step in assessing PV efficiency and electrical yield is determining the PV cell temperature, since higher temperatures reduce performance (see Equation 6). In PV modules, cell temperature follows the module's steady-state temperature (Lämmle 2018), whereas in PVT collectors it is governed by the HTF temperature.

Solar radiation absorbed by a PV cell is partly converted to electricity; the remainder is dissipated to the ambient, influenced by wind speed, ambient temperature, and irradiance. Mounting-type-dependent cell-temperature models are reviewed in Maier et al. (2023).

A simple PVT estimate equates T_{PV} to the fluid outlet temperature $T_{\mathrm{fluid,out}}$ (Brottier 2019), as implemented in tools such as Polysun. A more accurate method extends the Hottel–Whillier–Bliss equation with a PV–fluid heat-transfer coefficient U_{AbsFluid} (Florschuetz 1976; Zondag 2008; Jonas et al. 2019).

Based on Lämmle (2018) and Jonas et al. (2019), the model couples the ISO 9806 quasi-dynamic thermal collector (Equation 4) with a two-node PV performance formulation (see Section 3.1.1), linking the PV cell and the HTF via U_{AbsFluid} ($\mathrm{W\,m^{-2}\,K^{-1}}$) to yield the cell-temperature relation in Equation 7

$$T_{\mathrm{cell}} = T_m + \frac{\dot{q}_{\mathrm{th}}}{U_{\mathrm{AbsFluid}}}, \tag{7}$$

where $\dot{q}_{\mathrm{th}}$ is the specific thermal power output of the collector ($\mathrm{W\,m^{-2}}$), and T_m is the mean HTF temperature (K), calculated as the average of the HTF inlet and outlet temperatures, T_{in} and T_{out} respectively.

In principle, U_{AbsFluid} can be calculated from the series combination of the absorber-to-pipe conductive resistance, characterized by the conductive heat transfer coefficient U_{AbsPipe}, and the pipe-to-fluid convective resistance, characterized by the convective heat transfer coefficient $U_{\mathrm{PipeFluid}}$, based on the collector geometry, material properties, and operational conditions (e.g., fluid flow rate) using analytical models or finite-element (FEM) simulations (Kramer, Amrizal, et al. 2020). However, for most commercial PVT collectors, the absorber-specific parameters required for such detailed modeling are not readily available, as only standardized test reports or manufacturer data are typically accessible. In such cases, an approximate approach needs to be applied. Stegmann et al. (2011) developed an approach for unglazed PVT collectors, using thermal parameters extracted from OC tests. Lämmle (2018) similarly proposed a procedure for glazed PVT collectors that derives U_{AbsFluid} from standard MPP

test data using a makeshift procedure. In this work, the glazed and unglazed parameter estimation methods are integrated in the Modelica model, such that U_{AbsFluid} can be derived solely from manufacturer datasheet parameters for MPP tested collectors, including wind-speed dependence. Details of this method are given in Section 3.1.2.

2.3 Novelty of the work

Existing PVT modeling approaches (Lämmle 2018; Jonas et al. 2019) provide robust frameworks for simulating PVT systems. However, these models typically rely on detailed parameter identification procedures or proprietary datasets, often requiring experimental calibration, finite element modeling, or surface temperature measurements. While such methods can yield high accuracy, they limit accessibility and replicability for broader research and system simulations, hinder transparent product comparisons, and ultimately restrict the market uptake of PVT technology. Furthermore, existing simplified (non-validated) Modelica models often fail to capture the full dynamic thermal behavior of PVT systems and do not adequately integrate the coupling between thermal and electrical components. For instance, Stegmann et al. (2011) proposed a simplified model based on datasheet parameters, but this model is not implemented in Modelica and is strictly limited to unglazed PVT collectors, as it relies on the steady-state ISO 9806 formulation for unglazed collectors (see Equation 2) rather than the more general quasi-dynamic approach.

In summary, the current literature regarding PVT modeling has two main research gaps: (1) the absence of a PVT model that leverages only publicly available datasheet parameters thereby eliminating the need for unavailable or proprietary measurement data, and (2) the absence of a comprehensive Modelica model that accurately represents the full dynamic thermal behavior of PVT systems while incorporating thermal-electrical coupling.

In this work, a novel Modelica-based PVT collector model is introduced and validated with experimental data from unglazed PVT collectors. Unlike previous approaches, the proposed model relies solely on datasheet parameters, thereby eliminating the need for complex calibration procedures. Moreover, the model's performance is critically assessed, highlighting its limitations and comparing its accuracy against methods based on detailed parameter estimation.

3 Model development

In this section, the methodology used to develop, implement, and validate the novel PVT collector model is presented. In Section 3.1, the underlying modeling framework is detailed while in Section 3.2, the metrics employed to assess model accuracy are discussed.

3.1 PVT collector model

The PVT collector model consists of two coupled submodels—thermal and electrical—which are detailed in Sections 3.1.1–3.1.3. To account for the changing HTF temperature along the lenght of the collector, the model is discretized into n_{Seg} segments along the flow path. Each segment is solved in a coupled thermal–electrical loop. For clarity, the core segment index is omitted in the equations presented in the following subsections.

3.1.1 Thermal performance

To model the thermal behavior of PVT collectors, a two-node, one-capacitance network is adopted (schematically presented in Figure 1), extending the ISO 9806 solar-thermal formulation. An internal heat transfer coefficient U_{AbsFluid} couples the mean HTF temperature node T_m and the absorber (PV cell) temperature T_{cell}. All physical heat capacities (fluid, absorber, frame, insulation) are lumped into an effective thermal capacity c_{eff}, which is assigned to the mean HTF temperature T_m (Fischer and Müller-Steinhagen 2009). Physically, this means that the thermal inertia of the entire collector system is at one uniform temperature T_m.

The T_m-node energy balance is formulated by Equation 8

$$c_{\text{eff}} \frac{dT_m}{dt} = \dot{q}_{\text{rad}} - \dot{q}_{\text{loss}} - \dot{q}_{\text{th}}, \tag{8}$$

where $\dot{q}_{\text{rad}}$ are radiative gains ($\mathrm{W\,m^{-2}}$), $\dot{q}_{\text{loss}}$ conduction/-convection losses ($\mathrm{W\,m^{-2}}$), and $\dot{q}_{\text{th}}$ the useful thermal output per area ($\mathrm{W\,m^{-2}}$). Rearranging Equation 8 and substituting the radiative gain and loss terms from Figure 1 yields: (1) the steady-state ISO 9806 model (Equation 1), by setting $c_{\text{eff}} = 0$ and including only the blue terms. (2) the quasi-dynamic ISO 9806 model (Equation 4), by retaining c_{eff} and adding the red wind-speed and sky-temperature terms.

The ISO 9806 thermal network model (dashed lines in Figure 1) is available in the open-source IBPSA Modelica library for simulating STC behavior (Wetter et al. 2019), where it is based on a simplified version of the quasi-dynamic ISO 9806 model (Equation 4) with $c_3 = c_4 = c_6 = 0$, thus neglecting wind and sky-temperature dependencies. This existing implementation serves as the starting point for the PVT collector model developed in this work. To extend its validity to unglazed collectors, wind-speed and long-wave loss terms are in-

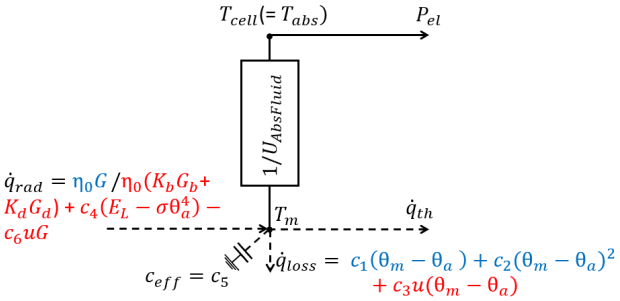


Figure 1. Two-node, one-capacitance thermal network for PVT collectors (ISO 9806: dashed lines; extension: solid lines).

DOI 10.3384/ecp218273

corporated, resulting in the quasi-dynamic formulation (Equation 4), which captures the effects relevant for unglazed PVT thermal operation (Stegmann et al. 2011; Fischer and Müller-Steinhagen 2009). Compared to a two-node, two-capacitance network[2], this two-node, single-capacitance approach is ISO-compliant, simpler, and directly uses standardized parameters—and has been demonstrated as the preferred method for the PVT collectors investigated by Jonas et al. (2019).

The specific thermal output is calculated as:

$$\dot{q}_{\text{th}} = \frac{\dot{m}\, c_p\, (T_{\text{out}} - T_{\text{in}})}{A_G}, \tag{9}$$

where $\dot{m}$ is the mass flow rate ($\mathrm{kg\,s^{-1}}$) and c_p the HTF specific heat capacity ($\mathrm{kJ\,kg^{-1}\,K^{-1}}$).

3.1.2 Thermal–electrical coupling

Both numerical and experimental approaches can be used to determine U_{AbsFluid} for a specific absorber design. However, the absorber-specific parameters required by these methods are rarely available for commercial PVT collectors, where test reports and datasheets are usually the only information source. PVT models that rely on parameter estimation typically fit U_{AbsFluid} alongside other thermal and electrical parameters. Since this work aims to develop an accurate PVT model based solely on datasheet inputs, a datasheet-based derivation of U_{AbsFluid} is proposed by merging two established approaches. Stegmann et al. (2011) derived U_{AbsFluid} from OC-mode thermal tests (including wind-speed dependence) making it valid for unglazed collectors. Lämmle (2018) generalized this to MPP-mode glazed collectors but they omitted wind effects. The MPP-mode approach is preferred because the electrical operating point significantly influences thermal performance; if only OC-mode parameters are available, the corresponding MPP coefficients can be obtained via the conversion procedure of Lämmle (2018), which properly accounts for the coupling between electrical and thermal losses and avoids physically incorrect simplifications (Kramer and Helmers 2013).

Combining these two methods yields a unified, wind-dependent formulation using MPP datasheet coefficients:

$$U_{\text{AbsFluid}} = \frac{\left[(\tau\alpha)_{\text{eff}} - \eta_{0,\text{el}}\right]\left(c_1 + c_3\, u + b_{1,\text{el}}\right)}{\left[(\tau\alpha)_{\text{eff}} - \eta_{0,\text{el}}\right] - \left(1 - \frac{c_6}{\eta_{0,\text{th}}}\, u\right)\eta_{0,\text{th}}}. \tag{10}$$

Here $(\tau\alpha)_{\text{eff}} = 0.901$ for unglazed PVT collectors as reported in Lämmle (2018), and the electrical temperature-dependence coefficient is given by $b_{1,\text{el}} = |\beta|\, G_{\text{nom}}$, where β is the temperature coefficient in $\%\,\mathrm{K^{-1}}$ and $G_{\text{nom}} = 1000\,\mathrm{W\,m^{-2}}$. When computed across varying wind speeds u, the resulting U_{AbsFluid} values should remain approximately constant, serving as a consistency check on the datasheet parameters (Stegmann et al. 2011).

[2]Some PVT models introduce a second capacitance linked to T_{cell} to represent the thermal inertia of the PV panel, but this additional capacity cannot be determined using typically available manufacturer data.

3.1.3 Electrical performance

The electrical output power P_{el} is calculated using the PVWatts V5 formulation (Dobos 2014) as presented by Equation 11

$$P_{\text{el}} = P_{\text{nom}}\, \frac{G}{G_{\text{nom}}}\left[1 + \gamma(T_{\text{cell}} - T_{\text{nom}})\right](1 - P_{\text{loss}}), \tag{11}$$

where P_{nom} is the module's rated DC power under standard test conditions ($G_{\text{nom}} = 1000\,\mathrm{W\,m^{-2}}$, $T_{\text{nom}} = 25\,^{\circ}\mathrm{C}$), T_{cell} is the PV cell temperature from the two-node thermal model (Equation 7), and P_{loss} is a constant system loss factor covering soiling, shading, mismatch, wiring and connection losses, inverter inefficiency, light-induced degradation (LID), and availability (Dobos 2014).

The term $\left[1 + \gamma(T_{\text{cell}} - T_{\text{STC}})\right]$ in Equation 11 explicitly accounts for the temperature-dependence of the electrical efficiency, similar to the performance ratio formulation in Equation 6. All remaining losses are merged into a system loss factor (P_{loss}). In this study, panels are assumed unshaded with negligible soiling, yielding a P_{loss} of 9 % per Dobos (2014). Given PVT 2's higher positive electrical output tolerance (up to +5 % vs. +3 % for PVT 1; see Table 1), its loss factor is reduced to 7 %.

3.2 Error analysis

Model validation is performed against the measurement data (Section 4) by comparing both instantaneous power and cumulative energy for the thermal and electrical outputs. Instantaneous errors are quantified using the mean absolute error (MAE) and root mean square error (RMSE), while cumulative deviations are evaluated over the full dataset. For a scale-independent assessment, normalized MAE (nMAE) and normalized RMSE (nRMSE) are also computed.

The MAE and nMAE between simulated and measured outputs are defined by Equation 12 and Equation 13 respectively.

$$\text{MAE}_Y = \frac{1}{n}\sum_{i=1}^{n} |Y_{\text{sim}}(t_i) - Y_{\text{meas}}(t_i)|, \tag{12}$$

$$\text{nMAE}_Y = \frac{\text{MAE}}{\frac{1}{n}\sum_{i=1}^{n} |Y_{\text{meas}}(t_i)|}. \tag{13}$$

The RMSE and nRMSE are defined by Equation 14 and Equation 15 respectively.

$$\text{RMSE}_Y = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (Y_{\text{sim}}(t_i) - Y_{\text{meas}}(t_i))^2}, \tag{14}$$

$$\text{nRMSE}_Y = \frac{\text{RMSE}}{\frac{1}{n}\sum_{i=1}^{n} |Y_{\text{meas}}(t_i)|}. \tag{15}$$

Where, Y represents either the thermal power output $\dot{Q}_{\text{th}}$ or the electrical power output P_{el}, depending on the context.

Table 1. Datasheet parameters for the two PVT collectors.

Parameters	PVT 1	PVT 2
Electrical		
A_{gross} (m^2)	1.66	1.64
β ($\%\,K^{-1}$)	-0.41	-0.375
P_{max} (W_p)	280 (0/+3 %)	300 (0/+5 %)
$\eta_{0,\text{el}}$ (%)	16.87	18.3
Thermal		
c_1 ($\text{W}\,\text{m}^{-2}\,\text{K}^{-1}$)	7.411	10.74
c_2 ($\text{W}\,\text{m}^{-2}\,\text{K}^{-2}$)	0	0
c_3 ($\text{kJ}\,\text{m}^{-3}\,\text{K}$)	1.7	1.1
c_4 ($-$)	0.437	0.363
c_5 ($\text{J}\,\text{m}^{-2}\,\text{K}^{-1}$)	42200	22100
c_6 ($\text{s}\,\text{m}^{-1}$)	0.003	0.036
$\eta_{0,\text{th}}$ (%)	0.475	0.535

4 Collector data and test sequences

Experimental data from two different unglazed PVT collecors, provided directly by the authors of the respective studies, are used for validation and are briefly summarized below.

The first dataset (PVT 1) was obtained from experimental measurements on an outdoor test bench at the Laboratory for Solar Energy Systems of the University of Applied Sciences (HTW Saar) in Saarbrücken, Germany. In this case, the PVT collector is an uncovered collector that features a rear cover and thermal insulation on the back of the absorber. A detailed description of this dataset and the corresponding test setup is provided in Jonas et al. (2019). The experimental measurements cover the typical days[3]as defined in ISO 9806 and include both standard thermal performance measurements and continuous recordings of the relevant electrical values.

The second dataset (PVT 2) was obtained from the real operation of a flat-plate, uncovered, and uninsulated PVT collector on a test bench in Austria. This dataset spans 58 summer days (from July 11 til September 6) with a 5-second sampling interval. The data, which is publicly available, is described in detail in Veynandt, Inschlag, et al. (2023). A selection of the measured variables from this dataset has previously been used in the development of a grey-box PVT model (Veynandt, Klanatsky, et al. 2023).

Table 1 lists the datasheet parameters describing the thermal and electrical behavior of both collectors. The thermal coefficients follow the quasi-dynamic (general) test method, following Equation 4. For PVT 2, the thermal coefficients were originally obtained under the unglazed steady-state method (Equation 2) and then converted to quasi-dynamic form (Equation 4) following the approach detailed in Solar Keymark Network (2019).

[3]Day type 1: η_0-conditions, mostly clear sky conditions.Day type 2: Elevated operating temperature or η_0-conditions, partly cloudy conditions including broken cloud and clear sky conditions.Day type 3: Mean operating temperature conditions including clear sky conditions.Day type 4: High operating temperature conditions including clear sky conditions.

As both datasets did not contain all data required to do a (one-to-one) validation of the model presented in Section 3.1, some assumptions had to be made. First, for PVT 1, the measured long-wave irradiance on the tilted plane was found to be unreliable, necessitating the estimation of long-wave radiation losses from the effective sky temperature. Only the solar irradiance components on the tilted plane were measured. Estimating cloud cover (and thus sky temperature) requires horizontal-plane irradiance data to determine the diffuse fraction. In the absence of such data, the diffuse fraction is assumed identical across all orientations. This assumption affects the sky temperature estimate and, consequently, the modeled long-wave radiation losses. Second, for PVT 2 the total irradiance on the tilted collector plane is used directly in the simulation following Equation 2; accordingly, no conversion of the datasheet value for $\eta_{0,\text{hem}}$ is necessary.

5 Results and analysis

This section is organized as follows: Section 5.1 validates the thermal and electrical performance against the measurement data introduced in Section 4, both in terms of instantaneous power and cumulative energy. Sections 5.2 and 5.3 examine how thermal and electrical parameters affect the model's accuracy.

5.1 Model validation

The accuracy of the PVT collector model is assessed against the two datasets introduced in Section 4, using only the datasheet coefficients from Table 1 to set up the PVT collector model. The evaluation of PVT 1 is based on Table 2 and Figure 3, while PVT 2 is assessed using Figure 2. The sole model parameter not directly provided in the datasheet, namely the overall absorber-fluid heat transfer coefficient U_{AbsFluid}, is computed for each PVT collector using Equation 10. As discussed in Section 3.1.2, Stegmann et al. (2011) argue that similar U_{AbsFluid} values across different wind speeds is indicative of robust datasheet parameters. In our case, both datasets exhibit a pronounced variation of U_{AbsFluid} with wind speed, suggesting a possible inconsistency in the wind-dependent coefficients c_3 and c_6. For the reference scenario, the wind-independent U_{AbsFluid} values for PVT 1 and PVT 2 are 32.76 $\text{W}\,\text{m}^{-2}\,\text{K}^{-1}$ and 56.85 $\text{W}\,\text{m}^{-2}\,\text{K}^{-1}$, respectively. Section 5.3 will further explore how the choice of U_{AbsFluid} affects the accuracy of the predicted electrical power output.

Table 2 reports the error metrics (introduced in Section 3.2) for PVT 1. For day types 1 to 3, thermal performance shows acceptable agreement between model and measurements, with energy deviation (ΔE) remaining below 5 %. In contrast, day type 4 shows a significantly higher deviation of 36.7 %, indicating model limitations under high temperature difference and low irradiance conditions (see Figure 3(d)). These discrepancies likely stem from underestimated wind-dependent losses (coefficients c_3 and c_6) and increased radiative losses due to the larger

DOI 10.3384/ecp218273

HTF-to-ambient temperature difference. As the absolute thermal output is low in this case, model limitations have a stronger effect, leading to disproportionately large relative deviations. While the MAE and RMSE are lowest under clear-sky conditions, larger errors during variable-sky periods (type 2 and type 4) point to a mismatch in the dynamic response of the thermal model, involving the c_5 coefficient. Additionally, the use of a sky temperature approximation introduces uncertainty in the long-wave loss term c_4, as noted in Section 4, and warrants closer examination. Electrical performance validation of PVT 1 yields normalized MAE and RMSE values below 3.1 % (Table 2), confirming the accuracy of the electrical component of the PVT collector model.

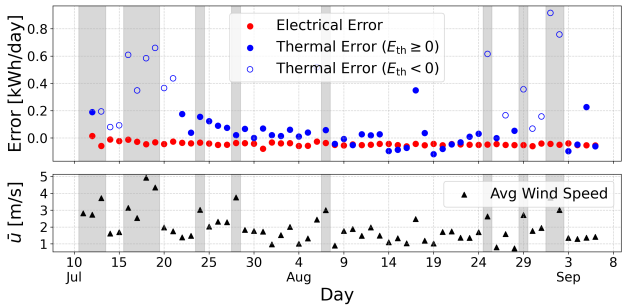


Figure 2. Top: Error between the measured and simulated daily thermal and electrical energy production of the PVT 2 collector, expressed in $\mathrm{kWh\,d^{-1}}$, over the entire dataset. Bottom: Corresponding average wind speed during operation for each day, with high wind speed periods ($\bar{u} > 2.5\,\mathrm{m/s}$) indicated by the grey sections.

For PVT 2, Figure 2 compares measured and simulated daily thermal and electrical energy alongside average wind speed, and Table 2 summarizes the key error metrics. Figure 2 indicates a clear correlation between thermal energy mismatch and higher wind speeds, suggesting further investigation of the wind-dependent coefficients c_3 and c_6. Over the full validation period at 5-s resolution the thermal model predicts 20.4 kWh vs. 13.4 kWh measured ($\Delta E = +53.1\%$), but this large error is driven by many periods of negative thermal output that would not occur in real operation. Filtering to $Q_{\mathrm{th,sim}}(Q_{\mathrm{th,meas}}) > 0$ yields a more realistic energy error of $\Delta E = +6.85\%$ (-4.7%). Electrical predictions remain robust (MAE: 3.2 W, ME: –1.8 W, $\Delta E = -2.9\%$, 83.8 kWh predicted vs. 86.3 kWh measured) and are essentially unaffected by wind speed.

5.2 Thermal parameters

Thermal performance is first examined for PVT 1 under controlled conditions, followed by an assessment of deviations for PVT 2 including high wind speed operation.

PVT 1: standard test conditions

Analysis is based on four representative day types (Figure 3), spanning clear to overcast skies with mean HTF–ambient temperature differences of 0 K (types 1–2), 10 K (type 3) and 20 K (type 4). An artificial blower maintained a constant wind speed of $3\,\mathrm{m\,s^{-1}}$ to $4\,\mathrm{m\,s^{-1}}$, such that wind effects remain uniform across cases. The stacked loss distributions represent the individual heat loss contributions within the quasi-dynamic thermal model (Equation 4), each defined by a specific datasheet coefficient. The impact of these loss terms on thermal performance is examined in the remainder of this section. The heat-loss coefficients c_1 and c_2 are excluded from further discussion due to their negligible influence on model accuracy under the given operating conditions.

Wind-speed dependence (c_3, c_6). In this study, wind speeds during all day type measurements (Figure 3) remained within $3\,\mathrm{m\,s^{-1}}$ to $4\,\mathrm{m\,s^{-1}}$, placing them at the upper edge of the ISO calibration range (0–3.5 m/s) and calling the accuracy of the wind-dependent coefficients into question. The persistent steady-state error across all day types indicates an underestimation of c_6, which is significant even for day types 1–2 where c_3 has limited impact due to small $\theta_m - \theta_a$, while c_3 becomes more influential at larger temperature differences. These findings suggest that convective losses do not scale linearly beyond the calibration range.

Sky-temperature dependence (c_4). The c_4 term accounts for radiative heat losses based on the difference between ambient air and sky temperature, expressed as $\left(E_L - \sigma\,\theta_a^4\right)$. This term is typically negative, with values around $-100\,\mathrm{W\,m^{-2}}$ for clear skies, $-50\,\mathrm{W\,m^{-2}}$ under hazy conditions, and approximately zero for overcast skies (ISO 9806 2013). In this work, the sky temperature for PVT 1 is approximated according to the assumptions in Section 4, which may introduce steady-state errors in the long-wave loss term and affect the accuracy of the thermal simulation.

Thermal capacity (c_5). ISO 9806 limits uncertainty in HTF heat capacity to ±10 %, yet quasi-dynamic identification often yields larger deviations (Weißmüller, Frenz, and Krämer 2012), attributable to limited variation in HTF temperature during standard tests. Calculations based on material properties and ISO weighting factors can reduce this uncertainty (Lämmle 2018). However, identified values may still fail to capture rapid irradiance transients (Stegmann et al. 2011). In this study, the datasheet c_5 is too large, slowing the model's thermal response (Figure 3) and causing notable errors during fast-changing conditions. Reducing c_5 would sharpen transients but exacerbate steady-state bias, indicating that adjustment of c_5 alone cannot resolve the dynamic mismatch.

Comparison with quasi-dynamic identification. For day types 1–2, underestimation of steady-state losses leads to overestimated thermal output, implicating the sky-temperature term c_4 and/or the wind-dependent term c_6. A similar bias occurs in types 3–4, but larger $\theta_m - \theta_a$ elevates the influence of c_3, while the model's slower transient response points to an overestimated thermal capacity c_5. Compared to the parameter set identified by Jonas et al. (2019), the datasheet coefficients differ by

Table 2. Model validation results for PVT 1 and PVT 2. Computations use a 3-segment discretization, datasheet parameters, a loss factor of 9 % for PVT 1, and 7 % for PVT 2. For PVT 1, $U_{\text{AbsFluid}} = 32.76\,\text{W}\,\text{m}^{-2}\,\text{K}^{-1}$; for PVT 2, $U_{\text{AbsFluid}} = 56.85\,\text{W}\,\text{m}^{-2}\,\text{K}^{-1}$.

	Thermal Performance							
	MAE (W)	nMAE (%)	RMSE (W)	nRMSE (%)	ME (W)	E_{sim} (kWh)	E_{meas} (kWh)	ΔE (%)
PVT 1 (Day type 1)	18.5	3.33	20.4	3.66	18.5	3.73	3.61	3.33
PVT 1 (Day type 2)	91.5	18.8	132.2	27.1	10.7	3.06	2.99	2.19
PVT 1 (Day type 3)	58.4	19.9	92.5	31.6	12.2	2.33	2.23	4.17
PVT 1 (Day type 4)	97.4	85.7	151.2	133.0	41.7	0.78	0.57	36.7
PVT 2	10.7	111.2	33.9	352.9	5.10	20.4	13.4	53.1
PVT 2 ($Q_{\text{th,sim}} > 0$)	43.1	30.8	66.9	47.8	9.57	26.6	24.9	6.85
PVT 2 ($Q_{\text{th,meas}} > 0$)	35.2	21.0	52.1	31.2	−7.86	25.5	26.8	−4.7
	Electrical Performance							
	MAE (W)	nMAE (%)	RMSE (W)	nRMSE (%)	ME (W)	E_{sim} (kWh)	E_{meas} (kWh)	ΔE (%)
PVT 1 (Day type 1)	1.04	0.49	1.36	0.64	0.91	1.00	1.00	0.43
PVT 1 (Day type 2)	4.30	2.48	5.34	3.08	3.72	1.14	1.11	2.15
PVT 1 (Day type 3)	2.80	1.61	3.43	1.97	2.35	1.29	1.27	1.35
PVT 1 (Day type 4)	3.26	1.98	3.88	2.36	2.78	0.86	0.84	1.69
PVT 2	3.21	5.18	6.11	9.86	−1.80	83.8	86.3	−2.91

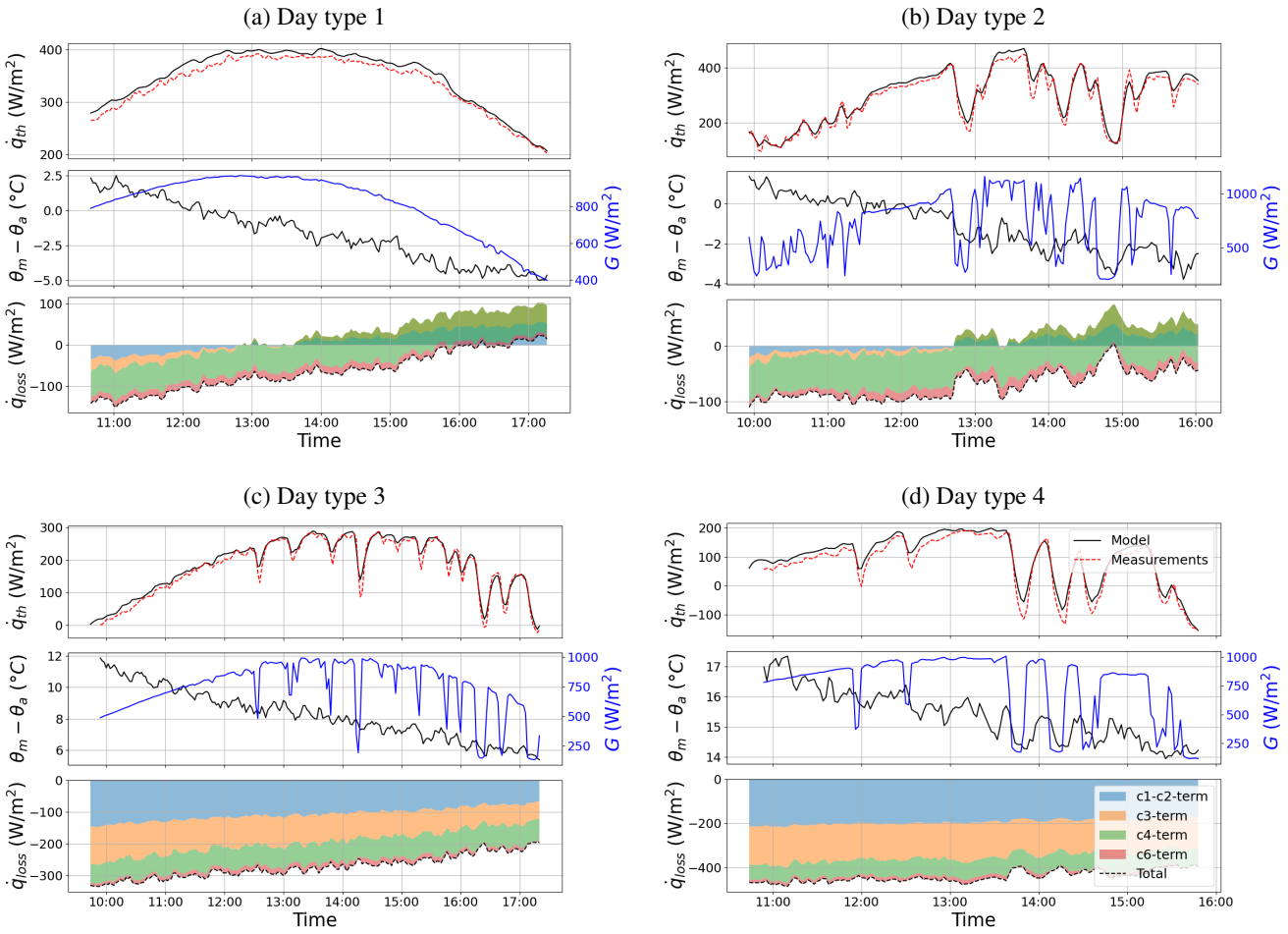


Figure 3. Comparison between simulated and measured thermal output and losses for different day types. Each subfigure shows: Top: thermal output $\dot{q}_{th}$ (model vs. measurements); Middle: temperature difference $\theta_m - \theta_a$ and solar irradiance; Bottom: simulated thermal loss contributions and total loss. A single common legend (identical across day types) is displayed in day type 4.

DOI
10.3384/ecp218273

$\Delta c_3 = -1.4\,\%$, $\Delta c_4 = 63.1\,\%$, $\Delta c_5 = 11.5\,\%$, and $\Delta c_6 = -90.3\,\%$. The strong negative bias in c_6, partly offset by a positive bias in c_4, highlights that model limitations peak under high wind speeds, as further illustrated by the thermal behavior of PVT 2.

PVT 2: high wind conditions

As shown in Figure 2 and discussed in Section 5.1, the model systematically overestimates the thermal output of PVT 2, particularly under higher wind speeds. The collector frequently operates near its upper temperature limit, with a mean HTF-to-ambient temperature difference of 19.52 K, which is suboptimal for unglazed PVT collectors. This condition, resulting from the test setup described in Veynandt, Inschlag, et al. (2023), leads to phases of low or even negative thermal output. While not representative of typical operation, this validation is valuable for exposing model limitations.

The pronounced overestimation under high wind conditions (occasionally exceeding 10 $\mathrm{m\,s^{-1}}$) suggests that the wind-dependent coefficients c_3 and c_6 do not scale linearly beyond the calibrated range. This effect is exacerbated by the uninsulated rear side of the collector, increasing its sensitivity to convective losses. Additionally, the use of a varying mass flow rate, differing from the (nominal) one used for datasheet coefficient determination, may contribute to discrepancies. Because these effects primarily affects convective heat transfer and potentially electrical performance, they will be addressed in Section 5.3.

5.3 Electrical parameters

As discussed in Section 2.2.2, the PV cell temperature strongly affects the electrical efficiency of a PVT collector (see Equation 6). To account for the influence of thermal operation on the PV cell temperature, a thermal coupling between the PV cell and the HTF is introduced via the coefficient U_{AbsFluid}, which is formulated in Equation 7. In this work, U_{AbsFluid} is estimated using a simplified, datasheet-based method (Section 3.1.2). This section examines the impact of variations in the estimated U_{AbsFluid} on the accuracy of the modeled electrical power output.

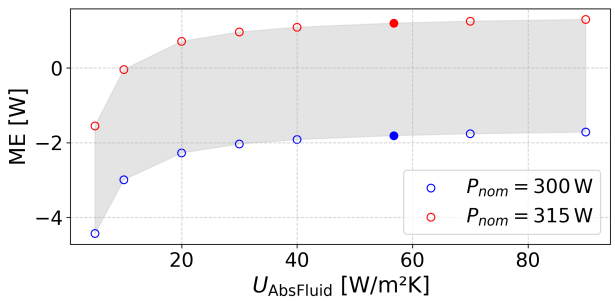


Figure 4. Mean electrical error versus U_{AbsFluid} for $P_{\mathrm{nom}} =$ 300 W and 315 W. Filled markers denote the estimated (datasheet-based) value of U_{AbsFluid}.

Figure 4 shows the normalized mean error in electrical output versus variations in U_{AbsFluid} for two nominal powers: $P_{\mathrm{nom}} = 300\,\mathrm{W}$ and 315 W (reflecting a +5 % tolerance; see Table 1). A constant system loss factor of 9 % is applied throughout (Equation 11). As shown, a shift in U_{AbsFluid} has minimal effect under good thermal coupling, while the grey band, representing uncertainty from rated power tolerance and assumed P_{loss}, causes greater deviation. The trend observed in Figure 4 aligns with Lämmle (2018), who found that values above $60\,\mathrm{W\,m^{-2}\,K^{-1}}$ indicate good coupling, with little gain beyond. Hence, a simplified datasheet-based U_{AbsFluid} suffices for accurate output predictions. Making it mass-flow dependent adds complexity without meaningful accuracy gains, as constant values already yield reliable results.

5.4 Model limitations

The following limitations apply primarily to unglazed PVT collectors: (1) High wind speeds: convective losses (coefficients c_3, c_6) are increasingly underestimated above the ISO calibration range ($\approx 3.5\,\mathrm{m\,s^{-1}}$); (2) Dynamic response: thermal capacity (c_5) may be misestimated, impacting the model's transient behavior under rapidly varying irradiance. Glazed PVT collectors are not affected by (1) but remain subject to (2); this hypothesis should be investigated in future work.

6 Conclusion and future work

By offering an open-source, modular, and transparent modeling framework, this work sets a significant step toward democratizing high-fidelity PVT system simulations, thereby empowering researchers, engineers, and policymakers to optimize PVT systems without relying on proprietary tools or inaccessible data.

Validation against two experimental datasets demonstrated that the electrical sub-model achieves very good accuracy, with the nMAE and nRMSE both below 3.1 % for PVT 1 (day types 1–4) and 5.2 % / 9.9 % for PVT 2. The thermal model reproduces daily energy within 3.3 %, 2.2 %, 4.2 % and 36.7 % for PVT 1 day types 1–4, respectively, and 6.85 % for PVT 2 (excluding negative thermal output periods).

These results confirm that, except under extreme conditions, the simplified datasheet-based approach yields reliable predictions. Limitations arise primarily for unglazed collectors for high wind speeds, where convective and radiative losses are not fully captured.

Future work should validate glazed PVT collectors, whose insulation and higher operating temperatures may mitigate the limitations identified for unglazed systems. For unglazed collectors, investigations into non-linear convective formulations could better characterize the wind-dependent loss coefficients c_3 and c_6. Refining the sky-temperature approximation is also needed to reduce steady-state radiative errors. Finally, embedding this PVT model within full energy-system simulations (e.g., district heating networks or building HVAC models) would enable end-to-end optimization of hybrid solar technologies across diverse climates and applications.

7 Data availability

The PVT model and its validation are available in the IDEAS Library: `https://github.com/open-ideas/IDEAS/pull/1446`. The measurement data is available in the IDEAS resources folder.

Acknowledgements

The authors acknowledge the funding by the KU Leuven C3 ASiRE^2 project (C3/22/029) and the European Union through the SEEDs project under the Horizon Europe Programme. Views and opinions expressed are however those of the author(s) only and do not neces- sarily reflect those of the European Union or CINEA. Neither the European Union nor the granting authority can be held responsible for them.

References

Brottier, Laetitia (2019). "Optimisation biénergie d'un panneau solaire multifonctionnel : du capteur aux installations in situ". PhD thesis. Université Paris-Saclay (COmUE).

Dobos, Aron P. (2014-09). *PVWatts Version 5 Manual*. Tech. rep. NREL/TP-6A20-62641, Contract No. DE-AC36-08GO28308, Prepared under Task No. SS13.5030. 15013 Denver West Parkway, Golden, CO 80401, USA: National Renewable Energy Laboratory (NREL). URL: https://www.nrel.gov/docs/fy14osti/62641.pdf.

ESTIF/SKN (n.d.). *The Solar Keymark Scheme Rules*. http://www.estif.org/solarkeymarknew/the-solar-keymark-scheme-rules. Accessed 2024.

Fischer, S. and H. Müller-Steinhagen (2009). "Collector efficiency testing – the 2-node collector model ready for implementation in European Standard EN 12975". In: *Proceedings of the ISES Solar World Congress 2009*. International Solar Energy Society (ISES). Johannesburg, South Africa.

Florschuetz, L. W. (1976). "Extension of the Hottel-Whillier model to the analysis of combined photovoltaic/thermal flat plate collectors". In: *Solar Energy* 22, pp. 361–366.

Herrando, María, Christos N. Markides, and Klaus Hellgardt (2014). "A UK-based assessment of hybrid PV and solar-thermal systems for domestic heating and power: System performance". In: *Applied Energy* 122, pp. 288–309. DOI: 10.1016/j.apenergy.2014.01.061.

Hofmann, P. et al. (2010). "Measurements and Benchmark of PV-T Collectors According to EN 12975 and Development of a Standardized Measurement Procedure". In: *Proceedings of the EuroSun 2010 Conference*. Graz, Austria.

IEA SHC (2018). *Task 60. PVT Systems: Application of PVT Collectors and New Solutions in HVAC Systems. Official Description.* https://www.iea-shc.org/task60. International Energy Agency Solar Heating and Cooling Programme.

Jonas, Danny et al. (2019). "Performance modeling of PVT collectors: Implementation, validation and parameter identification approach using TRNSYS". In: *Solar Energy* 193, pp. 51–64. DOI: 10.1016/j.solener.2019.09.047.

Kang, A. et al (2021). "Modeling of photovoltaic-thermal district heating with dual thermal modes". In: *Journal of Physics: Conference Series* 2042.1, p. 012090.

Kramer, Korbinian, N. Amrizal, et al. (2020-10). *Status Quo of PVT Characterization*. Tech. rep. SHC Task 60 - Report B1. Fraunhofer ISE, University of Perpignan, Dual Sun, University of Catania, University of Stuttgart, University of Zaragoza, Saarland University, TNO, ZAE Bayern, Czech Technical University, Dalarna University. DOI: 10.18777/ieashc-task60-2020-0004.

Kramer, Korbinian and Henning Helmers (2013). "The interaction of standards and innovation: Hybrid photovoltaic–thermal collectors". In: *Solar Energy* 98, pp. 434–439. DOI: 10.1016/j.solener.2013.08.042.

Lämmle, Manuel (2018-01). "Thermal management of PVT collectors: development and modelling of highly efficient glazed, flat plate PVT collectors with low emissivity coatings and overheating protection". PhD thesis. University Freiburg.

Lämmle, Manuel et al. (2020-05). *Basic Concepts of PVT Collector Technologies, Applications and Markets*. Tech. rep. SHC Task 60/Report D5. Fraunhofer ISE, University of Zaragoza, Sunovate. DOI: 10.18777/ieashc-task60-2020-0002.

Maier, Laura et al. (2023-10). "Comparative Study and Validation of Photovoltaic Model Formulations for the IBPSA Modelica Library based on Rooftop Measurement Data". In: *Proceedings of the Modelica Conference 2023*. Aachen, Germany. DOI: 10.3384/ecp204131.

Skoplaki, E. and J.A. Palyvos (2009). "On the temperature dependence of photovoltaic module electrical performance: A review of efficiency/power correlations". In: *Solar Energy* 83.5, pp. 614–624. DOI: 10.1016/j.solener.2008.10.008.

Solar Keymark Network (2019-10). *Thermal Performance Parameter Conversion to the ISO9806:2017*. URL: https://solarheateurope.eu/wp-content/uploads/2019/10/SKN-N0474R0_Thermal-performance-parameter-conversion-to-the-ISO9806-2017.pdf.

Stegmann, M. et al. (2011). "Model of an unglazed photovoltaic thermal collector based on standard test procedures". In: *Proceedings from SWC2011: ISES Solar World Congress*. kassel (GE), 28 August - 2 September 2011.

Veynandt, François, Franz Inschlag, et al. (2023). "Measurement data from real operation of a hybrid photovoltaic-thermal solar collectors, used for the development of a data-driven model". In: *Data in Brief* 49, p. 109417. DOI: 10.1016/j.dib.2023.109417.

Veynandt, François, Peter Klanatsky, et al. (2023). "Hybrid photovoltaic-thermal solar collector modelling with parameter identification using operation data". In: *Energy and Buildings* 295, p. 113277. DOI: 10.1016/j.enbuild.2023.113277.

Weißmüller, C., H. Frenz, and E. Krämer (2012-05). *Proficiency Test - QAiST testing of solar collectors and solar systems 2010 - 2011. Final report*. Tech. rep. Final report. Daimlerstraße 8, D-45770 Marl, Germany: Institut für Eignungsprüfung GmbH. URL: https://task43.iea-shc.org/data/sites/1/publications/Final%20QAiST%20Report%205-29-131.pdf.

Wetter, M. et al. (2019-09). "IBPSA Project 1: BIM/GIS and Modelica framework for building and community energy system design and operation–ongoing developments, lessons learned and challenges". In: *IOP Conference Series: Earth and Environmental Science*. Vol. 323. Graz, Austria: IOP Publishing, p. 012114. DOI: 10.1088/1755-1315/323/1/012114.

Zondag, H.A. (2008). "Flat-plate PV-Thermal collectors and systems: A review". In: *Renewable and Sustainable Energy Reviews* 12.4, pp. 891–959. DOI: 10.1016/j.rser.2005.12.012.

Modelling Aquifer Thermal Energy Storage (ATES) System with Buoyancy Flow

Markus Gillner[1] Arne Speerforck[1]

[1]Hamburg University of Technology, am Schwarzenberg-Campus 1, Hamburg, Germany, Institute of Engineering Thermodynamics, {markus.gillner,arne.speerforck}@tuhh.de

Abstract

Aquifer Thermal Energy Storages (ATES) as long-term storages have a strong potential to address the seasonal discrepancy of supply and demand of thermal energy. The operation of high-temperature ATES (HT-ATES) and their integration into district heating systems are the subject of current research projects. Buoyancy plays an important role in determining how HT-ATES performs. The target of this paper is to present a system model in Modelica that takes these buoyancy effects into account. The validation with experimental data and numerical simulations shows that the system model represents the buoyancy effects well. A sensitivity analysis underlines the importance of optimizing the grid structure and shows that a high resolution in the aquifer is necessary, especially in the vertical direction. Finally, a 10-year simulation shows the deviation of the heat recovery factor, i.e. the ratio of the amount of heat extracted to the amount of heat injected, between a model with and without buoyancy effects.

Keywords: High Temperature Aquifer Thermal Energy Storage, ATES, Buoyancy, System Model, TransiEnt Library

1 Introduction

In 2023, only 17 % of Germany's final energy consumption in the heating sector was produced from renewable sources (Umweltbundesamt 2024). One of the major challenges transforming the heating transition is the seasonal offset between the supply of sustainable heat and demand. Aquifer thermal energy storage (ATES) is one of the most promising seasonal storage systems due to its high storage capacity, efficiency and small above-ground footprint (Guelpa and Verda 2019). The principle of ATES is based on two wells (doublets) drilled into water-bearing soil layers, called aquifer. Thermal water is extracted from one well (cold or compensation well), heated and injected into the second well (hot well). During the heating period, the heated thermal water is pumped up from the hot well, the heat is extracted via heat exchangers and returned to the second well.

A distinction is made between low-temperature ($< 50\,^{\circ}\mathrm{C}$) and high-temperature ($\geq 50\,^{\circ}\mathrm{C}$) aquifer storage, depending on the temperature of the thermal water to be stored. Due to the low temperature level, LT-ATES also serve as cold storage for the summer. While LT-ATES are widely used, only 5 HT-ATES are in operation worldwide (Fleuchaus 2020). Some have been closed after years of operation (e.g. Utrecht University, Hooge Burch in Zwammerdam and Neubrandenburg). In Neubrandenburg and Hooge Burch, Zwammerdam this was due to uneconomic operation due to fluctuating heat demand and supply (Fleuchaus et al. 2021; Drijver, Bakema, and Oerlemans 2019). The ATES at the University of Utrecht closed due to problems in the well and a discrepancy between the required temperature level and the temperature supplied from the aquifer storage (Drijver, Bakema, and Oerlemans 2019). This means that the interaction between the aquifer storage and its surroundings, be it individual buildings or district heating systems with their fluctuating heat demand and supply, requires further research.

New HT-ATES projects such as the Living Lab "GeoSpeicher Berlin" (Federal Ministry for Economic Affairs and Climate Action 2024) and "PUSH-IT" (Push it 2024) are building HT-ATES systems for integration into district heating systems, and projects such as "ATES Vienna" (Forschungsinitiative Green Energy Lab 2022) and "OptInAquiFer" (Hamburg Insitut 2022) are investigating the integration of HT-ATES into district heating systems. Köhler et al. (2025) from the OptInAquiFer project have identified a great potential for seasonal thermal energy storage in district heating systems, for which detailed models for the realisation of efficient storage systems are needed in order to be able to investigate the grid-side challenges in advance.

The most commonly used computer codes for the simulation of underground aquifers are complex CFD-based codes such as FEFLOW (DHI-WASY GmbH 2012), OpenGeoSys (Kolditz et al. 2012), MODFLOW (Langevin et al. 2017), HSTWin-2D (Kipp 1987) etc. These programs can represent complex geometries and take into account physical effects such as chemical reactions and groundwater flow in addition to heat and mass transport, but are only marginally suitable for system analysis. The simulation of aquifer storage in systems can only be represented with these programs using complex co-simulations. System level modelling platforms such as TRNSYS and Modelica are suitable for modelling the interaction of aquifer storage systems with buildings and district heating. For both modelling platforms, ATES

models already exist. The TRNAST (Schmidt 2005) model for TRNSYS allows the simulation of an aquifer storage system whose wells are thermally and hydraulically separated from each other. Heat transport is radial and vertical, while mass transport is radial only. The model of Maccarini et al. (2023) in the Modelica library IBPSA is a low-order model that is designed for fast numerical calculation and therefore only transports heat and mass radially. Both models neglect buoyancy flow due to larger temperature differences in the aquifer.

Various literature and studies have shown that buoyancy flow can have a significant impact on the efficiency of ATES systems (Buscheck, Doughty, and Tsang 1983; Gao et al. 2024; Beernink et al. 2024; Heldt, Beyer, and Bauer 2024). Gao et al. (2024) have shown with a non-dimensional approach that the design of the temperature profile in the aquifer can be classified into three different regimes: Conduction dominated, buoyancy dominated and a transition regime. Nield and Bejan (2013) gives the critical Rayleigh-Darcy number above which buoyancy can no longer be neglected as

$$\mathrm{Ra}_{\mathrm{critical}} = \frac{g\rho_{\mathrm{f}}^2 c_{p,\mathrm{a}} \beta \Delta T K L}{\mu \lambda} = 4\pi^2, \tag{1}$$

with β as the thermal expansion coefficient, g as the gravitational acceleration, ρ_{f} as the density, $c_{p,\mathrm{a}}$ as the specific heat capacity of the aquifer, K as the permeability, L as the characteristic length (here: height of aquifer), μ as the dynamic viscosity and λ as the thermal diffusivity. This means that even at low permeabilities (e.g. 0.1 Darcy) the temperature difference between injected and existing thermal water must not exceed 40 °C. This is almost always exceeded for high-temperature storage tanks. In addition to 2D heat transport, the ATES model presented here also takes buoyancy flow into account. In section 2 the mathematical and numerical model is described and validated against experimental and numerical data in section 3. section 4 presents an example for a 10 Years simulation of an ATES and in section 5 the main results are summarized. The entire model is to be added to the TransiEnt library (Gillner et al. 2025).

2 Method

The purpose of the model is to simulate the thermohydraulic behaviour of an ATES system for high and low temperature injection. The model should then be suitable for integration into larger system levels.

2.1 Modeling approach

The following assumptions are made in the model:

- The aquifer is homogenous and isotropic, so a 2D cylindrical representation is sufficient.
- A local thermal equilibrium between the fluid and the solid phase is assumed.
- Regional groundwater flow is neglected.
- Thermal interference between hot and cold well is neglected.
- The fluid in the aquifer is assumed to be pure water.
- The porosity, the permeability, the height of the layers and the material properties of the solid phase are constant.
- Radiative effects, viscous dissipation and work done by pressure changes are neglected in the energy equation.
- The well screen fully penetrates the aquifer.
- The aquifer is confined by impermeable layers above and below.

The last two assumptions depend on the modeling. They can be adjusted with little effort.

To describe convection and conduction in the aquifer, the model is based on the three governing equations for mass, momentum and energy conservation. The mass conservation equation is given by

$$\phi \cdot \frac{\partial \rho_{\mathrm{f}}}{\partial t} + \nabla \cdot (\rho_{\mathrm{f}} \cdot \mathbf{v}) = 0, \tag{2}$$

where ϕ is the porosity, ρ_f is the fluid density and $\mathbf{v}$ is the seepage velocity or also called Darcy flux given as $\mathrm{m}^3\,\mathrm{m}^{-2}$. The subscript f refers to fluid. The momentum conservation equation for porous medium considering laminar flow is given by

$$\mathbf{v} = -\frac{\mathbf{K}}{\mu}(\nabla p - \rho_f \mathbf{g}), \tag{3}$$

where $\mathbf{K}$ is the permeability tensor, μ is the dynamic viscosity, p is the pressure, and $\mathbf{g}$ is the gravitational acceleration vector. The energy conservation equation in the aquifer is given by

$$\left(\rho c_{\mathrm{p}}\right)_{\mathrm{a}} \frac{\partial T}{\partial t} + \left(\rho c_{\mathrm{p}}\right)_{\mathrm{f}} \mathbf{v} \cdot \nabla T = k_{\mathrm{a}} \cdot \nabla \cdot (\nabla T), \tag{4}$$

where c is the heat capacity, T is the temperature, k is the thermal conductivity, and the subscript a represents the coupled material values of the liquid and solid phase. As the model is assumed to be in thermal equilibrium, heat conduction is parallel in both phases. For an effective thermal conductivity, density and heat capacity, the weighted arithmetic mean is used as follows:

$$k_{\mathrm{a}} = (1-\phi)\, k_{\mathrm{s}} + \phi k_{\mathrm{f}}, \tag{5}$$

$$\rho_{\mathrm{a}} = (1-\phi)\, \rho_{\mathrm{s}} + \phi \rho_{\mathrm{f}}, \tag{6}$$

$$\rho_{\mathrm{a}} c_{\mathrm{p,a}} = (1-\phi)\, \rho_{\mathrm{s}} c_{\mathrm{s}} + \phi \rho_{\mathrm{f}} c_{\mathrm{p,f}}. \tag{7}$$

DOI 10.3384/ecp218283

Since the adjacent layers are assumed to be impermeable, the energy equation of Equation 4 is reduced by the advection term to

$$\left(\rho c_{\mathrm{p}}\right)_{\mathrm{a}} \frac{\partial T}{\partial t} = k_{\mathrm{a}} \cdot \nabla \cdot (\nabla T). \tag{8}$$

Density, heat capacity and dynamic viscosity of water in the aquifer are calculated as a function of temperature and pressure. The material parameter of the solid phase and the adjacent layers are assumed to be constant.

The boundary conditions for pressure and temperature at the outer edge of the aquifer are defined as the initial values for pressure and temperature. This corresponds to the hydrostatic pressure distribution for the pressure and the undisturbed groundwater temperature for the temperature. The same applies to the temperature boundary conditions at the lower and upper surfaces of the adjacent layers, respectively. For accurate simulation of ATES, it is therefore important to ensure that the ATES is modeled to a sufficient size to minimize boundary effects. In the radial direction, this can be estimated by the thermal radius, which is defined as follows:

$$R_{\mathrm{th}} = \sqrt{\frac{c_f \cdot V}{c_{\mathrm{a}} \cdot \pi \cdot H}}, \tag{9}$$

where c_f is the volumetric heat capacity of the fluid, V is the total volume of thermal water injected, c_{a} is the effective volumetric heat capacity of the aquifer and H is the thickness of the aquifer. The thermal radius indicates how far the thermal plume spreads in a radial direction. However, it should be noted that this estimate does not take into account natural convection, convection due to regional groundwater flow and heat conduction in the vertical direction. Nordell, Snijders, and Stiles (2021) recommends three times the thermal radius as the distance between the hot and cold well to prevent thermal interference. At least one and a half times the thermal radius should be selected for the mapping of a single well.

The adiabatic boundary condition at the inner edge of the neighbouring layers results from the cylindrical implementation of the ATES.

To solve the partial differential equations, the Finite Volume Method (FV) is used for local discretization. The nodes are placed in the center of the control volume. The upwind difference scheme is used as an approximation for the conservation variable. The advantage of this approximation lies in its robustness; it is the only approximation that does not lead to oscillating solutions (Ferziger, Perić, and Street 2020). However, the approximation has a leading truncation error term that is diffusive, i.e. it smoothes the solution. To keep this error term small, the grid must be chosen very finely.

2.2 Model implementation

The model is implemented in Modelica and uses components from the TransiEnt library (Senkel et al. 2021) and the Clara library (Vojacek et al. 2023). The TILMedia library available in Clara is used to calculate the temperature- and pressure-dependent properties of pure water. The model can roughly be divided into three levels. The top level represents the overall model of the underground well and can be used for instantiation in larger system levels. In this model, the models of aquifer, adjacent clay layers, well are instantiated and connected. Also the boundaries are set. One level lower the well, the aquifer and the stone layers are described. The energy equation, the continuity equation and the pressure loss model are located in the lowest level.

In Figure 1 the top layer with the models, boundary conditions and connections is shown schematically. The diagram is intended to illustrate that the discretization of the adjacent layers (clay layers) takes place on this level. However, the discretization of the aquifer and the well filter is implemented in the lower levels in order to reduce the number of fluid ports and fluid models. Furthermore, the Figure 1 shows that the *Aquifer* model also has fluid ports in the vertical direction. Thus, modeling of heterogeneous layered aquifers is also possible, as shown in section 3.

The material properties, the geometry of the aquifer and the adjacent layers, the initial conditions are set in a record and can be easily changed in the top level. The mesh for discretization is also set in the record. It is possible to design the discretization in vertical and radial direction very individually. The only restrictions are the same number and size of control volumes in the radial direction for the aquifer and the adjacent layers, as well as the axially symmetrical arrangement of the control volumes of the adjacent layers.

3 Validation

To validate the model, the results are compared with measured values from field experiments at Auburn University in 1981 and their simulation results. The experiment is described in Molz et al. (1983) and the simulation in Buscheck, Doughty, and Tsang (1983).

3.1 Experiment and simulation at Auburn University

In the 1970s, Auburn University began a series of field experiments on water storage at temperatures above 35 °C. In the third experiment, injection temperatures were increased on average to 58.5 °C in the first loop and to 81 °C in the second loop to study the effect of buoyancy flow on efficiency (Molz et al. 1983). The geometric, operational and material-related data on the aquifer and the experiment are taken from Buscheck, Doughty, and Tsang (1983). The aquifer under consideration lies at a depth of between 40 and 61 m and is bounded above by a 9 m thick clay layer. Above this is another aquifer. Below the aquifer is a layer consisting of clay, sand and sandstone. The undisturbed groundwater temperature is 20 °C. The

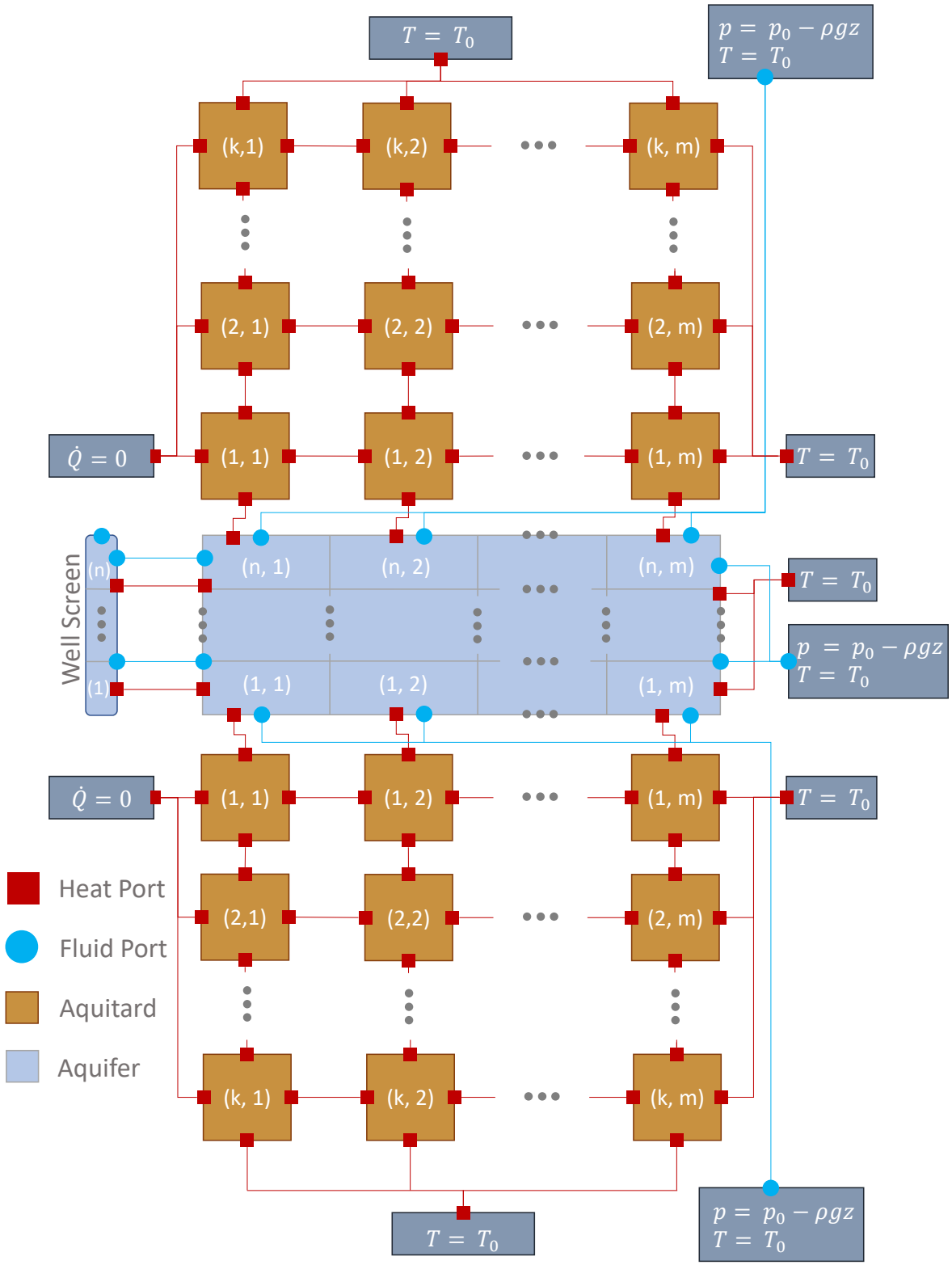


Figure 1. Schematic drawing of connecting the main layers at the top level.

DOI
10.3384/ecp218283

well screen extends across the full thickness of the aquifer and has a radius of 0.1 m. A compilation of the values used in this simulation can be found in Table 1. Although the second cycle is run at significantly higher temperatures, the calculation of the Rayleigh number according to Equation 1 with $\mathrm{Ra} = 403 > 4\pi^2 = \mathrm{Ra}_{critical}$ shows that buoyancy effects must be taken into account already in the first cycle. Therefore, the validation in this paper focuses on the first cycle.

The operating parameters of the first cycle are listed in the table Table 3. The first period of injection, storage and production lasts just under 87 days. The flow rate and temperature of the water in the injection phase are variable over time in the experiment. For the simulation, Buscheck, Doughty, and Tsang (1983) calculated mean values for injection temperature and volume flow. However, since these values can only be read from a graph, the values were compared to Lopik, Hartog, and Zaadnoordijk (2016).

Table 1. material parameters

Properties	*Parameter values*
Aquifer properties	
Porosity[1]	0.25
Density of rock[1]	$2600\,\mathrm{kg\,m^{-3}}$
Heat capacity of rock[1]	$696.15\,\mathrm{J\,kg^{-1}\,K^{-1}}$
Thermal conductivity of aquifer[1]	$2.29\,\mathrm{W\,m^{-1}\,K^{-1}}$
Horizontal permeability[1]	$0.63 \times 10^{-10}\,\mathrm{m^2}$
Vertical permeability[1]	$0.09 \times 10^{-10}\,\mathrm{m^2}$
Aquitard properties	
Porosity[1]	0.35
Density of rock[1]	$2600\,\mathrm{kg\,m^{-3}}$
Heat Capacity of rock[1]	$696.15\,\mathrm{J\,kg^{-1}\,K^{-1}}$
Thermal conductivity of aquitard[1]	$2.56\,\mathrm{W\,m^{-1}\,K^{-1}}$
Fluid properties	
Heat capacity of water	$4186\,\mathrm{J\,kg^{-1}\,K^{-1}}$
Thermal conductivity of water	$0.58\,\mathrm{W\,m^{-1}\,K^{-1}}$
Geometry	
Aquifer thickness[1]	21 m
Aquitard thickness[1]	9 m
Radius of well screen[1]	0.1 m
Initial conditions	
Initial temperature of aquifer[1]	20 °C
pressure of aquifer	5 bar

[1] Parameters taken from Buscheck, Doughty, and Tsang (1983).

In Buscheck, Doughty, and Tsang (1983) the field experiment is represented by two models. In the first model the aquifer is assumed to be a homogeneous layer. The second model takes more account of the actual heterogeneity of the aquifer by modelling a three-layer aquifer

Table 2. Horizontal permeability and height values for the three layers in heterogeneous model taken from Buscheck, Doughty, and Tsang (1983)

	Thickness in m	*Permeability in* $\mathrm{m^2}$
Upper Layer	9.6	4.6×10^{-10}
Middle Layer	5	1.16×10^{-10}
Lower Layer	6.6	4.6×10^{-10}

The vertical to horizontal permeability ratio in each layer is 1:6.

Table 3. Operational parameters for the first cycle taken from Buscheck, Doughty, and Tsang (1983).

Phase	*Duration in days*	*Flow rate in* $\mathrm{m^3\,d^{-1}}$	*Inj. temperature in* °C
Injection	20	760	60
	7	1100	58
	4	600	52
Storage	32	-	-
Production	2	-1684.4	-
	21	-1054.08	-
Rest	26	-	-

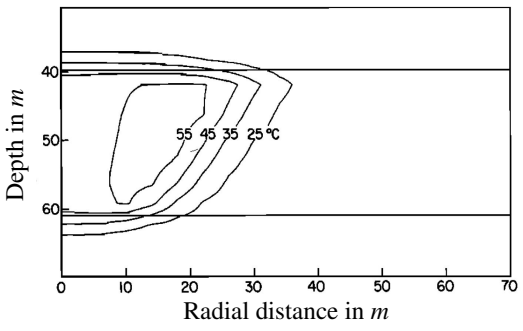
where the middle layer has a permeability 2.5 times higher than the two adjacent aquifer layers. The permeabilities and thicknesses of the aquifer layers are given in Table 2. A thickness of 9 m is assumed for the adjacent lower and upper layers and a permeability of $1 \times 10^{-15}\,\mathrm{m^2}$ is assumed for Buscheck, Doughty, and Tsang (1983). For the Modelica simulation, the upper and lower confined layers are assumed to be impermeable to water (aquitard). The temperature distribution in the aquifer and adjacent layers at the end of injection for the homogenous and heterogenous model can be seen in Figure 2.

3.2 Comparison of simulation results

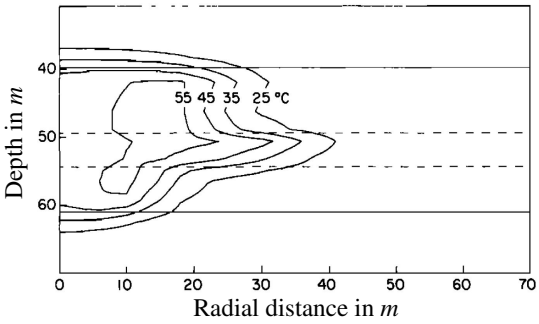
Both models with the homogeneous and heterogenous aquifer are rebuilt in Modelica and the results are compared with the results of the experiment and the two models of Buscheck, Doughty, and Tsang (1983). The comparison is made for the temperature distribution in the aquifer and its adjacent layers at the end of the injection phase and the temperature profile of the water in the well during production phase. The aquifer and its adjacent layers are discretized in volumes with 1 m edge length in vertical direction (except two smaller ones in heterogenous model to rebuild the height of the layers) and 1 m in radial direction until the end of the thermal plume at 40 m and then 10 volumes with 3 m. The temperature distribution in the aquifer and its adjacent layers for the homogeneous and heterogenous model at the end of the injection phase is shown in Figure 3a and 3b, respectively. The tilting of the thermal plume due to natural convection can be seen in both figures. In addition, the three layers in the aquifer are clearly visible in the path of the thermal plume in the het-

erogeneous model. Due to the higher permeability of the middle layer, the largest mass flow is expected there. The temperature distribution in the aquifer and the adjacent layers show a high degree of agreement with the simulations in Figure 2 and the experiment (Buscheck, Doughty, and Tsang 1983, p. 1130). Figure 4 shows the temperature curves of the water during the production phase in the wells for the experiment and all numeric models. The black line shows the measured values of the experiment, the red and green line shows the homogenous and heterogenous model, respectively. The dashed lines are the simulation results of Buscheck, Doughty, and Tsang (1983), the solid lines are the simulation results of this work. The temperature curves of the homogenous model is almost identical to the simulation results of Buscheck, Doughty, and Tsang (1983). The temperature curves of the heterogenous model show a slightly higher temperature in the production well than the simulation results of Buscheck, Doughty, and Tsang (1983). Both Modelica models initially show an increase in well temperature after the start of production. This is due to the different loading temperature of the thermal water during the injection phase and can also be seen in the temperature distribution. This first increase in the well temperature is neither recognizable in the simulation results of Buscheck, Doughty, and Tsang (1983) nor in the measured values from the experiment.

These results show that the Modelica model can not only predict the production well temperature affected by buoyancy, but also model more complex aquifer structures.

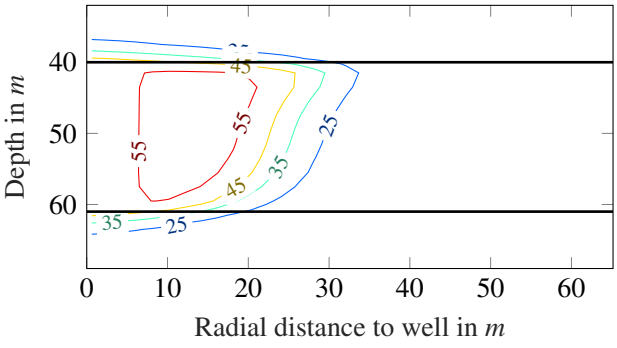


(a) Homogenous model

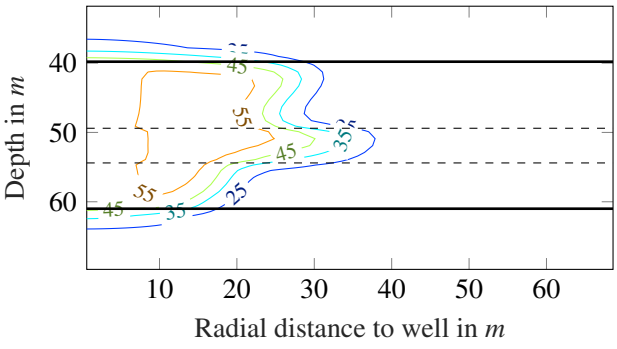


(b) Heterogenous model

Figure 2. Temperature distribution inside the aquifer and its adjacent layers at the end of the injection phase for the homogeneous (a) and heterogenous (b) model of Buscheck, Doughty, and Tsang (1983).



(a) Homogenous model



(b) Heterogenous model

Figure 3. Temperature distribution inside the aquifer and its adjacent layers at the end of the injection phase for the homogeneous (a) and heterogenous (b) Modelica model

4 Grid Analysis and 10 Years Simulation

This chapter presents the results of a 10-year simulation of a single well model. It shows that the model can be used as a system component in larger systems and compares the results with and without buoyancy.

For the simulation, the material properties and the geometry of the aquifer are taken from the section 3. A medium sized aquifer with a discharge of $50\,\mathrm{m^3\,h^{-1}}$ (Fleuchaus 2020) and a temperature of $60\,^{\circ}\mathrm{C}$ is assumed. The calculation of the thermal radius according to Equation 9 results in $R_{\mathrm{th}} = 66\,\mathrm{m}$. After first impressions of the simulation results, the modeled radius is set to 130 m. To keep the model structure simple, the adjacent layers are represented as 30 m thick aquitards, although there is a wide aquifer above the 9 m thick clay layer. The aquifer is also assumed to be homogeneous. The loading and unloading schedule is kept constant, and the individual phases - injection, storage, production, and resting - each last 91.25 days.

4.1 Grid study

In order to keep the accuracy of the results high and the computational speed as low as possible, a grid study is

DOI
10.3384/ecp218283

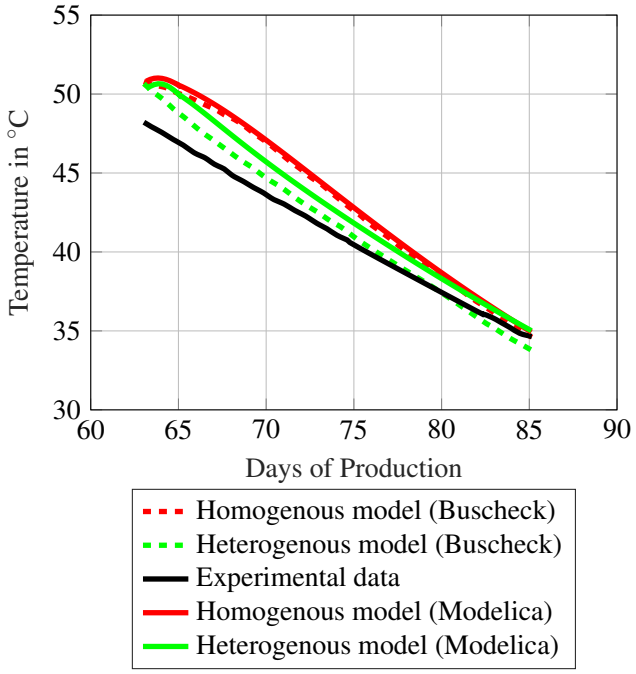


Figure 4. Temperature curves of the water during the production phase in the well for the experiment and all numeric models.

essential. For this purpose, a sensitivity analysis is performed in which the grid is changed individually and together in the radial and vertical directions. In addition, the grid structure is changed so that a finer grid structure is chosen in the radius of the thermal plume than outside of it. Inside the plume, the temperature gradients are steeper than outside, depending on the time. This is due to the numerical diffusion by the upwind difference scheme approach in FVM (Ferziger, Perić, and Street 2020).

Table 4 lists the structure of the grids in the sensitivity analysis. A total of nine different grid structures are tested. The first column assigns a number to the individual simulations. The second column lists the number of volumes in the radial direction, vertical direction of the aquifer and vertical direction of the aquitards, in that order. The third column contains the extent of the volumes in meters. If there are vectors in the second and third columns, the columns are assigned to each other. Radially, the discretization is performed from the well; vertically, the discretization is performed from bottom to top for the aquifer and from aquifer to outside for the adjacent layers. The fourth column shows the total number of volume elements. An example for the grid structure is shown in Figure 5.

The first simulation has the finest grid with an edge length of one meter. The subsequent simulations increase the edge length in the radial direction (S2 and S3) and in the vertical direction (S4 and S5). Simulations 6 and 7 combine the changes from simulations 2 and 4 as well as simulations 3 and 5. Simulations 8, 9, 10 and 11 increase the edge lengths outside the thermal plume and in radial direction.

Table 4. Grid Structure Variation

Number of Simulation	*Number of Volumes*	*Size of Volumes in* m	*Overall number of volumes*
First row: Grid in radial direction Second row: Grid in vertical direction in Aquifer Third row: Grid in vertical direction in Aquitard			
S1	130 21 30	1 1 1	10530
S2	65 21 30	2 1 1	5265
S3	{1,32} 21 30	{2,4} 1 1	2673
S4	130 {10,1} 15	1 {2,1} 2	5330
S5	130 {5,1} {1,7}	1 {4,1} {2,4}	2860
S6	65 {10,1} 15	2 {2,1} 2	2665
S7	{1,32} {5,1} {1,7}	{2,4} {4,1} {2,4}	726
S8	{80,10} 21 30	{1,5} 1 1	7290
S9	{80,5} 21 {1,1,3,3}	{1,10} 1 {1,2,3,6}	3145
S10	{40,5} 21 {1,1,3,3}	{2,10} 1 {1,2,3,6}	1665
S11	{1,19,5} 21 {1,1,3,3}	{2,4,10} 1 {1,2,3,6}	925

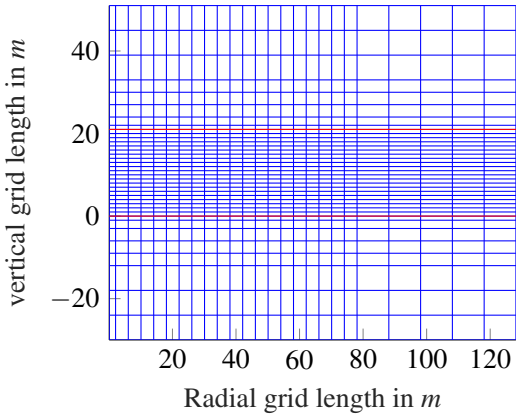


Figure 5. Visualisation of grid structure of simulation S11.

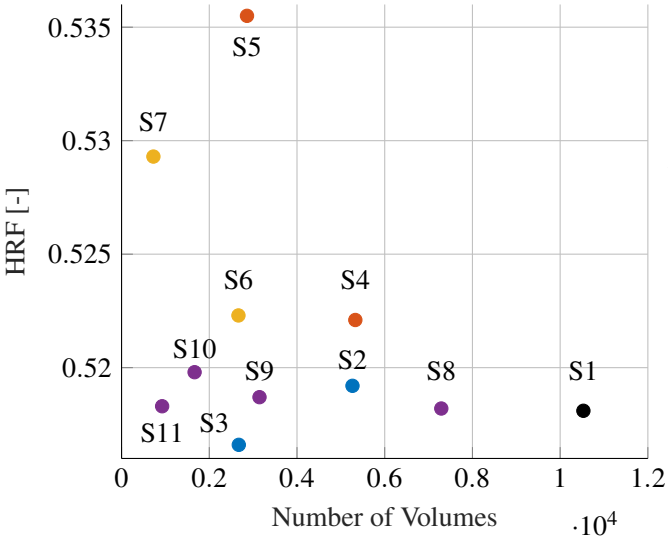


Figure 6. Heat Recovery Factor of the individual simulations of grid study after one year operation against the total number of volumes.

The simulations are compared using the Heat Recovery Factor (HRF), which is used as a measure of the efficiency of the ATES after a complete cycle. The HRF is calculated from the ratio of the amount of heat extracted to the amount of heat injected from the thermal water in a cycle:

$$\mathrm{HRF} = \frac{\int \dot{Q}_{\mathrm{out}} \mathrm{d}t}{\int \dot{Q}_{\mathrm{in}} \mathrm{d}t} = \frac{\int \dot{m}_{\mathrm{out}} \cdot c_{\mathrm{p}} \cdot (T_{\mathrm{out}} - T_{\mathrm{initial}})\, \mathrm{d}t}{\int \dot{m}_{\mathrm{in}} \cdot c_{\mathrm{p}} \cdot (T_{\mathrm{in}} - T_{\mathrm{initial}})\, \mathrm{d}t}, \quad (10)$$

where $\dot{Q}$ is the heat flow, $\dot{m}$ is the mass flow, c_p is the specific heat capacity, T is the temperature, and T_{initial} is the temperature of the undisturbed aquifer.

In Figure 6 the HRF of each simulation is plotted against the total number of volumes. The colors of the dots indicate the similarity of the variation. As expected, the coarser the grid structure, the greater the deviation of the HRF from the reference simulation. A coarser grid in the vertical direction in the aquifer has a significantly greater effect on the HRF than in the radial direction. Adjusting the grid outside the thermal plume has the least effect on the deviation. The number of volumes have a linear impact on the calculation time. By reducing the number of volumes by more than 11 times in simulation S11 compared to the reference simulation, the computation time is also reduced by almost 14 times to 10.2 s (Dymola2025x, Advanced.Translation.SparseActivate = true, Solver: DASSL, 1000 intervals, tolerance 0.0001 %). Simulation 11 is therefore used for the 10-year simulation.

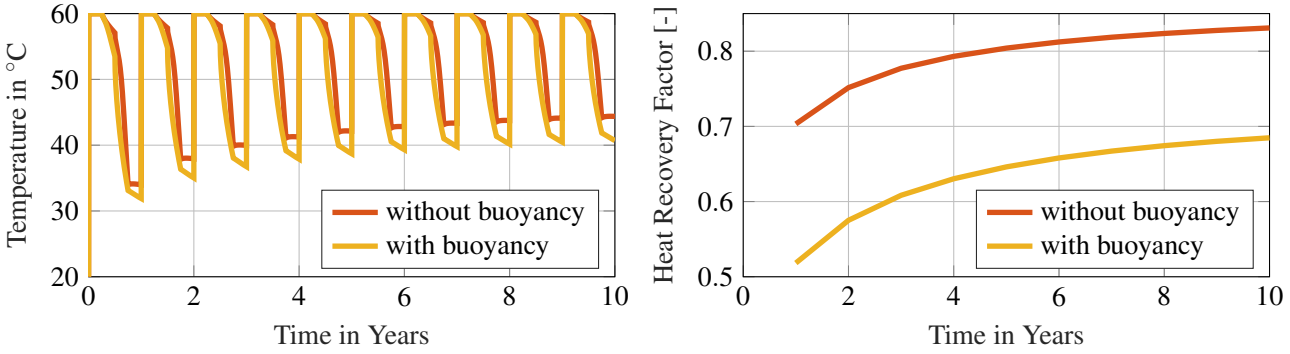
4.2 10 Years Simulation

The well temperature curve and the HRF of the ATES system are shown in Figure 7a and 7b respectively. In both figures the simulation without buoyancy is compared to the simulation with buoyancy. Comparing the well temperature of the two simulations shows significant differences in the three phases - storage, production and rest. In the storage and rest phases, the well fluid cools significantly faster. This is due to buoyancy and the resulting influx of colder fluid from the lower part of the aquifer. In addition, the lower adjacent layer is less heated than the upper adjacent layer. The production phase is characterized by a steeper decline in well temperature, resulting in a lower temperature at the beginning of the rest phase. This is caused by mixing processes in the well of cold thermal water from the lower part of the aquifer with warmer thermal water from the upper part of the aquifer. This results in a significantly lower heat recovery factor. The evolution of the HRF in Figure 7b of both simulations is similar. After 10 years the ATES has a HRF of more than 83 % without buoyancy. With buoyancy, the heat recovery drops below 70 %. Although the gap to the simulation without buoyancy decreases from over 18 % pt. in the first year to just under 15 % pt. in the tenth year, this clearly shows the relevance of considering the buoyancy forces. It can be seen from the two figures that the HRF will increase slightly in the following years.

5 Conclusion

The expansion of thermal energy storage is essential for the implementation of the heat transition, and in particular seasonal storage solutions are needed. ATES are a promising technology for seasonal heat storage. With their integration into district heating systems, HT-ATES in particular have become the focus of current research. However, buoyancy effects have to be taken into account in HT-ATES. The author is not aware of any system models that take this effect into account. This paper presents a model for the simulation of HT-ATES systems in Modelica. The model is capable of simulating the thermohydraulic properties of an ATES system with high and low temperature injection. The model has been validated with the results of field tests at Auburn University. The results show that the model is able to predict the temperature distribution in the aquifer and the temperature profiles in the production well with good accuracy. In addition, the model is capable of modeling more complex aquifer models.
A sensitivity analysis of the grid structure has been imple-

DOI
10.3384/ecp218283



(a) Well temperature curve over 10 years of operation

(b) Heat Recovery Factor developmet over 10 years of operation

Figure 7. Comparison of well temperature curve and Heat Recovery Factor between simulation with and without buoyancy flow.

mented to reduce computation time in case of acceptable deviations of the heat recovery factors from the reference simulation. The grid analysis shows that outside the thermal plume, the grid structure has less influence on the results. However, the vertical grid structure has a greater influence on the results than the radial grid structure. By adjusting the grid structure, the computation time can be reduced by more than 6 times with an acceptable deviation of the heat recovery factor from the reference simulation. In future work, the optimization of the grid structure in radial and vertical direction could be investigated to achieve further savings in computation time. The relevance of buoyancy effects for the simulation of ATES systems is emphasized by comparing the simulation results with and without buoyancy effects.

Acknowledgements

The authors greatly acknowledge the funding from the German Federal Ministry for Economic Affairs and Climate Action for the subproject "Planung und Bau eines Aquiferspeichers zur saisonalen Speicherung von Ab- und Überschusswärme" (Design and construction of Aquifer Thermal Energy Storage for seasonal storage of waste and surplus heat, project number: 03EWR007O2) within the project "Reallabore der Energiewende" (Living Labs of the Energy Transition).

References

Beernink, Stijn et al. (2024). "Heat losses in ATES systems: The impact of processes, storage geometry and temperature". In: *Geothermics* 117, p. 102889. ISSN: 0375-6505. DOI: https://doi.org/10.1016/j.geothermics.2023.102889. URL: https://www.sciencedirect.com/science/article/pii/S0375650523002444.

Buscheck, Thomas A., Christine Doughty, and Chin Fu Tsang (1983). "Prediction and analysis of a field experiment on a multilayered aquifer thermal energy storage system with strong buoyancy flow". In: *Water Resources Research* 19.5, pp. 1307–1315. ISSN: 00431397. DOI: 10.1029/WR019i005p01307.

DHI-WASY GmbH (2012). *FEFLOW finite element subsurface flow and transport simulation system – Users manual/Reference manual/White papers. v. 6.1.* Tech. rep. DHI A/S.

Drijver, Benno, Guido Bakema, and Peter Oerlemans (2019). "State of the art of HT-ATES in The Netherlands". In: *European Geothermal Congress: Proceedings, Den Haag, Netherlands*.

Federal Ministry for Economic Affairs and Climate Action (2024). *Unterirdischer Speicher sorgt für klimafreundliche Wärme*. URL: https://www.energieforschung.de/de/aktuelles/projekteinblicke/2023/unterirdischer-speicher-sorgt-fuer-klimafreundliche-waerme (visited on 2025-02-27).

Ferziger, Joel H., Milovan Perić, and Robert L. Street (2020). "Finite-Volumen-Methoden". In: *Numerische Strömungsmechanik*. Ed. by Joel H. Ferziger, Milovan Perić, and Robert L. Street. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 91–124. ISBN: 978-3-662-46543-1. DOI: 10.1007/978-3-662-46544-8{\textunderscore}4.

Fleuchaus, Paul (2020). "Global application, performance and risk analysis of Aquifer Thermal Energy Storage (ATES)". PhD thesis. Karlsruhe: Karlsruhe Institute of Technologie. DOI: 10.5445/IR/1000120042. URL: https://scholar.archive.org/work/5ea34jntpnbujilx4574rmapce/access/wayback/https://publikationen.bibliothek.kit.edu/1000120042/77340903.

Fleuchaus, Paul et al. (2021). "Aquiferspeicher in Deutschland". In: *Grundwasser* 26.2, pp. 123–134. ISSN: 1430-483X. DOI: 10.1007/s00767-021-00478-y.

Forschungsinitiative Green Energy Lab (2022). *ATES Vienna - Integrating aquifer thermal energy storage into district heating networks to achieve complete decarbonisation*. URL: https://greenenergylab.at/projects/ates-vienna/?lang=en (visited on 2025-02-27).

Gao, H. et al. (2024). "Estimation of Recovery Efficiency in High–Temperature Aquifer Thermal Energy Storage Considering Buoyancy Flow". In: *Water Resources Research* 60.11. ISSN: 1944-7973. DOI: 10.1029/2024WR037491.

Gillner, Markus et al. (2025). "Status of the transient library: Transient simulation of complex integrated energy systems". In: *Modelica Conferences*.

Guelpa, Elisa and Vittorio Verda (2019). "Thermal energy storage in district heating and cooling systems: A review". In: *Applied Energy* 252, p. 113474. ISSN: 03062619. DOI: 10.1016/j.apenergy.2019.113474.

Hamburg Insitut (2022). *Forschungsvorhaben "Optimierte Integration Thermischer Aquiferspeicher in Fernwäremsysteme" (OptInAquiFer)*. URL: https://www.hamburg-institut.com/projects/forschungsvorhaben-optimierte-integration-thermischer-aquiferspeicher-in-fernwaermesysteme-optinaquifer/ (visited on 2025-02-27).

Heldt, Stefan, Christof Beyer, and Sebastian Bauer (2024). "Uncertainty assessment of thermal recovery and subsurface temperature changes induced by high-temperature aquifer thermal energy storage (HT-ATES): a case study". In: *Geothermics* 122, p. 103086.

Kipp, Kenneth L. (1987). *HST3D: A Computer Code for Simulation of Heat and Solute Transport in Thre-Dimensional Ground-Water Flow Systems*. U.S. Geological Survey.

Köhler, Benjamin et al. (2025-02). "Optimierte Integration thermischer Aquiferspeicher in Fernwärmesysteme". In: *3R Rohre*, pp. 84–91.

Kolditz, O. et al. (2012). "OpenGeoSys: an open-source initiative for numerical simulation of thermo-hydro-mechanical/chemical (THM/C) processes in porous media". In: *Environmental Earth Sciences* 67.2, pp. 589–599. ISSN: 1866-6280. DOI: 10.1007/s12665-012-1546-x.

Langevin, Christian D. et al. (2017). *Documentation for the MODFLOW 6 Groundwater Flow Model*. Tech. rep. U.S. Geological Survey. DOI: 10.3133/tm6A55.

Lopik, Jan H van, Niels Hartog, and Willem Jan Zaadnoordijk (2016). "The use of salinity contrast for density difference compensation to improve the thermal recovery efficiency in high-temperature aquifer thermal energy storage systems". In: *Hydrogeology Journal* 24.5, p. 1255.

Maccarini, Alessandro et al. (2023). "Low-order aquifer thermal energy storage model for geothermal system simulation". In: *Linköping Electronic Conference Proceedings*. Linköping University Electronic Press, pp. 389–396. ISBN: 1650-3686. DOI: 10.3384/ecp204389.

Molz, F. J. et al. (1983). "Aquifer thermal energy storage : A well doublet experiment at increased temperatures". In: *Water Resources Research* 19.1, pp. 149–160. ISSN: 00431397. DOI: 10.1029/WR019i001p00149.

Nield, Donald A. and Adrian Bejan (2013). *Convection in Porous Media*. New York, NY: Springer New York. ISBN: 978-1-4614-5540-0. DOI: 10.1007/978-1-4614-5541-7.

Nordell, Bo, Art Snijders, and Lynn Stiles (2021). "The use of aquifers as thermal energy storage systems". In: *Advances in thermal energy storage systems*. Ed. by Luisa F. Cabeza. Woodhead publishing series in energy. Duxford, United Kingdom and Cambridge, MA: Woodhead Publishing, pp. 111–138. ISBN: 0128198885. DOI: 10.1016/B978-0-12-819885-8.00005-X.

Push it (2024). *Delft (Netherland) - Demo site for Aquifer Thermal Energy Storage (ATES)*. URL: https://www.push-it-thermalstorage.eu/pilots/delft/ (visited on 2025-02-27).

Schmidt, Thomas (2005). *Aquifer Thermal Energy Storage - TRNAST Two Well Model for TRNSYS*. Tech. rep. Stuttgart: Solites - Steinbeis Research Institute for Solar and Sustainable Thermal Energy System.

Senkel, Anne et al. (2021). "Status of the TransiEnt library: Transient simulation of complex integrated energy systems". In: *Modelica Conferences*, pp. 187–196.

Umweltbundesamt (2024). *Renewable energies in figures*. URL: https://www.umweltbundesamt.de/en/topics/climate-energy/renewable-energies/renewable-energies-in-figures (visited on 2025-02-27).

Vojacek, Ales et al. (2023). "Status of the ClaRa Library: detailed transient simulation of complex energy systems". In: *Modelica Conferences*, pp. 617–626.

TAeZoSysPro: A Modelica Library for Thermal Aeraulic and Buildings Thermodynamics Calculations

Pascal BOREL[1] Rafik MOULOUEL[1] Felix MARSOLLIER[2] Antoine CHUPIN[2]

[1]EDF R&D, France, pascal.borel@edf.fr, rafik.moulouel@edf.fr
[2]SIL3X, France, felix.marsollier@sil3x.fr, antoine.chupin@sil3x.fr

Abstract

TAeZoSysPro is a Modelica library developed for thermal and fluid dynamics simulations in industrial applications. The library provides components and models for heat transfer, fluid flow, and mass-energy release calculations. It includes modules for heat transfer modeling, fluid dynamics phenomena, specialized media, and partial differential equations. This paper presents the motivations behind developing this specific library and demonstrates its capabilities through industrial use cases, particularly for applications involving ventilation systems, equipment thermal behavior, and accident scenarios.

Keywords: fluid dynamics, heat transfer, Modelica, simulation, buildings thermodynamics

1 Introduction

The development of industrial systems relies increasingly on automated electrical equipment. In modern facility design, particular attention is paid to installation constraints, which has led to a higher concentration of thermal dissipation. This has highlighted new challenges related to thermal transients, especially in the event of ventilation loss.

Beyond these concerns, the TAeZoSysPro MODELICA library addresses another critical aspect: the impact of transients such as water/steam circuit breaches and boiling transients. These phenomena require detailed evaporation-condensation calculations and flexible modeling tools to effectively simulate installations, analyze transient behaviors, and perform sensitivity studies. For instance, (R. Moulouel et al. 2023) performed a sensitivity analysis on a nodal model to better understand how certain parameters influence the air temperature in an electrical room.

In the field of thermal modeling of buildings and industrial facilities, several Modelica libraries already exist, notably Buildings (Lawrence Berkeley National Laboratory 2025) and BuildSysPro (EDF R&D 2022). These libraries are primarily primarily designed for commercial building thermal systems. TAeZoSysPro positions itself as complementary by focusing on industrial thermal aspects and fine modeling of fluid dynamics phenomena. The library is compatible with the MODELICA Standard Library (Modelica Association 2021) interfaces and existing solutions while providing specific functionalities for industrial applications.

Recent studies (R. Moulouel et al. 2023) (Beiza et al. 2014) have emphasized the importance of accurately modeling thermal phenomena in industrial installations, particularly for predicting the behavior of temperature-sensitive equipment. However, existing libraries often lack specialized components that can address the complex thermal and fluid interactions typical of such environments.

The development of TAeZoSysPro was guided by several key requirements: accurate thermal modeling of industrial equipment such as electrical cabinets and transformers; implementation of the CARROLL method for improved separation of convective and radiative heat transfers; support for bi-species fluids, including handling of null species fractions and condensation phenomena; the capability to simulate transient thermal scenarios, such as ventilation loss; and full compatibility with existing Modelica libraries and the MODELICA Standard Library interfaces.

The present paper aims to provide a general overview of the TAeZoSyPro Modelica library which is available within the `OpenMODELICA` library manager or at https://gitlab.pam-retd.fr/TAeZoSysPro with detailed `html` description for each components. It first presents the theoretical foundations of the library, organized into two main subpackages: thermal heat transfer and fluid dynamics. For each subpackage, the main modeling assumptions are described and discussed in terms of their implications and limitations. Key components are illustrated to convey the overall modeling philosophy. Finally, typical application examples are presented, including comparisons with experimental results when available.

2 Library Overview

TAeZoSysPro is organized into four main modules as illustrated on Figure 1 and detailed further in the following subsections.

2.1 Heat Transfer

This module provides components for modeling heat transfer through conduction, convection, and radiation. It includes models for walls, equipment, and thermal exchanges with the environment. Built on a nodal approach, the module enables a detailed and flexible representation

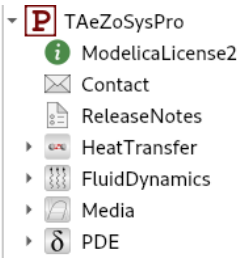


Figure 1. TAeZoSysPro library general architecture

of thermal phenomena.

2.2 Fluid Dynamics

This module handles air flows and mass-energy transfers, enabling the simulation of evaporation-condensation phenomena and multi-species flows. A key feature is the ability to model phase changes, which are essential for capturing a wide range of industrial thermal processes.

2.3 Media

The Media module defines the thermodynamic properties of fluids used in the library, with particular attention to bi-species mixtures and condensation phenomena. The module implements specialized materials whose properties vary with temperature, pressure, and composition. Unlike the standard Modelica Media library, TAeZoSysPro media can handle null species fractions, which is crucial for certain industrial scenarios such as pool boiling, where non-condensable species (e.g., air) are initially present and gradually replaced by steam during the boiling transient.

2.4 PDE

This module provides tools for solving partial differential equations (PDEs), enabling detailed modeling of physical phenomena. It supports spatial discretization schemes tailored to specific types of transport processes and is designed to be used as a subcomponent within more complex systems. The module includes a first-order upwind 1D spatial discretization scheme for purely advective transport, and a second-order centered 1D spatial discretization scheme for diffusive transport. Temporal discretization is handled directly by the Modelica solver used in the simulation environment. Temporal discretization scheme is directly handled by used Modelica solver. The module solves advection-dominated transport equations of the form:

$$\alpha \partial_t Y + \beta \partial_x Y = \dot{\omega} \tag{1}$$

Here, α and β are constants across the spatial domain, specified by the modeler. $\dot{\omega}$ is a source term that can vary from node to node, and Y represents the transported quantity (e.g., enthalpy). Typically, $\alpha = \rho$ and $\beta = \rho U_b$ for a dynamic pipe where U_b stands for the bulk velocity within the pipe. The module also supports diffusion-dominated transport described by:

$$\alpha \partial_t Y = \gamma \partial_{x^2} Y + \dot{\omega} \tag{2}$$

In this case, γ is the diffusion coefficient. The same conventions apply for α, $\dot{\omega}$, and Y as in the advection case.

3 General framework

All physics model embedded in the library are based on the three fundamentals balance equations of the Navier-Stokes equations which are conservation of mass, conservation of motion and conservation of energy. The first rely on mass conservation, the second is establish using the Newton second law while the latter is established with the first principle of thermodynamics:

$$\partial_t \rho + \underline{\nabla} \cdot (\rho \underline{u}) = 0 \tag{3}$$

$$\partial_t (\rho \underline{u}) + \underline{\nabla} \cdot (\rho \underline{u} \otimes \underline{u}) = -\underline{\nabla} p + \underline{\nabla} \cdot \underline{\underline{\tau}} + \rho \underline{g} \tag{4}$$

$$\partial_t (\rho e_t) + \underline{\nabla} \cdot (\rho \underline{u} e_t) = \underline{\nabla} \cdot \left(\underline{u} \cdot \underline{\underline{\sigma}} \right) + \rho \underline{u} \cdot \underline{g} - \underline{\nabla} \cdot \underline{q} + \dot{\omega} \tag{5}$$

where ρ is the density, $\underline{u}$ the velocity, μ the kinematic viscosity, p the pressure, $\underline{g}$ the gravity, $\underline{\underline{\sigma}}$ the stress tensor, e_t the total energy, q the heat flux, $\dot{\omega}$ the volumetric heat gain. The stress tensor can be decomposed as follow:

$$\underline{\underline{\sigma}} = \underline{\underline{\tau}} - p \underline{\underline{1}} \tag{6}$$

where $\underline{\underline{\tau}}$ is the viscous stress tensor. The total energy e_t can be decomposed as the sum of the internal energy and the kinetic energy.

$$e_t = e + \frac{1}{2} u^2 \tag{7}$$

In zonal approaches, local equations previously defined are generally integrated over a large volume of control defined by physical boundary such as a whole pipe, a room or a vessel. Effects of unresolved small scales needs then to be adequately accounted at the resolved scales level, usually relying on relevant correlations. The following subsection presents how local balances are integrated and solved within the TAeZoSysPro modelica library.

3.1 Heta transfer sub-package approach

Objective of this sub-package is to solve the energy equation 5 integrated over the whole volume of components used to model the studied system. A balance of kinetic energy is performed by multiplying Equation 4 on both side by $\underline{u}$ and substitute to Equation 5 leading to:

$$\partial_t (\rho e) + \underline{\nabla} \cdot (\rho \underline{u} e) = -\underline{\nabla} \cdot \underline{q} + \dot{\omega} + \mu S^2 - p \underline{\nabla} \cdot \underline{u} \tag{8}$$

where $S^2 = \underline{\underline{\tau}} : \underline{\underline{\tau}}$ is dissipated kinetic energy power due to fluid viscosity which is generally neglected. The enthalpy $h = e + p/\rho$ is introduced in Equation 8. Assuming constant pressure transformation, incompressible flows and constant specific heat c_p, enthalpy can be expressed as $h = c_p T$ and Equation 8 reads:

$$c_p \partial_t \rho T + c_p \underline{\nabla} (\rho \underline{u} T) = -\underline{\nabla} \cdot q + \dot{\omega} \tag{9}$$

DOI
10.3384/ecp218293

Integration of this latter equation over a volume of control, denoted c, whose mean temperature is defined as:

$$T_c = \frac{1}{m_c} \iiint_V \rho T dV \tag{10}$$

leads to:

$$V_c \rho_c \partial_t T_c - \dot{W} = \sum_i \dot{m}_i c_p (T_i - T_c) + \sum_j \Phi_j \tag{11}$$

where Φ_j is the thermal flux at the boundary j of the control volume, $\dot{m}_i$ the mass flux at the boudary i and $\dot{W}$ the total heat gain at the component level volume which can eventually account for thermal radiant transfers absorption. For each boundary with inlet mass flux, flow being assumed incompressible, a boundary with exactly the mass flux exists assuming leaving temperature is the mean component temperature. Within the thermal sub-package, mass fluxes are considered as model parameters that needs to be provided by the modeler, the only state variable of models only constructed with this sub-package is thus the temperature T.

Elementary components assembly rely on a staggered scheme, as illustrated on Figure 2, where state variables are computed within capicity components, the nodes, with time derivative ∂_t and fluxes between inertial components are computed within fluxes components based on connected state variables values. Typically, left hand side of Equation 11 is handled in capacity components while right hand side is handled within flux components.

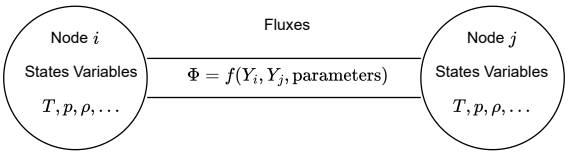


Figure 2. Staggered scheme between nodes

Thermal fluxes at component boundaries can involve radiant, convective, or conductive transfers. Convective heat transfers at a solid-fluid boundary refer to the conductive heat fluxes at the wall, which are then carried away by the fluid motion within the boundary layer. Conductive heat fluxes are accounted using Fourrier's law:

$$\phi_{cond} = -\lambda \underline{\nabla} T \tag{12}$$

where λ is the thermal conductivity. Convective fluxes are estimated using Newton's cooling law:

$$\phi_{cv} = h_{cv}(T_w - T_c) \tag{13}$$

where T_w is the wall surface temperature and h_{cv} is the heat transfer coefficient which have to either provided by the modeler or estimated using correlation in the typical form $Nu = f(Ra, Pr)$ with Nu being the Nusselt number, Ra the Rayleigh number and Pr the Prandtl number.

Radiant heat fluxes between surfaces can be either modeled with radiant heat exchange coefficient h_r to be added to h_{cv} or with a fictive mean radiant node, denoted the Carroll node, as exposed by (Carroll 1980). In this context, the radiosity of the Carroll node is:

$$J_{MRT} = \frac{\sum_k F_k J_k S_k}{\sum_k F_k S_k} \tag{14}$$

where k relates to the k^{th} opaque surface involved with S_k surface, J is the radiosity and F_k is the Carroll surface viewing factor. This latter factor is computed using the following heuristic equation:

$$F_i = \frac{1}{1 - \frac{S_i F_i}{\sum_k F_k S_k}} \tag{15}$$

The net radiant fluxes for each involved opaque surface is then estimated as follow:

$$\phi_{rad} = F_i' \left(J_{MRT} - \sigma T_i^4 \right) \tag{16}$$

where σ is the Stefan-Boltzmann constant and F' the viewing factor modified to account emissivity of surface.

$$\frac{1}{F_i'} = \left(\frac{1 - \varepsilon_i}{\varepsilon_i} + \frac{1}{F_i} \right) \tag{17}$$

This approach is applied to each components, such air volume control, slice of walls or isothermal mass, for each them boundary fluxes needs to be modeled with relevant approaches to account thermal conductive fluxes within the unresolved boundary layer.

3.2 Fluid dynamics sub-package approach

Fluid dynamics aims to solve the three fundamental balance equations—mass conservation (3), momentum conservation (4), and energy conservation (5)—at an integral scale level by integrating them at the level of macroscopic components with physical boundaries.

A staggered approach, depicted in Figure 2, is also applied within this module. The node modeled within this package can handle mixtures of species, including possible condensation.

Mass conservation is solved for each species, while global equations for the conservation of momentum and energy are solved. Equations of state for each species enable the system to be closed. In defining the number of equations solved, the equations of state are usually not counted. Therefore, the Fluid Dynamics Package is commonly referred to as a model with four equations in the case of humid air, which is the primary focus of this module.

Integration of mass conservation for each species leads to:

$$\partial_t m_i = \sum_j \dot{m}_i + \Gamma_i \tag{18}$$

where $\dot{m}_i$ are the mass fluxes for the species $_i$, Γ_i is the mass generation (can be negative) of a given species which

can be due to phase change leading to drain vapor from humid air because of condensation for instance. Within each node, mean temperature T_c is assumed to be the same for each species while pressure and density is computed as follow:

$$p = \sum_i p_i \quad \rho = \sum_i \rho_i \quad e = \sum_i X_i e_i \quad X_i = \rho_i / \sum_i \rho_i \tag{19}$$

where X_i is the mass fraction of species i.

Mass fluxes at fluid interfaces are estimated by assuming a ratio between the Nusselt number and the Sherwood number close to one ($Nu/Sh \approx 1$). Additionally, the Lewis number ($Le = \alpha/D$), where α is the thermal diffusivity and D is the mass diffusivity, is also assumed to be close to one. This allows for the definition of the mass transfer coefficient and the associated mass flux as follows:

$$h_m = \frac{h_{cv}}{c_p} \tag{20}$$

$$\phi_m = h_m (X_w - X_c) \tag{21}$$

where X_w is the mass fraction of the species at the wall, which is assumed to be equal to the saturated mass fraction as a maximum.

Conservation of momentum energy is integrated between node, within equivalent pipes or duct of section S using Bernouilli's theorem which can be derived from Equation 4 for incompressible and irrotational flow. The fluid head is then defined as:

$$H = p + \frac{1}{2}\rho u^2 + \rho g \tag{22}$$

Within an equivalent pipe the integral solve equation is:

$$\underbrace{S_p L_p \rho_m \partial_t u_b}_{\text{unsteady term}} + \underbrace{S_p \left[(p_o + \rho_m g z_o) - (p_i + \rho_m g z_i) \right]}_{\text{head flux integral balance}} = \underbrace{S_p \Delta H}_{\text{singular and regular head losses}} \tag{23}$$

where u_b is the bulk velocity, ρ_m is the mean fluid density within pipe, z the altimetry, L_p its length and S_p its cross section surface. $_i$ stands for pipe inlet and $_o$ for pipe outlet. S_p is assumed to be the same at pipe inlet and outlet, implying and equal dynamic pressure at the inlet and the oulet, however singular head losses due to singularities such as pipe diameter change can be accounted in the singular head losses term. The flow within pipes is assumed to be quasi static regarding mass balance equation, the mass flux within pipe is thus:

$$\dot{m}_p = u_b \rho_m S_p \tag{24}$$

Within each node, only the total pressure defined as $p_t = p + 1/2\rho u^2$ and the total energy are used within balance equation.

The energy conservation equation 5 is integrated at the node level neglecting the work of the viscous forces at the boundaries $\underline{\nabla} \cdot \underline{\underline{\tau}}$ and the work of the volumes forces $\rho \underline{u} \cdot \underline{g}$. One can notice that work of the pressure force can be combined with the advection of internal energy:

$$\underline{\nabla} \cdot (\rho \underline{u} e) + \underline{\nabla} \cdot \left(\underline{u} \cdot p \underline{\underline{1}} \right) = \underline{\nabla} \cdot (\rho \underline{u} h) \tag{25}$$

where $h = e + p/\rho$ is the enthalpy. At the node level, the resulting balance is:

$$V_c \partial (\rho_c e_{c,t}) - \dot{W} = \underbrace{\sum_i \dot{m}_i h_i}_{\text{enthalpy fluxes}} + \underbrace{\sum_j \Phi_j}_{\text{heat fluxes}} \tag{26}$$

3.3 Moist air media

Theory presented in §3.1 and §3.2 is valid for any media within the volume control. Thermodynamics properties are computed separately within the TAeZoSysPro library to enable the use of the `Media` packages of the `Modelica Standard Library` if needed. The library has a specific media dedicated to moist air, with a formulation based on temperature and density (ρ, T) as explicit variables to handles potential total disappearance of one species while still being numerically stable. The enthalpy of the fluid is estimated as the ratio of the different mass fraction X:

$$\begin{aligned} h = & X_a \underbrace{c_{p,a}(T - T_{ref})}_{\text{dry air enthalpy}} \\ & + X_v \underbrace{\left[c_{p,wv}(T - T_{ref}) + h_{vl}(T_{ref}) \right]}_{\text{vapour enthalpy}} \\ & + X_l \underbrace{c_{p,wl}(T - T_{ref})}_{\text{liquid water enthalpy}} \end{aligned} \tag{27}$$

where $h_{vl}(T_{ref})$ is the latent heat of water at the reference temperature. The mass fraction of liquid water within a node is computed as a function of the saturation density $\rho_{sat} = f(T)$ ensuring ρ_{wv} is below the saturation density. The fraction of liquid water within a gas node is expected to be small but possible, for instance, in the case of fog. Specific nodes for this medium are available in the library to manage the drainage of liquid from the node. This is based on the fall speed of water droplets and the a priori number of droplets per unit of volume.

4 Heat Transfer Details

4.1 Bases classes subpackage

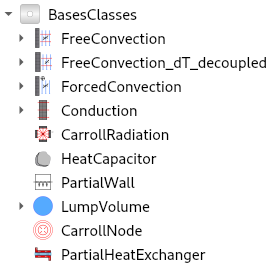


Figure 3. BasesClasses TAeZoSysPro subpackage overview

DOI 10.3384/ecp218293

The BasesClasses, illustrated in Figure 3, subpackage includes elementary functions that can be used directly within a model or serve as building blocks for more complex macro components.

Ports used by components within this package involve only on one state variable declared as follow:

Listing 1. Thermal port for heat transfer appoach

```
within TAeZoSysPro.HeatTransfer.Interfaces;

partial connector HeatPort "Thermal port
    for 1-dim. heat transfer"
  Modelica.SIunits.Temperature T "Port
      temperature";
  flow Modelica.SIunits.HeatFlowRate Q_flow
```

These ports are fully compatible with the thermal port defined in the Modelica Standard Library as well as in BuildSysPro (EDF R&D 2022)

4.1.1 Convection

Several correlations are available to handle various free and forced convection scenarios encountered in industrial installations, including vertical walls, ceilings, ground surfaces, and cylinders. These correlations are used to compute the Nusselt number (Nu), from which the convection coefficient is determined using the relation $h_{cv} = Nu\lambda/L_c$. Alternatively, this coefficient can also be directly specified by the user. The heat convection flux is then computed using Equation 13.

4.1.2 Conduction

For conduction through a homogeneous material, the heat conduction flux modeled using Fourier's law (Equation 12). In practice, the PDE module is typically used to solve the heat equation for a wall with a given thermal conductivity by discretizing the wall into multiple layers or nodes.

4.1.3 Carroll Radiation

This component implements the Carroll radiation method, as detailed in subsection 3.1. It is associated with each radiative surface, for which the area and emissivity are specified, and is connected to the CarrollNode. Additionally, it interfaces with the FviewCalculator, which computes and supplies the Carroll view factor, accounting for the configuration of all radiative surfaces.

4.1.4 Elementary components

In addition to the previously introduced functions for handling various modes of heat transfer, the BasesClasses subpackage includes elementary components such as the HeatCapacitor, which is characterized by a given mass and thermal capacitance and allows for the supply of a heat flux. The LumpVolume component models the thermal inertia of an open lumped volume by linking variations in potential enthalpy (or temperature for an ideal gas) to the heat flow balance. It operates under the assumption of constant pressure and accounts for isobaric expansion, as the volume boundaries are not sealed.

4.2 Main Components

The library provides several basic components for thermal and basic HVAC modeling:

- Wall: models walls with consideration of thermal inertia and homogeneous material (an special multilayer wall model is also available). The wall model solves the heat equation with appropriate boundary conditions using the PDE module assuming 1-D thermal conduction:
- InertMass: represents equipment thermal masses with a lumped thermal capacity:

$$C \cdot \frac{dT}{dt} = \sum \dot{Q}_{in} - \sum \dot{Q}_{out} \tag{28}$$

  where C is the thermal capacity, T is the temperature, and $\dot{Q}_{in}$ and $\dot{Q}_{out}$ are the incoming and outgoing heat flows.
- HeatExchanger: models heat exchangers using an effectiveness approach:

$$\dot{Q} = \varepsilon \cdot C_{min} \cdot (T_{h,in} - T_{c,in}) \tag{29}$$

  where ε is the effectiveness, C_{min} is the minimum heat capacity rate, and $T_{h,in}$ and $T_{c,in}$ are the inlet temperatures of the hot and cold fluids.
- Ventilation and FanVentilation: simulate ventilation systems with variable flow rates and temperature influences.
- FviewCalculator: implements the CARROLL method to calculate radiative temperatures and view factors for radiative exchanges. This component is a key element in the library's approach to radiation modeling. It computes view factors between surfaces without requiring detailed geometric information, using only the surface areas as inputs.
  The CARROLL method creates a virtual node (represented by the CarrollNode component) that acts as a fictitious blackbody enclosure exchanging radiation with all surfaces. This node's temperature corresponds to the Mean Radiant Temperature (MRT).
- CabinetPower: a significant example of a macro component is the model, which represents an electrical cabinet with its thermal dissipations. Having a RealInput port connection for the heat power allows for a fine control of the thermal power dissipated by the cabinet, including transient behaviors.
  The model solves a system of equations for each thermal node representing the emitter inside the cabinet and the casing of the cabinet. The heatload are dissipated within the emitter. The emitter exchanges by convection with the fluid between the emitter and the casing. Regarding the convection, the emitter is supposed flat and horizontal or vertical depending on the value of correlation_internal. The convective surface of the emitter ($A_conv_emitter$) has thus

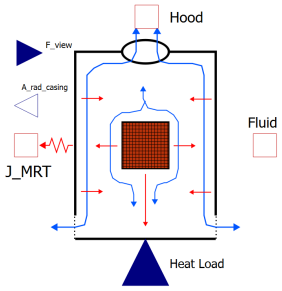


Figure 4. Icon view of the CabinetPower model

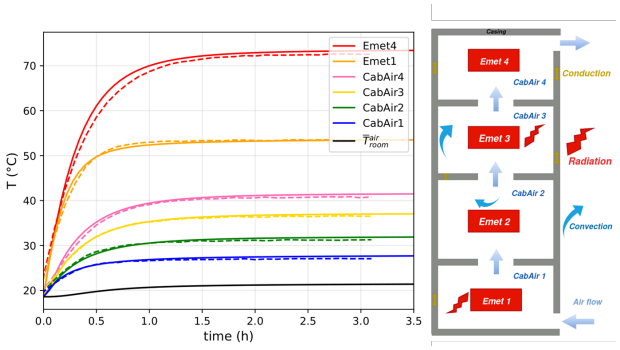


Figure 5. Comparison of the Zephyr electrical cabinet model with experimental data

to be an equivalent surface is both vertical and horizontal surfaces are present. For the casing for the inner and outer convection, the correlation is always a correlation for a vertical plate.

Based on the `CabinetPower` model, a more detailed representation can be developed to investigate various physical phenomena. For instance, the model of the electrical cabinet room (cf. Figure 6) reproduces the actual experimental setup: a simple parallelepiped cabinet with four vertically stacked drawers. Each drawer is represented as an enclosing structure containing a mass-based heat emitter. These emitters exchange heat via radiation with the cabinet's enclosure and through natural convection with the air inside the drawers. The enclosure, in turn, exchanges heat by convection with the ambient room air and by radiation with Carroll's radiosity node (MRT). Additionally, the cabinet features openings that enable air circulation between the drawers and with the room's aeraulic node. This detailed electrical cabinet model was calibrated and validated using experimental data shown in Figure 5.

4.3 Application Example: Room with Electrical Cabinet

In order to study thermo-aeraulic phenomena in industrial spaces containing thermosensitive electrical equipment, EDF-R&D has developed a dedicated experimental facility within the ZEPHYR laboratory (Rafik Moulouel et al. 2024). This full-scale, instrumented facility, shown in Figure 6 is representative of the electrical or control-

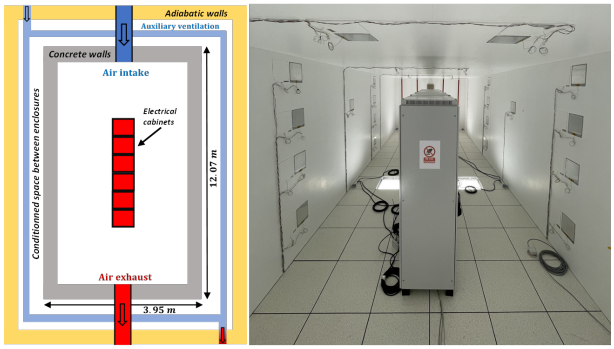


Figure 6. Picture of the electrical cabinets room facility

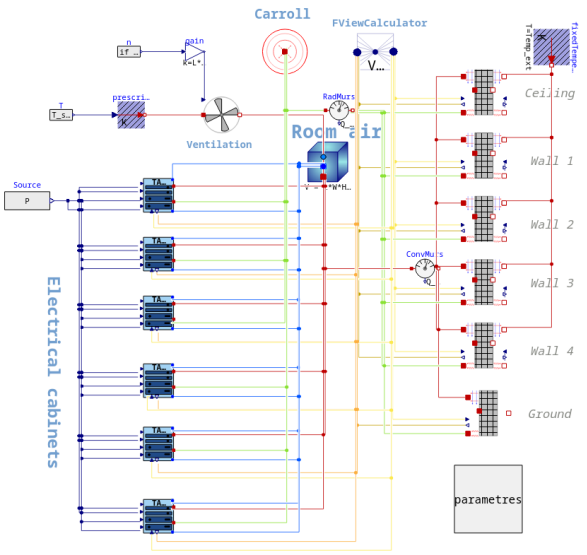


Figure 7. TAeZoSysPro model of the Zephyr electrical room cabinet facility

command rooms found in nuclear power plants. It allows detailed temperature fields measurements to be taken in the air and in the surrounding walls under controlled and repeatable conditions, making it suitable for validating numerical simulation models.

This experimental cell was modeled using the TAeZoSysPro library, as illustrated in Figure 7, to investigate its behavior and perform sensitivity analysis (R. Moulouel et al. 2023). The thermal interactions between the cabinets and their surrounding environment were simulated, along with the air circulation within the various compartments of the cabinets. The influence of the ventilation and the external conditions were studied. Multiple experiments, varying several thermo-aeraulic parameters, were conducted. The results were then compared to both a CFD model and the TAeZoSysPro model, as shown in Figure 8. The 3-D modeling is implemented using the OpenSource CFD solver Code Saturne((Archambeau, Méchitoua, and Sakiz 2004), (EDF 2021)). This comparison demonstrates the TAeZoSysPro library's capability to predict air temperature, although it tends to overestimate it due to its assumption of a homogeneous temperature across the entire room volume, which does not account for thermal strati-

DOI 10.3384/ecp218293

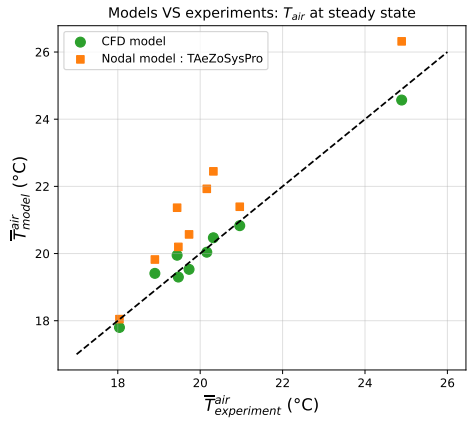


Figure 8. Comparison of TAeZoSysPro thermo-aeraulic model prediction to experimental data

fication. In contrast, the CFD model provides a more accurate prediction of the average air temperature at steady state. These findings will be discussed in further publications.

4.4 Current Limitations

The current approach of separating radiative exchanges presents certain limitations, particularly in the absence of thermal stratification modeling. While the CARROLL method provides advantages for detailed analysis of radiative phenomena, the lack of a stratification model can lead to less accurate results in some scenarios. Developments are ongoing to improve the modeling of these phenomena.

5 Fluid Dynamics Details

The fluid dynamics package is organized in a similar way as the heat transfer package described in §4 and illustrated on Figure 9. Each node is provided

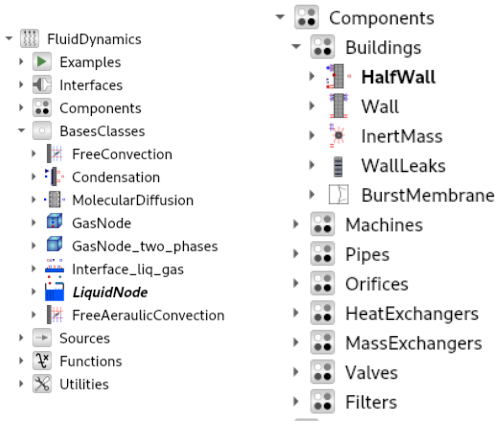


Figure 9. Overview of the fluid dynamics module

with three `Modelica` ports. One Thermal port presented on Listing 1 to handle heat fluxes, a `flowPort` dedicated to moist air media presented in §3.3 and a `Modelica.Fluid.Interfaces.FluidPort` to ensure compatility with the Modelica Standard Library.

Listing 2. FlowPort port for Fluid Dynamics module

```
within TAeZoSysPro.FluidDynamics.Interfaces
    ;
connector FlowPort "Connector flow port"
  replaceable package Medium = Modelica.
      Media.Interfaces.PartialMedium "
      Medium model" annotation(
    choicesAllMatching = true);
  Medium.Temperature T;
  Medium.Density[Medium.nX] d;
  flow Medium.MassFlowRate[Medium.nX]
      m_flow;
  flow Modelica.SIunits.EnthalpyFlowRate
      H_flow;
```

5.1 Specialized Media

The library implements two specific media models.

- **Moist Air Media** - For mass-energy release calculations ;
- **Air H2 Media** - As an example media to track a specific species.

The moist `MoistAir` is a specialized model using density and temperature as explicit state variables, which handles the phase transition between vapor and liquid water with superior numerical stability. A Generic bi-species mixtures, providing a framework for modeling various fluid combinations is also proposed.

The `MoistAir` model in TAeZoSysPro is formulated with density and temperature (ρ, T) as primary state variables, unlike other implementations that use pressure and temperature or enthalpy and pressure. This explicit formulation offers several significant advantages:

- **Enhanced numerical stability** - Particularly important during phase transitions where pressure may experience discontinuities, while density varies more continuously
- **Better handling of near-saturation states** - The (ρ, T) formulation facilitates calculations in regions near saturation, where property derivatives with respect to pressure can become very large
- **Direct mass conservation calculations** - Mass conservation equations are directly expressed in terms of density, simplifying the differential equations to be solved
- **Smoother phase transitions** - Special numerical treatments manage state transitions without the discontinuities that can lead to simulation failures

A key difference between TAeZoSysPro media and the standard `Modelica.Media.Air.MoistAir` is the ability to handle null species fractions. In the standard MSL implementation, having zero mass fraction of a species can lead to numerical issues in certain calculations. TAeZoSysPro implements robust handling of these edge cases, ensuring continuous simulation even when a species disappears completely due to phase change which can be typically the case in industrial buildings thermodynamics studies.

Additionally, the condensation model in TAeZoSysPro includes special numerical treatments to handle the transition between states smoothly, avoiding the discontinuities that can cause simulation failures in complex systems.

5.2 Main Components

An overview of some of the main components is presented below, though it is not exhaustive. The required media are declared as `Replaceable Package` within each component and subcomponent to allow the modeller to use any medium compatible with the proposed structure of the Modelica Standard Library.

The component `gas_node_two_phases` implements mass and energy conservation equations while allowing for phase changes of condensable species and drain of condensate based on fall velocity of liquid droplets. This component is specific to the TAeZoSysPro MoistAir Media.

The `condensation` elementary component model the mass flow rate and energy transfer induced by the condensation of a condensable species over a colder surface. The component is used on macro components such as wall dedicated to studies with potential condensation. In case of surface temperature below dew point, it is assumed that water vapour density within the boundary layer is equal to the saturation density estimated with the wall temperature. Mass transfer flowrate is computed using mass coefficient defined in §3.2.

`Interface_liq_gas`: This component models a flat interface between liquid and surrounding gas where evaporation-condensation, boiling, radiation, and convection occur. It is designed to work with the TAeZoSysPro moist air media for the gas media and Standard Water of the `MSL` for the liquid media. *Evaporation-condensation* is modelled with a similar approach than condensation on surface. Evaporation occurs is liquid surface saturation density is higher than water vapor density in the connected gas node. *Convection* and *Radiation* are modelled with the approach presented in §3.1 *Boiling* is modelled as homogeneous formation of equal-diameter bubbles that form in the liquid node and escape through the interface The boiling process is modeled thanks to the estimation of the gas fraction within liquid node from the dew and buble enthalpy when the fluid enthalpy is higher than bubble enthalpy but lower than dew point enthalpy:

$$h = (1 - X_g)h_{bubble} + X_g h_{dew} \tag{30}$$

where X_g is the gas mass fraction within the connected liquid node.

The library propose also flow junctions and separations, allowing for complex network modeling. A set of heat exchangers with phase change, which handle energy transfer between fluids while accounting for condensation or evaporation is available, which is particularly valuable for building thermodynamics or cooling coils studies.

A subpackage containing walls and building orifices is dedicated to components needed for building thermodynamics studies, typically involving vapor breaches.

These components are interconnected to model complex fluid systems with multi-species flows and phase changes. The module implements a connector-based approach, where connectors carry information about pressure, mass flow rate, enthalpy, and species composition.

5.3 Applications

Two application cases illustrate the library's capabilities:

5.3.1 Pressure Cooker Model

This example case validates the modeling of evaporation phenomena under pressure:

- Pressure rise through evaporation, following the relationship between temperature and saturation pressure
- Liquid-vapor thermodynamic equilibrium, ensuring consistency between phases
- Pressure regulation through relief valve, modeling the mechanical behavior of the system

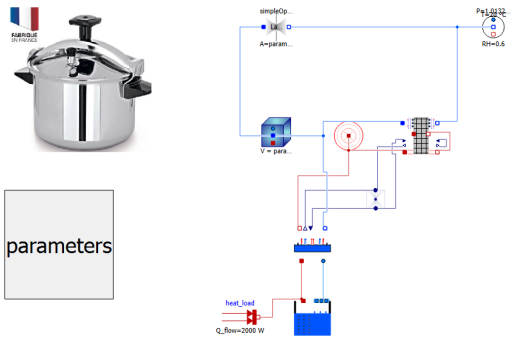


Figure 10. Model of a pressure cooker

The Modelica implementation of this model integrates the following key components:

- A two-phase gas node (`gasNode_two_phases`) representing the steam phase above the liquid
- A liquid node (`liquidNode`) modeling the water volume with thermal inertia
- A liquid-gas interface (`interface_liq_gas`) that handles mass and heat transfer between phases
- A vessel wall (`wall`) with thermal inertia and heat transfer capabilities
- A simple opening component (`simpleOpeningComp`) acting as a pressure relief valve
- A heat source (`heat_load`) providing 2000 W of thermal power to the system
- A Carroll node (`carrollNode`) implementing the CARROLL method for radiative heat transfer calculations

The model uses the `MoistAir` medium from TAeZoSysPro's Media library, which handles the water-vapor mixture with phase change capabilities. Key parameters include the total volume (`V_tot`), gas phase volume

DOI
10.3384/ecp218293

(`V_gas`), wall surface area (`S`), interface area (`A`), and thermal properties such as emissivity (`eps`).

This model demonstrates the library's ability to handle phase changes and pressure dynamics in a closed system. The resulting pressure and temperature curves match experimental data, confirming the accuracy of the phase change calculations.

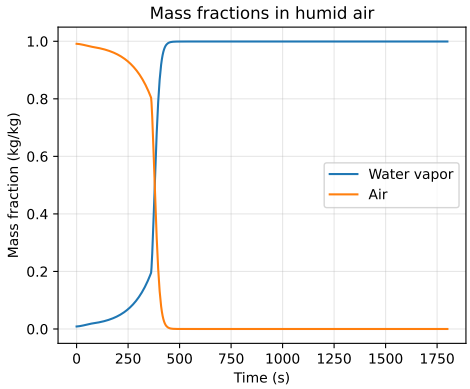


Figure 11. Evolution of mass fractions in the pressure cooker model.

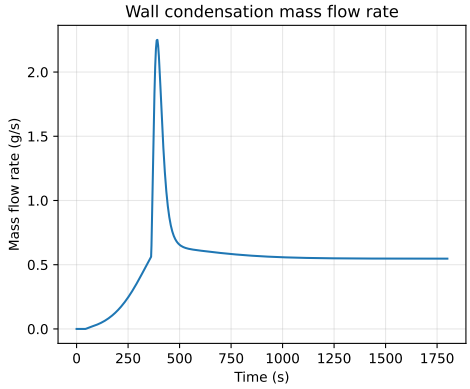


Figure 12. Condensation mass flow rate on the inner wall of the pressure cooker.

The simulation results illustrated in Figures 11 and 12 highlight several key phenomena that occur during the pressure cooker heating process:

- **Phase transition dynamics**: As shown in Figure 11, we observe the gradual evolution of species composition in the gas phase, with a continuous decrease in air mass fraction counterbalanced by an increase in water vapor fraction. This transition demonstrates how the model captures the progressive replacement of incondensable air by water vapor during heating.
- **Condensation phenomena**: Figure 12 illustrates the mass flow rate of condensation occurring at the inner wall of the pressure cooker. The initial peak corresponds to the early heating phase when temperature differences between the vapor and the cooler wall are significant, followed by variations that reflect the complex thermal equilibrium establishment.
- **Multi-phase interactions**: The combined analysis of both figures reveals the interplay between evaporation at the liquid-gas interface and condensation at the cooler walls, showing how the model accurately represents the complete cycle of water phase changes in a confined environment.

These results validate the library's capability to simulate the complex thermodynamic behavior of multi-species fluids undergoing phase changes, a critical feature for industrial applications involving pressurized systems and heat transfer.

5.3.2 Boiling and Steam Line Breaches Transients

A second application case concerns the simulation of transients in nuclear power plant scenarios, including high energy breaks (water/steam line breaches) and pool boiling phenomena (such as in spent fuel pools). These simulations allow for the observation of impacts on pressure and temperature in containment rooms, taking into account various physical phenomena:

- Progressive liquid evaporation and condensation processes, with bidirectional mass transfer between phases
- Thermodynamically-driven phase changes, calculated based on local conditions and interface properties
- Heat transfer through multiple mechanisms (conduction, convection, radiation) affecting phase transition rates and wall temperature
- Pressure increase dynamics resulting from vapor generation and gas compressibility effects
- Spatial distribution of thermal effects and resulting flow patterns

The model Figure 13 captures the complex coupled behavior between thermal, fluid dynamic, and phase change phenomena, providing detailed insights into the evolution of temperature, pressure, humidity, and phase composition during accident scenarios. This capability is particularly valuable for safety analyses where accurate prediction of containment conditions is essential.

This model demonstrates the library's ability to handle phase changes and pressure dynamics in a closed system. The resulting pressure Figure 14 and temperature curves match experimental data, confirming the accuracy of the phase change calculations. Results have been compared against experimental data including the Carolinas-Virginia Tube Reactor (CVTR) containment response tests (Schmitt, Bingham, and Norberg 1970), which provided valuable benchmark data for evaluating steam condensation models under conditions representative of containment structures in nuclear power plants. The initial comparisons are encouraging and show a reasonable level of agreement and a behaviour representative of the experimental data. Some discrepancies are observed in the pressure curve, which is likely due to the complex geometry of the CVTR containment structure and the associated modeling bias.

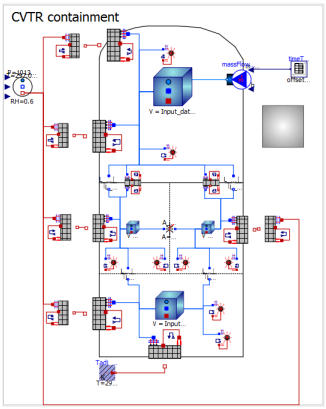


Figure 13. TAeZoSysPro model of the CVTR containment structure.

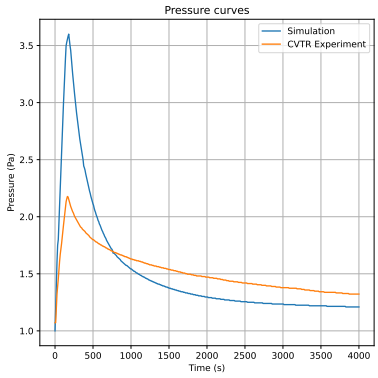


Figure 14. Comparison of pressure evolution between TAeZoSysPro simulation and CVTR experimental data.

6 Conclusion and Perspectives

The TAeZoSysPro library provides an innovative solution for modeling thermal and aeraulic phenomena in industrial installations including buildings thermodynamics. Its strengths include:

- Fine modeling of heat transfers with the CARROLL method and convective heat transfer correlations allowing separating convective and radiative phenomena
- Building Thermodynamics calculations using advanced management of bi-species media with robust handling of null species fractions and condensation
- Experimentally validated components
- Modular architecture, standard interfaces facilitating extensions and coupling with other libraries

Future developments will focus on:

- Improvement of thermal stratification models
- Extension of validation cases
- Qualification of the library according to industrial standards
- Implementation of automatic non-regression tests and MODELICA validation tests allowing the qualification files to be up to date with the latest version of the library
- Extension of multi-scale modeling capabilities

Finally, the modular design of TAeZoSysPro makes it particularly suitable for coupling with uncertainty propagation tools like OpenTURNS (OpenTURNS Team 2023), enhancing its utility for industrial applications where understanding parametric sensitivities is crucial.

Acknowledgments

The authors thank all contributors to the TAeZoSysPro project, as well as the teams who participated in the experimental validation campaigns.

References

Archambeau, Frédéric, Namane Méchitoua, and Marc Sakiz (2004-02). “Code Saturne: A Finite Volume Code for the computation of turbulent incompressible flows - Industrial Applications”. In: *International Journal on Finite Volumes* 1.1. Publisher: Institut de Mathématiques de Marseille, AMU, http://www.latp.univ–mrs.fr/IJFV/spip.php?article3. URL: https://hal.science/hal-01115371.

Beiza, Maximiliano et al. (2014). “Zonal thermal model of the ventilation of underground transformer substations: Development and parametric study”. In: *Applied Thermal Engineering* 62.1, pp. 215–228. ISSN: 1359-4311. DOI: https://doi.org/10.1016/j.applthermaleng.2013.09.032.

Carroll, Joseph A (1980). “An MRT method of computing radiant energy exchange in rooms”. In: *System Simulation and Economic Analysis*, pp. 343–348.

EDF (2021). *Code Saturne*. URL: https://www.code-saturne.org/.

EDF R&D (2022). *BuildSysPro: Bibliothèque Modelica de modélisation énergétique des bâtiments et systèmes thermiques*. URL: https://www.edf.fr/groupe-edf/inventer-lavenir-de-lenergie/rd-un-savoir-faire-mondial/nos-offres/nos-logiciels-et-codes-de-calcul/buildsyspro (visited on 2025-03-17).

Lawrence Berkeley National Laboratory (2025). *Modelica Buildings Library: Open source library for design and operation of building and district energy and control systems*. URL: https://simulationresearch.lbl.gov/modelica/ (visited on 2025-03-17).

Modelica Association (2021). *Modelica Standard Library*. Tech. rep. Version 4.0.0. Modelica Association. URL: https://github.com/modelica/ModelicaStandardLibrary.

Moulouel, R. et al. (2023). “Etude numérique thermo-aéraulique d’un local équipé d’une armoire électrique thermosensible: Analyse de sensibilité”. In: *Congrès Français de Thermique 2023*. Société Française de Thermique. DOI: 10.25855/SFT2023-050.

Moulouel, Rafik et al. (2024). “Measurement of air temperature fields in radiative environments using optical fibers”. In: *Measurement* 230, p. 114512. ISSN: 0263-2241. DOI: https://doi.org/10.1016/j.measurement.2024.114512.

OpenTURNS Team (2023). *OpenTURNS: An Open source initiative for the Treatment of Uncertainties, Risks’N Statistics*. URL: https://openturns.github.io/www/ (visited on 2023-12-20).

Schmitt, R.C., G.E. Bingham, and J.A. Norberg (1970). *CVTR Containment Response Tests*. Tech. rep. Document ID: ML12009A104. U.S. Nuclear Regulatory Commission. URL: https://www.nrc.gov/docs/ML1200/ML12009A104.pdf.

Development of a Refrigerant Mixture Package for Dynamic Simulation of Auto-Cascade Refrigeration: A Case Study with R23/R134a

Rohit Dhumane[1] Dan Gorman[1] Rajkumar K S[2] Dongping Huang[3]

[1]Trane Technologies, USA, {Rohit.Dhumane,Daniel.Gorman}@tranetechnologies.com
[2]Trane Technologies, India, rajkumar.kidaramshanmugam@tranetechnologies.com
[3]Trane Technologies, China, Dongping.Huang@tranetechnologies.com

Abstract

The auto-cascade refrigeration cycle offers higher reliability and lower manufacturing costs compared to the cascade refrigeration cycle, making it particularly attractive for applications involving significant temperature differences between the heat source and sink. However, this advantage comes with increased complexity in its operation. This article aims to develop a media package in Modelica to simulate auto-cascade refrigeration cycles, with the goal of enhancing understanding of their operation and control. The method is demonstrated using an R23/R134a mixture, a commonly used refrigerant combination in such applications. The media package is compatible with the `Modelica Standard Library` and is constructed using curve fits based on the REFPROP dataset. Unlike typical refrigeration systems, the media package for auto-cascade components includes the independent mixture composition (Xi) as an additional variable in function calls. A combination of polynomial and Chebyshev polynomial curve-fits for the refrigerant properties has been shown to provide an optimal balance between accuracy and computational efficiency. The article presents example simulations performed to demonstrate use of the media package with component models from the `Modelica.Fluid` package. A lumped model for the phase separator is developed and simulated with the R23/R134a media to demonstrate ideal phase separation.

Keywords: refrigeration, auto-cascade, two-phase

1 Introduction

Modern and next generation vapor compression systems are increasingly using refrigerant mixtures to address multiple competing design constraints like low global warming potential, improved system performance and costs. Accurate and fast refrigerant property models are essential for model-based design of vapor compression systems, where simulation speed is often bottlenecked by iterative property calculations.

1.1 Auto-Cascade Refrigeration Cycle

The auto-cascade refrigeration cycle (ARC) offers the advantage of achieving a higher temperature difference between the source and sink using a single compressor. This is made possible by employing a zeotropic refrigerant mixture and adjusting its composition to meet operational requirements. However, calculating the thermodynamic properties of zeotropic mixtures is more complex than for pure refrigerants, making the analysis of ARC systems more complicated (He et al. 2022).

1.2 Refrigerant Mixture Properties

Modeling and simulation is an indispensable tool to improve our understanding about complex systems like the ARC. Modelica provides a mature and robust environment for simulating refrigeration systems, while REFPROP (Lemmon et al. 2018) is the industry standard for calculating refrigerant properties. However, the complexity of mixture models in REFPROP can significantly increase the computational cost of simulations. To address this, curve fits derived from REFPROP-generated data are used to reduce computational demands during simulation.

REFPROP is a powerful and widely accepted tool for thermodynamic property calculations, but it presents several limitations when used for dynamic simulations. REFPROP's reliance on nested iterative root-finding algorithms often leads to non-convergence or local numerical anomalies, especially near phase boundaries and critical points. These anomalies manifest as spikes in density derivatives, which can destabilize dynamic simulations or yield inaccurate results (Laughman et al. 2024). Dynamic simulations in Modelica involve frequent property evaluations which can run into millions of calls in a short time window. REFPROP's iterative nature makes these calls computationally expensive. Simulations using REFPROP can be 100–5000 times slower than those using optimized lookup or interpolatory methods (Aute and Radermacher 2014). Additionally, computing thermodynamic properties for these mixtures using first principles is numerically complex and often yields non-physical anomalies, especially in standard tools like REFPROP (Laughman et al. 2024). These anomalies, particularly in the two-phase

region near the critical point can disrupt dynamic simulations by introducing discontinuities in property derivatives, leading to inaccurate or failed predictions. To overcome these limitations, researchers have developed specialized refrigerant property models using spline-based table look-up methods, polynomial fits, and interpolatory techniques (Aute and Radermacher 2014; Laughman et al. 2024; Li et al. 2018).

1.3 Chebyshev Polynomials

Chebyshev polynomials possess several attractive properties when it comes to function approximation. It has been proven that for any continuous function on a given interval, there exists a Chebyshev polynomial that can approximate the said function. These polynomials have been found particularly effective for approximating refrigerant properties, offering a favorable balance between accuracy and computational efficiency (Aute and Radermacher 2014). In this article, we will use Chebyshev polynomials of the first kind T_n which are defined by Equation 1.

$$T_n(cos(\theta)) = cos(n\theta) \tag{1}$$

These are typically calculated with a recurrence equation as shown by Equation 2 - Equation 4.

$$T_0(x) = 1 \tag{2}$$
$$T_1(x) = x \tag{3}$$
$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \tag{4}$$

A polynomial in a single variable, referred to as 1D, can now be calculated with a coefficient vector c_i, which can be calculated from a least squares fit over the data.

$$T_{n+1}(x) = \sum_{i=0}^{n} c_i T_i(x) \tag{5}$$

For the case of polynomials with two variables, separate series for each of the input is created. The orthogonality property of the Chebyshev polynomials is then used to create the 2D equation. The methodology and terminology from the Python's numpy package (Harris et al. 2020) is followed in this article. The reader may refer to its documentation for details.

$$T_{n+1,m+1}(x,y) = \sum_{i=0}^{n}\sum_{j=0}^{m} c_{i,j} T_i(x) T_j(y) \tag{6}$$

Similarly for an equation with three variables, the Equation 7 can be used.

$$T_{n+1,m+1,l+1}(x,y,z) = \sum_{i=0}^{n}\sum_{j=0}^{m}\sum_{k=0}^{l} c_{i,j,k} T_i(x) T_j(y) T_k(z) \tag{7}$$

The ability to create polynomial functions with different maximum degree for the inputs helps to reduce computational cost.

The objective of the article is to present a method to develop a two-phase refrigerant mixture package for zeotropic mixture combinations which are applicable for auto-cascade refrigeration systems. Compatibility with `Modelica Standard Library` Media package is to be maintained to enable reuse of components especially from the `Fluids` package. The method is applied to a binary refrigerant mixture and the required property functions are created in Modelica. The applicability of the autocascade media is demonstrated by simulating a few component level simulations involving change in refrigerant mixture concentration.

2 Fitting Property Data

The entire p-h domain of the zeotropic refrigerant properties domain can be divided into four regions: 1. subcooled, 2. two phase, 3. super-heated and 4. super-critical (Aute and Radermacher 2014). These will be referred to as Regions 1 - 4 henceforth. The state of a binary refrigerant mixture can be defined by three independent properties. In this article pressure (p), specific enthalpy (h) and mixture independent composition (Xi) are used as the independent properties. The properties like density, specific heat capacity, thermal conductivity which cover all the four regions are functions of three variables. The properties on the saturation line like bubble enthalpy, dew enthalpy, bubble density, dew density and so on, require only two independent variables. Finally, critical properties like critical pressure and critical temperature will be fitted to a 1D Chebyshev polynomial.

The method will be demonstrated with a mixture combination of R23-R134a which is commonly used in autocascade refrigeration systems as seen in several publications(Du et al. 2009; Bai, Yan, and Yu 2019; He et al. 2022). The mixture combinations for auto-cascade are such that the boiling points of the constituents are significantly different.

Python is used for the purposes of fitting. Numpy library provides fitting functions for 1D Chebyshev. Sampling the points to generate dataset from REFPROP is performed using Python's Scipy library (Virtanen et al. 2020). Finally, metrics like RMSE are obtained from functions in Scikit-learn (Pedregosa et al. 2011).

2.1 Data Generation

The datasets obtained from REFPROP are sanitized by removing anomalies and replacing them with smooth, physically consistent estimates. This approach enables fast, accurate, and robust simulations in Modelica.

2.1.1 Critical Properties

Critical properties of the zeotropic refrigerant mixture are functions of its mixture composition. These are hence a function of a single independent variable and referred to as 1D functions. For generating 1D dataset for critical properties, 1000 points are sampled within the range 0-1 by latin hypercube sampling (LHS). The critical properties

DOI
10.3384/ecp218303

are calculated by taking this 1D LHS as mass composition of R23.

To fit the data using Chebyshev polynomials, it must first be scaled to the range [-1,1]. This scaling is performed using the `MinMaxScaler` from the Scikit-learn library. The optimal degree of the polynomial is not known a priori, so a parametric study is conducted by varying the degree from 1 to 25. A tolerance value of 10^{-8} is selected for the fitting process.

The results of the parametric study for critical pressure are shown in Figure 1, while results for all properties are summarized in Table 1. A single calculation of the critical pressure is not sufficient to represent the computational expense accurately. Therefore, a random set of 1000 values of Xi is used for benchmarking.

The median computation time to calculate the critical pressure using REFPROP is calculated to be 357.335 ms. This is the median of the same calculation repeated 14 times. The mean was 357.799 ms $\pm$ 4.306 ms. For each polynomial degree between 1 and 25 in the Chebyshev fit, the computational expense for the same evaluation is approximately 30 μs. For these cases, 10000 calculations were used. The standard deviation is of the order 30 μs which contributes to the jagged nature of the plot of computational expense. Nevertheless, it is evident that using Chebyshev polynomials yields a speedup of approximately four orders of magnitude. Similar acceleration is observed for the computation of critical temperature and critical density.

Table 1. Polynomial degree to meet tolerance.

Property	*Degree*
Critical Pressure	20
Critical Temperature	12
Critical Density	24

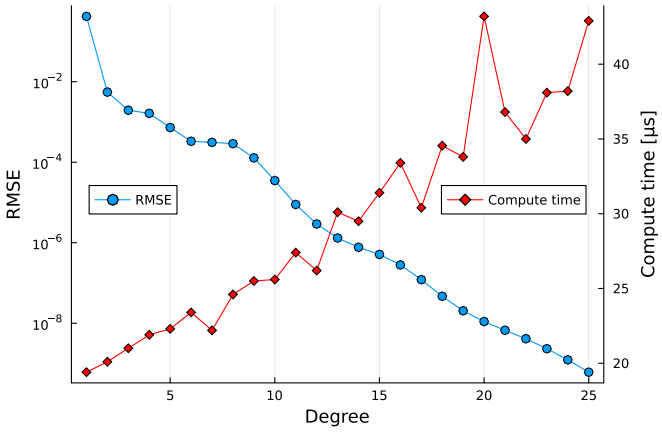


Figure 1. RMSE and compute time for Chebyshev polynomial fit as a function of its degree.

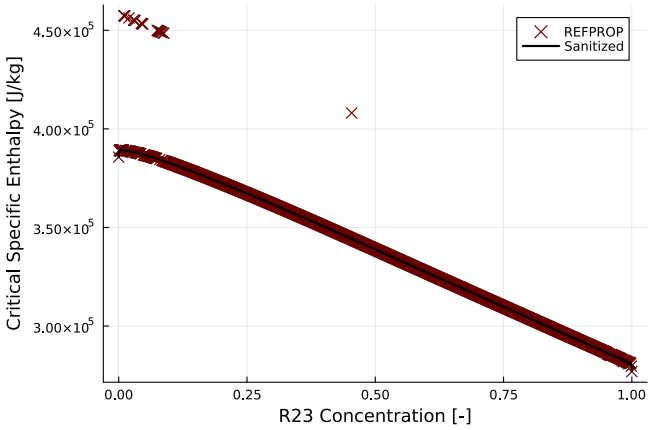


Figure 2. Sanitized values of specific enthalpy at critical points as function of R23 concentration

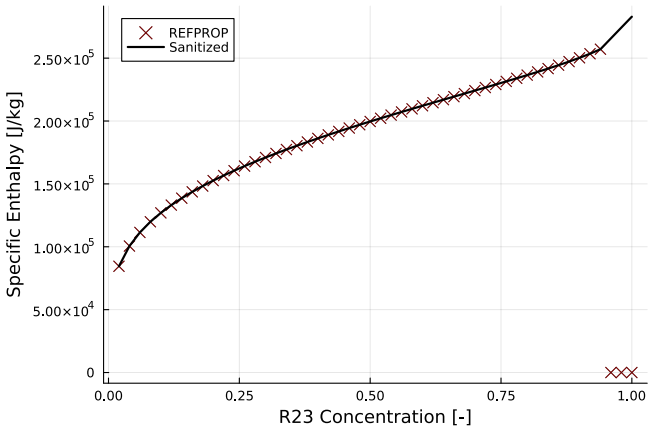


Figure 3. Sanitized values of bubble enthalpy as function of R23 concentration

2.1.2 Saturated Properties

The saturated properties, i.e. the properties which lie on the saturation line, require two independent variables as inputs. These are referred to as 2D properties in this article. Pressure and specific enthalpy are the most commonly used input variables for the properties of this type. To generate the dataset, a Latin Hypercube Sampling (LHS) of two variables of size 100 with values between 0 and 1 is created. Using these two columns of LHS, a total of 10000 is created as a tabular grid. The first variable of the LHS is assigned as the mixture concentration by mass. For pressure, the upper and lower ranges are not constant. This helps with filtering noisy data by observing property trends.

The upper limit is the critical pressure of the refrigerant mixture at that concentration. Lower limit (p_{min}) determination is arbitrary. For the purpose of this article, the bubble pressure corresponding to a temperature of $-80°C$ for the given mixture concentration is taken as the lower limit. The second variable of the LHS is then used to scale the point in between these two limits. This is achieved by applying Equation 8 to each row. x_i is i^{th} sample of the

second variable of the LHS.

$$p_i = p_{min,i} + x_i(p_{crit,i} - p_{min,i}) \tag{8}$$

Now that the pressure and mixture concentration is available, the following refrigerant properties on the vapor dome are calculated: bubble enthalpy, bubble temperature, bubble specific volume, bubble specific heat capacity at constant pressure, bubble thermal conductivity, bubble dynamic viscosity, surface tension, dew enthalpy, dew temperature, dew specific volume, dew specific heat capacity at constant pressure, dew thermal conductivity and dew dynamic viscosity.

As mentioned earlier, data obtained from REFPROP needs to be sanitized. The method used for the current article is described with an example of bubble enthalpy. The bubble enthalpy function depends on two independent variables - reduced pressure and Xi. A latin hypercube design of 100 rows and 2 columns can be used to obtain two vectors varying between 0 and 1. Specific enthalpy of critical points (Hcrit) is a function of Xi. For Xi values starting from 0 to 1 with a step size of 0.001, Hcrit is calculated using REFPROP and shown in Figure 2. It can be observed that there are a few points (22 points out of 1001) which are not following the trend and need to be removed. A simple filter (value > 4e5) is used to eliminate these anomalous values. Linear interpolation is then used to estimate the values for these points to create the sanitized set for critical specific enthalpy.

The linear interpolation of critical specific enthalpy can be used to fill in data by interpolation for some points which fail to compute near the critical point. For the evaluation of 10000 point dataset, a for loop can be constructed which calculates the bubble enthalpy for all the values of reduced pressure at a given value of R23 concentration. For example, for Xi = 0.98 the data is shown in Figure 3. A try-catch block is used and for the failed simulation cases near critical point a value of 0 is provided. These error points can be then easily filtered out (value < 1000). The final point in the interpolation data is for reduced pressure of 1 which corresponds to the critical specific enthalpy for the given Xi. This can be obtained from the interpolation of Figure 2. The removed points can now be added by interpolation since the end point is now available. This is shown in Figure 3. The method is repeated for all the values of Xi to generate the dataset.

Polynomials of degree 16 are used to create curve-fits of the dataset. It is observed that using natural logarithm of the pressure as a feature instead of absolute pressure provides a better fit in some cases. Standard scaling is also used since the two inputs have different orders of magnitude. Several statistics along with visual comparison are used to observe the goodness of the fit. An example plot showing the quality of the fit is shown in Figure 4. In this plot, the values obtained from REFPROP are on the x-axis while curve fits obtained are on the y-axis. Table showing

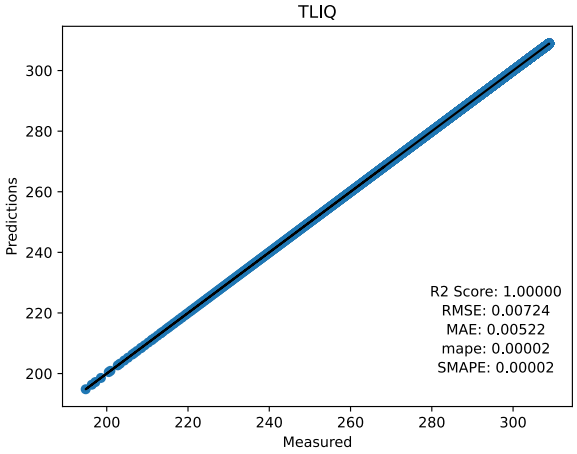


Figure 4. Checking goodness of polynomial fit for bubble temperature

the list of all the 2D properties and their inputs is shown in Table 2.

2.1.3 Regular Properties

The properties of the refrigerant mixture like specific entropy, density, etc. cover all the regions. These properties are referred to as 3D properties in this article. A dataset is created covering 10000 samples in 3 variables in each of the regions.

For the sub-cooled region (Region 1), the mixture composition by mass is obtained directly from one of the three columns of the 3x10000 LHS. Pressure is again obtained using the second variable and the application of Equation 8. To calculate specific enthalpy in the region, the range of enthalpy values and the third variable of the sampling are used. The maximum value of enthalpy possible for the region is the bubble enthalpy h_{bub}. The lower limit for enthalpy h_{min} is the value for which the subcooling is $-50°C$ for the given value of pressure and the mixture concentration. The enthalpy values can now be calculated using an equation similar to Equation 8 for these limits.

For the two phase region (Region 2), the mixture composition and pressure are taken from region 1. The third variable of sampling is used as the refrigerant quality to calculate the enthalpy in the two phase region using the bubble and dew enthalpy.

For the superheated region (Region 3), the mixture composition and pressure are taken from region 1. The minimum value of enthalpy is the dew enthalpy. The maximum value of enthalpy h_{max} is obtained by taking a superheat value of $+50°C$ at the given pressure and mixture composition. An equation of the form Equation 8 for these enthalpy limits is then used to calculate the specific enthalpy in Region 3.

For the supercritical region (Region 4), the mixture composition is obtained from the first variable of LHS. The pressure is obtained by using Equation 9. The maximum pressure is 1.5 times the critical pressure. The en-

DOI
10.3384/ecp218303

thalpy is sampled by calculating temperature in the range of $\pm 50°C$ of the critical temperature. The enthalpy is calculated using the pressure and this sampled temperature.

$$p_i = p_{crit,i} + x_i(p_{max,i} - p_{crit,i}) \tag{9}$$

Now with pressure, specific enthalpy and mixture composition available in each of the regions, temperature, specific entropy, specific volume, specific heat capacity, thermal conductivity, dynamic viscosity and speed of sound are calculated using REFPROP. A regular polynomial of degree 8 is used to create curve-fits for the dataset. Several regions are combined to create a curve-fit if the accuracy is good with combined datasets. However, only specific entropy is observed to have good accuracy (less than 1% mean absolute percentage error) for all four regions with a single 8 degree polynomial. In more typical cases, multiple polynomial fits of degree 8 are created and then a spline interpolation is performed at the border of the regions.

3 Media Class

The `Media` package in `Modelica Standard Library` provides examples for creating media. The refrigerant mixture media is creating by extending the class `Modelica.Media.Interfaces.PartialMedium`. In the extended class the `mediumName` and `substanceName` parameters are defined with string *R23|R134a* and two element string array *"R23","R134a"* respectively.

Since the mixture comprises of multiple components, `singleState` is false. It is preferable to work with only the independent mass fractions of the mixture since they sum to one. For this reason, `reducedX` is set as true. Finally, since mixture composition is expected to change during simulation, `fixedX` is set to false. The `ThermoStates` parameter is set to `phX` to define pressure, specific enthalpy and mixture composition by mass as the three independent variables defining the thermodynamic state of the media. The `BaseProperties` class is defined as shown in Listing 1.

Listing 1. Definition of the base class BaseProperties

```
redeclare replaceable model extends
    BaseProperties(
  h(stateSelect=if preferredMediumStates
      then StateSelect.prefer else
        StateSelect.default),
  p(stateSelect=if preferredMediumStates
      then StateSelect.prefer else
        StateSelect.default),
  Xi(each stateSelect=if
      preferredMediumStates then
      StateSelect.prefer
        else StateSelect.default),
  final standardOrderComponents=true)
  "Base properties of medium"
equation
  d =density_phX(
      p,
      h,
      X);
  T =temperature_phX(
      p,
      h,
      X);
  u = h - p/d;
  MM = 1/(Xi[R23]/MMX[R23] + (1.0 - Xi[
      R23])/MMX[R134a]);
  R_s = Modelica.Constants.R/MM;
  state = setState_phX(p,h,X);
```

Now that the basic structure of the media is setup, the property functions obtained from curve-fits can be added to it.

3.1 1D Functions

The critical pressure, critical temperature and critical density are not constant for the refrigerant mixture. These properties are functions of the mixture composition. The independent composition of the first component (R23 in this case) is referred to as Xi. The composition array of both the components is referred to by the variable X. The functions are defined as a function of X.

The recurrence relationship to calculate the Chebyshev (Equation 5), is coded as a function. This is used to create a Vandermonde matrix. A dot product of the coefficient vector with the Vandermonde matrix gives the value of the output. The implementation is similar to the numpy library (Harris et al. 2020). The inputs and outputs for Chebyshev functions are scaled to the range [-1,1] and then scaled back to output using `MinMaxScaler` (Pedregosa et al. 2011).

3.2 2D Functions

Recall that the 2D functions represent variables on the vapor dome. These are represented by two independent variables, typically pressure and composition. Standard scaling (Pedregosa et al. 2011) is applied to the inputs and then a dot product between the polynomial terms and the coefficients obtained from curve-fits is done to obtain the desired property. The functions created with this method as listed in Table 2.

The bubble point is the temperature at which a liquid creates the first bubble of vapor and begins the process of boiling. The dew point is the temperature at which the first drop of dew is formed from vapor commencing the condensation of the liquid. The bubble enthalpy is also referred to as liquid point enthalpy and dew enthalpy is also referred to as vapor point enthalpy in literature. The authors have preferred to use bubble and dew points since liquid and vapor enthalpy may also refer to those of subcooled or superheated phases causing confusion to readers.

Table 2. Saturation properties and their inputs.

Property	*Input 1*	*Input 2*
bubbleTemperature	p	X
dewTemperature	p	X
bubbleEnthalpy	p	X
dewEnthalpy	p	X
bubbleDensity	p	X
dewDensity	p	X
bubbleCp	p	X
dewCp	p	X
bubbleThermalConductivity	p	X
dewThermalConductivity	p	X
bubbleViscosity	p	X
dewViscosity	p	X
bubblePressure	T	X
dewPressure	T	X
bubbleXi	T	X
dewXi	T	X

3.3 3D Functions

Finally, the property functions which work in all the regions of the refrigerant mixture are added to the media package. The complete list of these functions is seen is Table 3. These are the functions available in the `PartialMedium` class and defined in the media with the 8 order polynomial.

Table 3. Saturation properties and their inputs.

Property	*Input 1*	*Input 2*	*Input 3*
temperature	p	h	X
density	p	h	X
dynamicViscosity	p	h	X
thermalConductivity	p	h	X
specificEntropy	p	h	X
specificHeatCapacityCp	p	h	X
specificHeatCapacityCv	p	h	X
velocityOfSound	p	h	X
specificEnthalpy	p	s	X
temperature	p	s	X
density-derh-p	p	h	X
density-derp-h	p	h	X
density-derX	p	h	X

Except for specific entropy, it is not possible to obtain a curve-fit with accuracy of 1% mean absolute percentage error with a single polynomial fitting all the regions. In these cases, polynomial fits are performed over single or two or three regions which meet the accuracy target. Then spline interpolation is done at the border of these regions. The spline function used is similar to the `SpliceFunction` in `Modelica.Media.Air.MoistAir.Utilities`. Separation of the transcritical region from the remaining is done with reduced pressure ($p/pcrit$). Then in an inner nested loop, the refrigerant quality is used to separate the remaining regions.

The density calculation is critical because it directly influences the conservation equations and can lead to non-physical scenarios if inaccurate. Additionally, density derivatives are required for time-stepping the state variables. To address this, a 3D lookup table is employed for density computation. The implementation is achieved using the `DataFiles` library by using the functions from `TableND` package. The grid variables for the table are reduced pressure, pseudo vapor quality, and Xi. Pseudo vapor quality is computed using the vapor quality formula but is not constrained between 0 and 1. Consequently, liquid enthalpy values may be negative, and vapor enthalpy values may exceed 1.0. The pseudo vapor quality range used spans from -0.5 to 1.5. Table of size $50 \times 41 \times 21$ is used for the purpose with dimensions for the three independent variables respectively.

The partial derivatives of density are obtained by finite difference method with symmetric difference quotient. Equation 10 shows the implementation for the density derivative with specific enthalpy with pressure and mixture composition constant.

$$\left(\frac{\partial \rho}{\partial h}\right)_{p,Xi} \approx \frac{\rho(h+\delta h) - \rho(h-\delta h)}{2\delta h} \tag{10}$$

4 Model Development

With the media package available it is now possible to run simulation. The `Fluid` subpackage in `Modelica Standard Library` contains several component models which can be used to run simulations.

4.1 Simulating Concentration Change

Figure 6 shows a model to simulate change in concentration of the mixture with time. The `MassFlowSource-h` and `Boundary-ph` blocks are taken from `Modelica.Fluid.Sources`. Inlet enthalpy of 300e3 kJ/kg and mass flow rate of 0.1 kg/s are provided to the mass flow source block. The mass flow source block exposes the concentration as an input. The ramp block sets up the variation of R23 composition in mixture from 0.2 to 0.7 during the simulation. Together with the constant block with input 1 and add block, the composition array X of {Xi, 1-Xi} is provided. A pressure of 10 bar is set in the boundary.

4.2 Simulating Fixed Temperature

An important benefit of Modelica is designing of controllers. For the purpose of demonstration, a simple simulation including a PID controller is created. As in the previous example from 4.1, let us say we want to maintain the temperature to a fixed value. This can be achieved by changing the specific enthalpy.

DOI 10.3384/ecp218303

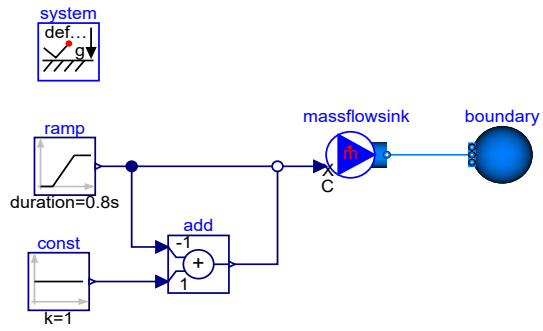


Figure 5. Simulating changing concentration of refrigerant mixture

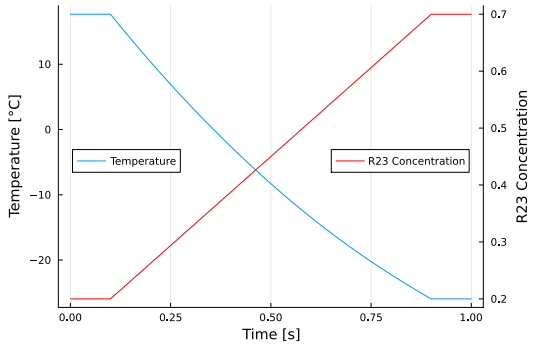


Figure 6. Temperature change of mixture with change of mixture composition

This model is setup as shown in Figure 7. A temperature sensor model is used to obtain the measured value for the PID. Setpoint value for the PID block is provided by a constant block. The value is set to 300 K.

The results can be seen in Figure 8. It can be observed that the PID is changing the enthalpy to obtain the setpoint temperature. As seen in Figure 6, the temperature of refrigerant mixture would have changed with the change in concentration. But by adjusting the specific enthalpy, it is maintained to the setpoint.

4.3 Flow Models

Thermal systems have models which determine refrigerant mass flow rate. A valve is an example of this type of model. There is no change required for these kind of models because the mass conservation, conservation of mixture constituents and energy conservation are trivial in these components. Valve models from the `Modelica.Fluid.Valves` work as is with the new media.

4.4 Lumped Refrigerant Control Volume

The components which model the control volumes with mass and energy conservation need some changes for handling refrigerant mixture. Since the concentration of mixture is also an independent variable, the density is a function of three parameters - p, h, Xi. So the density derivatives used to transform the mass and energy conservation have to include a partial derivative with Xi. For the purpose of this derivation, we will assume that the control volume has one inlet and one outlet. This assumption does not affect the derivation in any way but helps to understand it better.

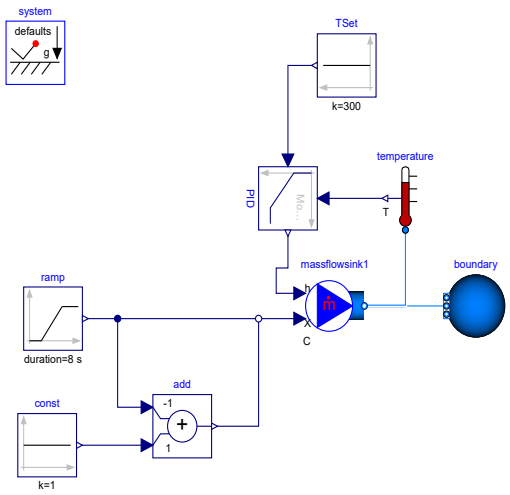


Figure 7. Simulating a simple controller to maintain temperature

4.4.1 Mass Conservation

The conservation of mass for this control volume with a volume V is given by Equation 11.

$$V\frac{d\rho}{dt} = \dot{m}_{in} - \dot{m}_{out} \tag{11}$$

Since density is a function of three independent variables, the derivative of density with time can be written in terms of its partial derivatives.

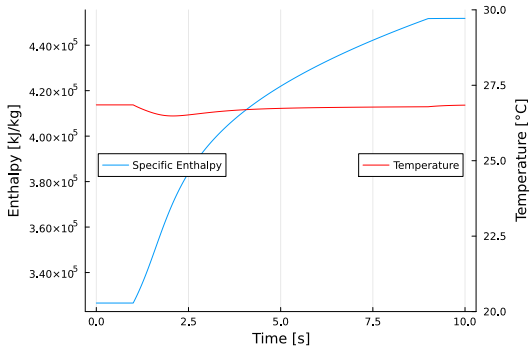


Figure 8. Controller to maintain a fixed temperature with change of mixture composition

$$V\left(\left.\frac{\partial\rho}{\partial p}\right|_{h,X}\frac{dp}{dt}+\left.\frac{\partial\rho}{\partial h}\right|_{p,X}\frac{dh}{dt}+\left.\frac{\partial\rho}{\partial X}\right|_{p,h}\frac{dX}{dt}\right)=\dot{m}_{in}-\dot{m}_{out} \tag{12}$$

This form can be evaluated with the availability of the partial derivatives calculated as shown in Equation 10.

4.4.2 Energy Conservation

Energy equation equation for the control volume can be written as below Equation 13.

$$\frac{d(\bar{m}\bar{u})}{dt}=\dot{m}_{in}h_{in}-\dot{m}_{out}h_{out} \tag{13}$$

Specific internal energy and specific enthalpy are related by:

$$\bar{u}=\bar{h}-\frac{p}{\rho} \tag{14}$$

Substituting for specific internal energy

$$\frac{d\left(\bar{m}\left(\bar{h}-\frac{p}{\bar{\rho}}\right)\right)}{dt}=\dot{m}_{in}h_{in}-\dot{m}_{out}h_{out} \tag{15}$$

For a control volume with a fixed volume.

$$V=\frac{\bar{m}}{\bar{\rho}} \tag{16}$$

Substituting Equation 16 into Equation 15. We obtain

$$\frac{d(\bar{m}\bar{h}-Vp)}{dt}=\dot{m}_{in}h_{in}-\dot{m}_{out}h_{out} \tag{17}$$

Since volume is fixed:

$$\frac{d(\bar{m}\bar{h})}{dt}-V\frac{p}{dt}=\dot{m}_{in}h_{in}-\dot{m}_{out}h_{out} \tag{18}$$

Using Chain rule of Calculus

$$\left(\bar{m}\frac{d\bar{h}}{dt}+\bar{h}\frac{d\bar{m}}{dt}\right)-V\frac{p}{dt}=\dot{m}_{in}h_{in}-\dot{m}_{out}h_{out} \tag{19}$$

We can use conservation of mass to substitute of $\frac{d\bar{m}}{dt}$

$$\bar{m}\frac{d\bar{h}}{dt}+\bar{h}(\dot{m}_{in}-\dot{m}_{out})-V\frac{p}{dt}=\dot{m}_{in}h_{in}-\dot{m}_{out}h_{out} \tag{20}$$

Simplifying

$$\bar{m}\frac{d\bar{h}}{dt}-V\frac{p}{dt}=\dot{m}_{in}(h_{in}-\bar{h})-\dot{m}_{out}(h_{out}-\bar{h}) \tag{21}$$

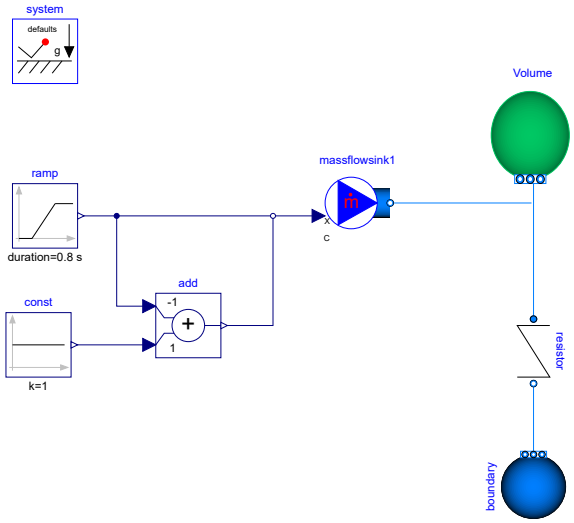


Figure 9. Control volume model simulation

4.4.3 Mixture Conservation

The conservation of the constituents of the mixture can be written as:

$$\frac{d(\bar{m}\bar{X}i)}{dt}=\dot{m}_{in}Xi_{in}-\dot{m}_{out}Xi_{out} \tag{22}$$

Using Chain rule of calculus:

$$\bar{m}\frac{d(\bar{X}i)}{dt}+\bar{X}i\frac{d(\bar{m})}{dt}=\dot{m}_{in}Xi_{in}-\dot{m}_{out}Xi_{out} \tag{23}$$

Using conservation of mass Equation 12

$$\bar{m}\frac{d(\bar{X}i)}{dt}+\bar{X}i(\dot{m}_{in}-\dot{m}_{out})=\dot{m}_{in}Xi_{in}-\dot{m}_{out}Xi_{out} \tag{24}$$

Finally, we obtain

$$\bar{m}\frac{d(\bar{X}i)}{dt}=\dot{m}_{in}(Xi_{in}-\bar{X}i)-\dot{m}_{out}(Xi_{out}-\bar{X}i) \tag{25}$$

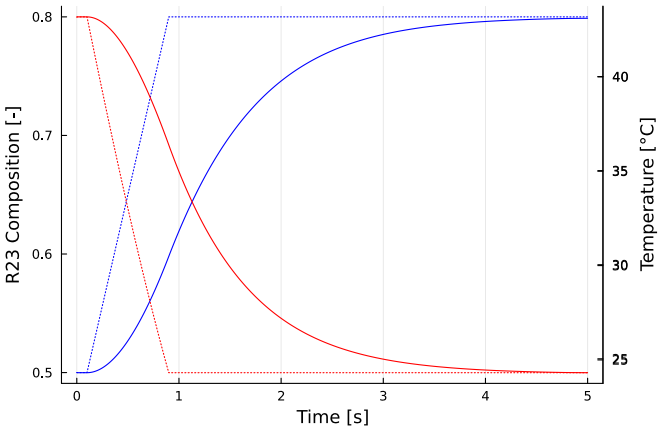


Figure 10. Control volume model simulation

DOI
10.3384/ecp218303

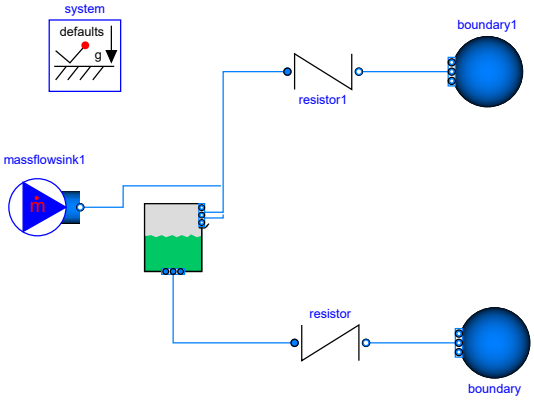


Figure 11. Separator tank model simulation

4.4.4 Model Simulation

The volume model is connected to `MassFlowSource_h` and `Boundary_ph`. The simulation model can be seen in Figure 9. A flow resistor is added between the volume and boundary. The model is simulated with varying refrigerant concentration. The refrigerant state of the lumped control volume is different from the inlet state of the refrigerant entering from `MassFlowSource_h`. The intention is to check that after the initial transients the two states become equal.

The results can be seen in Figure 10. The dotted lines are the refrigerant mixture state values from `MassFlowSource_h` while the solid lines are those from the `Volume`. It can be seen that the values converge after the initial transients.

4.5 Separator Tank

The separator tank is a key component of an auto-cascade refrigeration system. It is a specialized implementation of the lumped refrigerant control volume. A separator tank has an inlet port, a vapor outlet port and a liquid outlet port. When the liquid level is between 5% and 95% the vapor outlet port and liquid outlet port will provide pure vapor and pure liquid respectively. If the liquid level is at 0% or 100%, both outlets will provide vapor or liquid respectively. Between these values a mixed two phase flow is provided depending on a spline interpolation.

The liquid level (L) is calculated by the following equation:

$$L = \frac{1-q}{q \cdot \frac{\rho_{\text{liq}}}{\rho_{\text{vap}}} + (1-q)} \tag{26}$$

The liquid level is compared with the thresholds for dryout and flooded conditions (defaults 5% and 95%). Within the safe operating range, the outlet enthalpy at the vapor port corresponds to the dew enthalpy at the given pressure and mixture composition. For the liquid port, the bubble enthalpy at the same pressure and composition is used. For the mixture concentrations at the vapor port and liquid port, the functions `dew_Xi` and `bubble_Xi` which are functions of pressure and temperature are used. These were obtained from curve-fits of 2D REFPROP data. Since refrigerant vapor and liquid will be at thermodynamic equilibrium, they will have the same temperature and pressure. This will be equal to the temperature and pressure of the lumped control volume refrigerant mixture state.

Figure 11 shows the model with separator tank. The inlet port and vapor port can be connected into the connector array. There are two separate sinks and flow resistors for the vapor and liquid streams. The nominal flow rates in the flow resistors is adjusted to obtain points which do not have dryout or liquid overflow.

The separator tank model is initialized at 30 bar pressure, 250 kJ/kg specific enthalpy and a mixture composition of { 0.3, 0.7 }. The inlet refrigerant flow has 30 bar pressure, 350 kJ/kg specific enthalpy and a mixture composition of { 0.5, 0.5 }.

The results are shown in Figure 12, which displays the liquid level, mass flow rates at different ports, vapor quality, and mixture composition. We can observe that the vapor outlet has output of vapor and liquid outlet has quality close to that of liquid. The R23 composition is higher in the vapor phase than in the liquid phase, indicating effective mixture separation. The R23 composition changes very slowly during the simulation and appears nearly constant in the plot. Note that the separator tank and selected mass flow rates may not be optimally sized for the simulation.

5 Conclusions and Future Work

The article presented a method for simulating thermodynamic systems in Modelica, specifically addressing scenarios where the refrigerant mixture composition changes dynamically during operation. A detailed description of the media package creation was provided, with particular attention to accuracy and computational efficiency. The approach enables the use of component models from the `Modelica Standard Library` to perform simulations with the developed media package. The value of the modeling is demonstrated by the implementation of controllers designed to regulate refrigerant concentration in the mixture to achieve desired performance. A key component of the system, the separator tank used in auto-cascade refrigeration, was modeled in detail. Simulation results showed qualitative agreement with expected qualitative behavior.

The current work is part of an ongoing effort to simulate a complete auto-cascade refrigeration cycle. This will require the development of additional component models like compressor and heat exchangers which can work with the newly created refrigerant mixture media package. There is a lot of scope for enhancing the speed

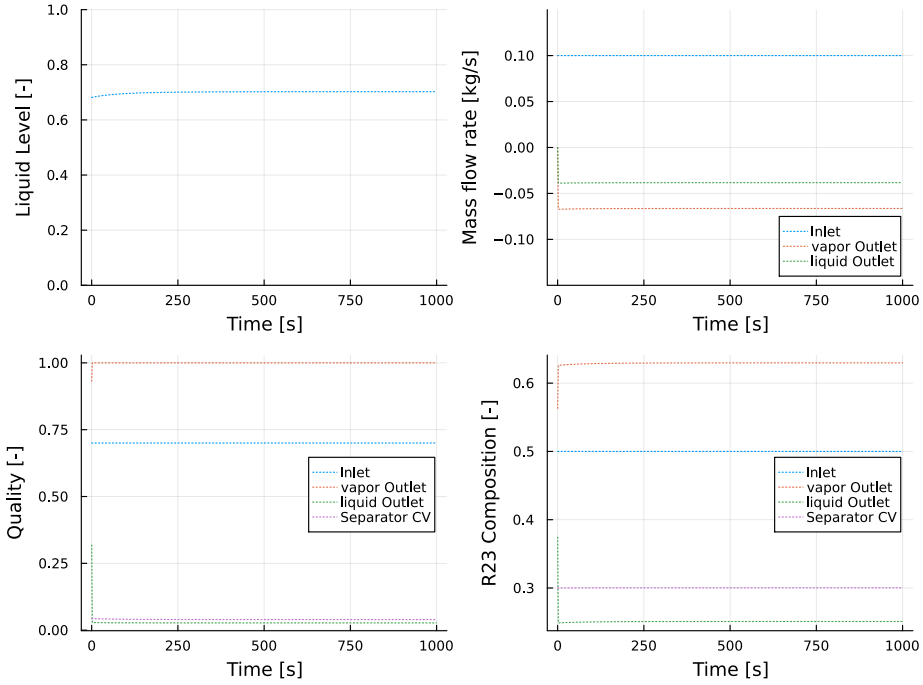


Figure 12. Results of separator tank model (a) Liquid Level (b) Mass Flow Rates (c) Quality (d) R23 Composition.

and accuracy of zeotropic refrigerant property evaluations. One promising direction is the application of multivariable Chebyshev polynomials to improve computational performance. To support applicability to other refrigerant mixture combinations, future developments will also address the automatic removal of erroneous REFPROP data.

Acknowledgements

The authors would also like to thank Network of Excellence, Trane Commerical HVAC and ETC-B for the support of this work.

References

Aute, Vikrant and Reinhard Radermacher (2014). "Standardized polynomials for fast evaluation of refrigerant thermophysical properties". In: *International Refrigeration and Air Conditioning Conference* Paper 1499.

Bai, Tao, Gang Yan, and Jianlin Yu (2019). "Experimental investigation on the concentration distribution behaviors of mixture in an ejector enhanced auto-cascade refrigeration system". In: *International Journal of Refrigeration* 99, pp. 145–152.

Du, Kai et al. (2009). "A study on the cycle characteristics of an auto-cascade refrigeration system". In: *Experimental Thermal and Fluid Science* 33.2, pp. 240–245.

Harris, Charles R et al. (2020). "Array programming with NumPy". In: *Nature* 585.7825, pp. 357–362.

He, Yongning et al. (2022). "Theoretical performance comparison for two-stage auto-cascade refrigeration system using hydrocarbon refrigerants". In: *International Journal of Refrigeration* 142, pp. 27–36.

Laughman, Christopher et al. (2024). "Enhancing Thermodynamic Data Quality for Refrigerant Mixtures: Domain-Informed Anomaly Detection and Removal". In: *International Refrigeration and Air Conditioning Conference*, Paper 2595. DOI: https://docs.lib.purdue.edu/iracc/2595.

Lemmon, E. W. et al. (2018). *NIST Standard Reference Database 23: Reference Fluid Thermodynamic and Transport Properties-REFPROP, Version 10.0, National Institute of Standards and Technology*. DOI: https://doi.org/10.18434/T4/1502528. URL: https://www.nist.gov/srd/refprop.

Li, Lixiang et al. (2018). "Fast calculation of refrigerant properties in vapor compression cycles using spline-based table look-up method (SBTL)". In: *The American Modelica Conference, Cambridge, MA, USA*, pp. 77–84.

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Virtanen, Pauli et al. (2020). "SciPy 1.0: fundamental algorithms for scientific computing in Python". In: *Nature methods* 17.3, pp. 261–272.

A Generic Non-Miscible Liquid-Gas Medium Model in Modelica with Analysis of Incompressibility Assumptions

Hubert Blervaque[1] Félix Marsollier[2]

[1]ModeliConseil, France, hbe@modeliconseil.com
[2]SIL3X, France, felix.marsollier@sil3x.fr

Abstract

Motivated by the representation of draining or leak scenarios in industrial processes, this paper introduces a generic two-phase non-miscible liquid-gas medium model. The model represents two distinct substances with independent thermodynamic states and enables transient simulations involving phase-separated liquid and gas domains. It is designed to remain compatible with the Modelica.Media and Modelica.Fluid interfaces. In developing this model, we analyze how specific assumptions about liquid compressibility — such as the dependency of enthalpy and density on pressure — affect the structure of the resulting balance equations through detailed mathematical analysis. We examine different incompressibility approaches and their implications for the formulation of conservation equations. This article focuses on a comprehensive theoretical analysis of the model formulation, highlighting the trade-offs involved and providing insights into selecting appropriate liquid models for dynamic simulations involving non-miscible flows.

Keywords: Liquid-Gas Medium, Draining and Leak Scenario, Incompressibility, Numerical Robustness, Small Modular Reactor

1 Introduction

The Modelica language (Modelica Association 2017) provides a powerful, equation-based environment for the dynamic simulation of complex physical systems, particularly thermal-hydraulic networks (Fritzson 2014). Libraries such as the Modelica Standard Library [MSL] `Modelica.Fluid`, Buildings (Wetter et al. 2014), AixLib (Müller et al. 2016), and ThermoFluidStream (Zimmer, Meißner, and Weber 2022) offer rich sets of components governed by the underlying thermodynamic properties encapsulated in replaceable `Medium` models via the `Modelica.Media` interface.

This framework is well suited for the analysis of advanced concepts, including new generation nuclear reactors: Small Nuclear Reactors (SMRs). Molten Salt Reactors (MSRs), for instance, present demanding simulation requirements due to their unique characteristics and operational scenarios (Serp et al. 2014). Applying Modelica to specific MSR transients, like draining or leak management, has highlighted challenges in modeling systems that transition from a fully liquid (molten salt) state to a fully gaseous (inert gas) state via non-miscible phase interactions.

Most components (pipes, pumps, volumes, etc.) within the `Modelica.Fluid` library — and related libraries based on the same design philosophy — are capable of simulating single-phase liquid or gas flows, as well as equilibrium two-phase flows (e.g., liquid-vapor equilibrium) (Casella and Leva 2024). These components typically rely on media definitions based on a single thermodynamic state and often assume local phase equilibrium when modeling two-phase flows. However, several applications —such as the simulation of draining scenarios or gas ingress— require the ability to model liquid and gas phases as coexisting but strictly non-miscible, without assuming thermodynamic equilibrium. In SMR safety studies, for instance, transient configurations may include the loss or recovery of coolant, resulting in domains where both gas and liquid are present but governed by distinct physical laws. Beyond the nuclear domain, similar configurations are also found in other industrial processes involving large storage tanks, fluid circuit maintenance, or safety-related depressurization events.

To address these needs, we propose a new generic Modelica medium that models two non-miscible substances while maintaining compatibility with the existing ecosystem of components. The proposed `BisubLiqGas` medium allows independent specification of compressibility assumptions for each substance. In particular, it enables the user to represent incompressible or weakly compressible liquids while preserving full thermodynamic modeling of the gas (as perfect gas). The model is designed for seamless integration with `Modelica.Fluid` components, allowing mixed-phase simulations across pipes, tanks, or valves. This allows simulation of systems that transition seamlessly between fully liquid, fully gaseous, or mixed states using standard components.

Creating such a medium, however, necessitates careful consideration of how the individual phases are modeled, particularly the liquid. Molten salts, like many liquids, are often approximated as incompressible for simplicity. Yet, incompressibility ($\rho = f(T)$ only) can lead to different modeling variants regarding the pressure dependency of thermodynamic properties like internal energy u and

enthalpy h. For example, defining enthalpy solely as $h(T)$ versus allowing an implicit pressure dependency leads to different system behavior, as detailed in Section 3. The choice of how to model liquid incompressibility and its thermodynamic consequences therefore becomes a critical aspect, impacting the robustness and efficiency of simulations involving such media.

The main contributions are thus twofold: (1) the presentation of a generic, standard-component-compatible `Medium` model for non-miscible liquid-gas systems, and (2) a detailed mathematical analysis of the trade-offs associated with different representations of liquid incompressibility within this context. Through rigorous derivation of conservation equations, we analyze the structural consequences of these modeling choices and their implications for equation complexity and physical accuracy. Our focus is on the theoretical and structural aspects of the medium formulation, providing a foundation for future implementation and performance studies.

The paper is structured as follows:

- Section 2 introduces a non-miscible two-substance `Medium` model (`BisubLiqGas`), detailing its formulation, assumptions, and general capabilities and limitations, applicable to various liquid-gas pairs.
- Section 3 delves into the core analysis of this paper: the modeling of liquid incompressibility. It discusses different incompressibility approaches and presents a comparative study, focusing on how different incompressibility assumptions impact the structure of the equations generated for a typical test case.

2 Non-miscible liquid-gas medium model

To represent the thermohydraulic behavior of the molten salt/cover gas mixture in scenarios such as draining or leak management, where the liquid and gas phases remain macroscopically separated, a specific medium model has been developed. This model, named `BisubLiqGas`, is designed to capture the essential properties of the mixture while maintaining manageable complexity for dynamic system simulation.

2.1 Medium Components

The `BisubLiqGas` medium is a binary mixture composed of two substances: one is a liquid (indexed L in formulas) and the second a gas (indexed G). While the specific substances used for illustration stem from the original MSR context (Molten Salt and Helium), the model's structure allows for generic application, as detailed in Section 2.5.

- **Liquid substance (Molten Salt)**: The specific model used is from the TRANSFORM library `TRANSFORM.Media.Fluids.NaClKClMgCl2.LinearNaClKClMgCl2_30_20_50_pT` (Greenwood et al. 2023). This choice implies modeling the ternary salt as a fluid whose density is a linear function of temperature T and pressure p, and whose specific heat capacity at constant pressure Cp_L is assumed constant:

$$\rho_L(p,T) = \rho_{L,ref}\left[1 - \beta\cdot(T - T_{ref}) + \kappa\cdot(p - p_{ref})\right] \tag{1}$$

  where $\rho_{L,ref}$, T_{ref}, and p_{ref} are the reference density, temperature, and pressure, respectively, and β and κ are the constant isobaric expansion and isothermal compressibility coefficients. The values for these constants and the transport properties (dynamic viscosity η_L, thermal conductivity λ_L) are derived from correlations or experimental data specific to this molten salt (Greenwood et al. 2023).

- **Gas substance (Helium)**: The standard MSL model `Media.IdealGases.SingleGases.He` is used. This implies that Helium is treated as an ideal gas:

$$p\cdot V = n\cdot R\cdot T \quad \Leftrightarrow \quad p = \rho\cdot R_s\cdot T \tag{2}$$

  where $R_s = R/MM_G$ is the specific gas constant for Helium. Furthermore, its specific heat capacity at constant pressure Cp_G is assumed constant and equal to $\frac{5}{2}R_s$, the theoretical value for a monoatomic ideal gas:

$$Cp_G = \frac{5}{2}\cdot R_s = \text{constant} \tag{3}$$

2.2 Fundamental Assumptions of the Mixture Model

The `BisubLiqGas` model relies on the following key assumptions for the mixture:

- **Non-miscibility**: The liquid (salt) and gas (helium) phases are considered completely immiscible. No dissolution of the gas into the liquid or vice versa is modeled. The interface between the phases is assumed to be sharp within components using this medium.
- **Absence of Phase Change**: No evaporation, condensation, solidification, or fusion is considered. The salt remains liquid, and the Helium remains gaseous throughout the simulated operating range.
- **Local Thermal Equilibrium**: Within an elementary control volume, both phases are assumed to be at the same temperature T.
- **Local Mechanical Equilibrium**: Within an elementary control volume, both phases are assumed to be at the same pressure p. (Note: Hydrostatic pressure differences can exist at the component level).

DOI 10.3384/ecp218313

- **Ideal Thermodynamic Mixing**: The specific enthalpy and internal energy of the mixture are calculated as the mass-fraction weighted average of the respective pure phase properties, implying no heat of mixing.

2.3 Implementation and State Variables

The model is implemented by inheriting from `MSL.Media.Interfaces.PartialMedium`. The chosen independent thermodynamic state variables are pressure p, temperature T, and mass fractions X (`ThermoStates.pTX` option). The liquid substance is considered as X_1 and gas as X_2. The model provides the standard set of functions required by the `PartialMedium` interface to compute thermodynamic and transport properties from various input variable combinations (e.g., `density(setState_phX(...))`, `temperature(setState_psX(...))`). The implementation ensures that calls to component property functions use the appropriate pure substance state derived from the mixture's p and T.

2.4 Mixing Rules

The thermodynamic and transport properties of the mixture are calculated from the properties of the pure components *Liq* and *Gas* (evaluated at the mixture's pressure p and temperature T) using the following mixing rules:

- **Density** (ρ): Derived from the additivity of volumes ($V = V_L + V_G$) and the definition of mass fractions ($m_i = X_i m$), using the linear equation of state for the liquid (1) and the ideal gas law (2) for the gas:

$$\rho(p,T,X) = \frac{m_L + m_G}{V_L + V_G} = \frac{\rho_L(p,T)}{X_L + X_G \frac{\rho_L(p,T) R_s T}{p}} \quad (4)$$

- **Specific Enthalpy** (h): Calculated by mass weighting, consistent with the ideal mixing assumption:

$$h(p,T,X) = X_L \cdot h_L(p,T) + X_G \cdot h_G(p,T) \quad (5)$$

where h_L is given by integrating Cp_L and accounting for the pressure dependency (from the linear model), and h_G is given by $Cp_G(T - T_0)$, with a consistent reference temperature T_0 (here 0 °C).

- **Specific Internal Energy** (u): Derived from enthalpy and density:

$$u(p,T,X) = h(p,T,X) - \frac{p}{\rho(p,T,X)} \quad (6)$$

This is equivalent to the mass-weighted average $u = X_L \cdot u_L + X_G \cdot u_G$.

- **Other Properties (Approximation)**: To simplify the model, other mixture properties Y_{mix} (dynamic viscosity η, thermal conductivity λ, specific entropy s, specific heat capacities Cp and c_v, isentropic exponent γ, speed of sound a) are approximated by a simple mass weighting of the pure phase properties:

$$Y_{mix}(p,T,X) \approx X_L \cdot Y_L(p,T) + X_G \cdot Y_G(p,T) \quad (7)$$

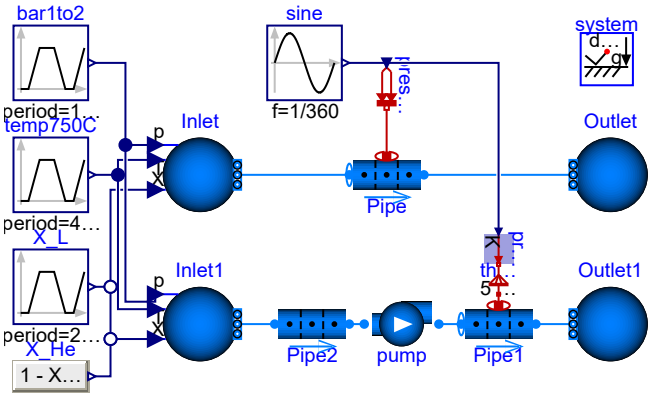


Figure 1. Unit test schematic using the `BisubLiqGas` model with a `DynamicPipe`.

2.5 Model Capabilities and Limitations

The `BisubLiqGas` medium model provides a framework for simulating systems containing non-miscible liquid-gas mixtures using standard Modelica fluid components.

2.5.1 Current Status and Validation

The model demonstrates functionality in several unit tests, such as simulating transient inlet conditions for a `Modelica.Fluid.Pipes.DynamicPipe` component (an example setup is shown in Figure 1). However, achieving robust performance across the full range of standard fluid components and complex scenarios remains an area of ongoing work. Challenges can arise from component-specific assumptions incompatible with a two-substance medium or from the numerical properties of the medium representation itself. Investigating these aspects, particularly the different approaches to modeling liquid incompressibility as discussed in Section 3, is a key objective of this paper.

2.5.2 Compatibility and Genericity

While initially motivated by an MSR application (molten salt/helium), the `BisubLiqGas` model is structured for broader applicability. The choice of liquid and gas substances can be adapted by replacing the specific component models, provided they adhere to the required base interfaces:

- **Liquid Substance**: The medium model for the liquid phase shall extend `Media.Interfaces.PartialLinearFluid`
- **Gas Substance**: The medium model for the gas phase shall extend `Media.`

`IdealGases.Common.SingleGasNasa` (or provide a compatible interface). While the base `SingleGasNasa` models can handle temperature-dependent properties, the enthalpy calculation within the `BisubLiqGas` mixing rules (Eq. (5)) currently assumes a constant specific heat capacity Cp_G (e.g., $5/2R_s$ for Helium) for simplicity in the implementation of inverse functions like `setState_phX`. This simplification should be considered when selecting or adapting the gas model for high-accuracy simulations over wide temperature ranges. The C_p for monatomic ideal gas is $5R/2$ and the value for diatomic ideal gas is $7R/2$.

2.5.3 Inherent Limitations

The use of the `BisubLiqGas` model is subject to the following fundamental limitations stemming from its core assumptions and implementation choices:

- **Component Model Validity**: The overall model validity is restricted to the pressure and temperature ranges where the chosen gas model (e.g., ideal gas behavior for Helium) and the chosen liquid model (e.g., linear approximation for the specific salt) remain acceptable representations of the real substance properties.
- **Non-Miscibility Assumption**: The assumption of perfect non-miscibility must hold. The model does not account for any solubility of the gas in the liquid or vice versa.
- **No Phase Change**: The model cannot simulate scenarios involving phase transitions (boiling, condensation, solidification, melting). Both substances are assumed to remain in their initial liquid or gas state.
- **Mixing Rule Approximations**: The linear mass-weighting rule (Eq. (7)) used for transport properties (dynamic viscosity η, thermal conductivity λ) and several thermodynamic properties (specific entropy s, specific heat capacities c_p, c_v, isentropic exponent γ, speed of sound a) is a significant simplification for non-miscible mixtures. The accuracy impact, especially for η, λ, and a, depends heavily on the specific substances and flow regime, and must be carefully evaluated for the target application.
- **Local Equilibrium Assumption**: The assumptions of local thermal and mechanical equilibrium (uniform T and p for both phases within a control volume) may break down in very rapid transients or highly heterogeneous flows.

Despite these limitations, the `BisubLiqGas` model provides a practical basis for studying dynamic system-level scenarios involving non-miscible liquid-gas mixtures where macroscopic phase separation is dominant, such as draining transients or tank pressurization/depressurization, using standard Modelica components.

3 Liquid state management

In this section, we will focus solely on the liquid substance and more specifically on the treatment of thermodynamic relationships and the handling of incompressibility aspects.

A terminological clarification is necessary because the use of the term incompressible is sometimes ambiguous. In this article, we will use the following terms with precise meanings:

- **Incompressible**: refers to a fluid whose density is constant (invariant with respect to all state variables)
- **Zero compressibility**: refers to a fluid whose isothermal compressibility is zero (or negligible), meaning that the density, which normally depends on temperature and pressure, does not vary with pressure but may vary with temperature

Let us recall that isothermal compressibility is defined by:

$$\chi_T = -\frac{1}{V} \cdot \left(\frac{\partial V}{\partial p}\right)_T = \frac{1}{\rho} \cdot \left(\frac{\partial \rho}{\partial p}\right)_T \tag{8}$$

In a single-species, single-phase fluid, the usual state variables are pressure, density, and temperature. Mass and energy balances provide equations for 2 of the 3 state variables. The closure equation is given by the fluid properties, particularly its equation of state.

The problem with liquids is that the equation of state often results in very small variations of density with pressure (low isothermal compressibility). This can cause numerical issues because pressure is often used as an explicit variable in state functions where density appears more in balance equations. We name an explicit variable one that is used as input to state functions. In this article, the following approaches will be defined and compared:

- Incompressible approach
- Semi-incompressible approach where density remains constant except in the mass conservation equation and in the buoyancy term of momentum conservation, where it is only a function of temperature. This is the counterpart of the Boussinesq approximation for system-scale calculations.
- Zero compressibility approach (density independent of pressure but dependent on temperature)
- Zero compressibility approach combined with the elimination of pressure force work in the internal energy and enthalpy relationships used in the energy balance

DOI 10.3384/ecp218313

The objective is to study the impact of deviations from rigorous thermodynamic relationships in favor of potentially better numerical performance.

3.1 Mathematical descriptions of the liquid state equations

First, let's establish the differentials of thermodynamic relations as functions of pressure and temperature.

3.1.1 Density

$$d\rho = \left(\frac{\partial \rho}{\partial p}\right)_T \cdot dp + \left(\frac{\partial \rho}{\partial T}\right)_p \cdot dT \tag{9}$$

By introducing the definition of isothermal compressibility χ_T (Eq. 8) and the definition of the isobaric thermal expansion coefficient α (Eq. 10), we obtain the differential of density (Eq. 11):

$$\alpha = \frac{1}{V} \cdot \left(\frac{\partial V}{\partial T}\right)_p = -\frac{1}{\rho} \cdot \left(\frac{\partial \rho}{\partial T}\right)_p \tag{10}$$

α can be expressed using the density ρ instead of the volume V by applying the variable change $V = m/\rho$. The mass contained in the volume being constant during the transformation.

$$d\rho = \rho \cdot (\chi_T \cdot dp - \alpha \cdot dT) \tag{11}$$

3.1.2 Internal Energy

Specific internal energy, often used for isochoric transformations and expressed as a function of temperature and density, can also be expressed as a function of pressure and temperature.

The Gibbs identity is given by:

$$dU = T \cdot dS - p \cdot dV \tag{12}$$

We assume that entropy S and volume V are functions of temperature T and pressure p :

$$S = S(T,p), \quad V = V(T,p) \tag{13}$$

By injecting the differentials of the expressions of S and V into the identity of dU (Eq. 12), we obtain:

$$\left(\frac{\partial U}{\partial T}\right)_p = T \cdot \left(\frac{\partial S}{\partial T}\right)_p - p \cdot \left(\frac{\partial V}{\partial T}\right)_p \tag{14}$$

$$\left(\frac{\partial U}{\partial p}\right)_T = T \cdot \left(\frac{\partial S}{\partial p}\right)_T - p \cdot \left(\frac{\partial V}{\partial p}\right)_T \tag{15}$$

Simplifying the partial derivative with respect to temperature by introducing the heat capacity coefficient (at constant pressure (Eq. 16)):

$$C_p = T \cdot \left(\frac{\partial S}{\partial T}\right)_p \tag{16}$$

the partial derivative of the internal energy can be expressed as function of α (Eq. 10) and C_p (Eq. 16):

$$\left(\frac{\partial U}{\partial T}\right)_p = C_p - p \cdot V \cdot \alpha \tag{17}$$

Continuing with the partial derivative with respect to pressure. To simplify the expression, we use one of Maxwell's relations.

$$\left(\frac{\partial S}{\partial p}\right)_T = -\left(\frac{\partial V}{\partial T}\right)_p \tag{18}$$

By injecting this relation into the expression of $\left(\frac{\partial U}{\partial p}\right)_T$, we obtain:

$$\left(\frac{\partial U}{\partial p}\right)_T = -T \cdot \left(\frac{\partial V}{\partial T}\right)_p - p \cdot \left(\frac{\partial V}{\partial p}\right)_T \tag{19}$$

We can now simplify this relation by introducing the isobaric thermal expansion coefficient α (Eq. 10) and the isothermal compressibility coefficient χ_T (Eq. 8).

$$\left(\frac{\partial U}{\partial p}\right)_T = -\alpha \cdot T \cdot V \; + \; p \cdot V \cdot \chi_T \tag{20}$$

Finally, let's express the differential of internal energy using only these coefficients.

$$dU = (C_p - \alpha \cdot p \cdot V) \cdot dT + (-\alpha \cdot T \cdot V + p \cdot V \cdot \chi_T) \cdot dp \tag{21}$$

This expression is general and applies to any real fluid as long as the volume can be expressed as a function of T and p. This equation remains valid when all extensive quantities are expressed per unit mass.

3.1.3 Enthalpy

$$dH = \left(\frac{\partial H}{\partial T}\right)_p \cdot dT + \left(\frac{\partial H}{\partial p}\right)_T \cdot dp \tag{22}$$

To determine the differential, let's combine the internal energy differential (Eq. 21) and a relation between enthalpy and internal energy (Eq. 23) and the differential of volume (Eq. **??**):

$$H = U + p \cdot V \tag{23}$$

$$dH = dU + p \cdot dV + V \cdot dp$$

When replacing dV by its differential (Eq. **??**) and simplifying by the compensating term, the differential of the enthalpy is obtained:

$$dH = C_p \cdot dT + V \cdot (1 - T \cdot \alpha) \cdot dp \tag{24}$$

This expression is the general differential form of specific enthalpy as a function of temperature and pressure, involving calorimetric and thermoelastic coefficients.

3.2 Mathematical descriptions of the studied approaches

3.2.1 Incompressible approach

Recall that in this approach, the density is constant and invariant with respect to all state variables ($\alpha = 0$ and $\chi_T = 0$):

The differentials of internal energy (Eq. 21) and enthalpy (Eq. 24) are then considerably simplified:

$$dU = C_p \cdot dT \tag{25}$$

$$dH = C_p \cdot dT + V \cdot dp \tag{26}$$

The problem with this approach comes from the impossibility of modeling natural circulation, density driven flows, because the driving force of the flow, induced by density variation, is zero. Therefore, we will introduce the semi-incompressible approach to tackle this problem.

3.2.2 Semi-incompressible approach

In this approach, the differentials of internal energy (25) and enthalpy (26) remain unchanged compared to the incompressible approach. Only the differential of density is modified. Actually this density relation is only used in the mass balance equation and in the buoyancy term of the momentum balance equation:

$$d\rho = -\rho \cdot \alpha \cdot dT \tag{27}$$

3.2.3 Zero compressibility approach

In this approach, the isothermal compressibility is zero ($\chi_T = 0$), which means that density is independent of pressure but can vary with temperature. From the differential of density (Eq. 11), we obtain:

$$d\rho = -\rho \cdot \alpha \cdot dT \tag{28}$$

The differentials of internal energy (Eq. 21) and enthalpy (Eq. 24) then become:

$$dU = (C_p - \alpha \cdot p \cdot V) \cdot dT \; - \; \alpha \cdot T \cdot V \cdot dp \tag{29}$$

$$dH = C_p \cdot dT + V \cdot (1 - \alpha \cdot T) \cdot dp \tag{30}$$

By noting that $\alpha \cdot p \cdot V \cdot dT = p \cdot dV$, we can find the classic form:

$$dH = dU \; + \; p \cdot dV \; + \; V \cdot dp \tag{31}$$

The zero compressibility approach is more realistic for liquids, where density variation with temperature is significant (allowing modeling of buoyancy effects), while variation with pressure is often negligible.

3.2.4 Neglected pressure work approach

In this approach, we maintain the zero compressibility assumption ($\chi_T = 0$), but we also assume that pressure work is negligible in the enthalpy equation, which means that enthalpy depends only on temperature:

$$dH = C_p \cdot dT \tag{32}$$

If we now calculate the differential of internal energy from the relationship between internal energy and enthalpy (Eq. 23), we obtain:

$$dU = C_p \cdot dT - p \cdot dV - V \cdot dp \tag{33}$$

With the zero compressibility assumption, we can express the differential of volume directly as a function of the differential of temperature which gives:

$$dU = (C_p - p \cdot V \cdot \alpha) \cdot dT \; - \; V \cdot dp \tag{34}$$

By comparing with the expression of dU obtained with the differential of internal energy assuming zero compressibility (Eq. 29), we can quantify the bias induced by this assumption of negligible pressure work. The bias is therefore:

$$\begin{aligned} \delta dU &= dU_{\delta W=0} - dU_{\chi_T=0} \\ &= -V \cdot dp - (-\alpha \cdot T \cdot V \cdot dp) \\ &= -V \cdot (1 - \alpha \cdot T) \cdot dp \end{aligned} \tag{35}$$

This simplified approach can be useful numerically, but it leads to a thermodynamic inconsistency that can affect the accuracy of the simulation, particularly in cases where pressure variations are significant.

3.3 Conservation equations

Let's establish the mass balance for a fixed volume of a single-species single-phase fluid.

DOI 10.3384/ecp218313

3.3.1 Conservation of mass

$$V \cdot \frac{\mathrm{d}\rho}{\mathrm{d}t} = \sum_{boundaries} \dot{m} \tag{36}$$

Since the mass flow rate is a function of pressure, it is preferable to choose pressure as one of the explicit variables. With the assumption of a single-phase fluid, it is preferable to take temperature as the second explicit variable.

The left term of the continuity equation (Eq. 36) expressed as a function of temperature and pressure derives using the differential of the density (Eq. 11):

The right term is a function of the mass flow rate at the boundaries of the volume. By simplifying the flow as non-inertial, the mass flow rate becomes exclusively a function of the pressure losses at the boundaries. The calculation of the pressure losses requires having the value of the density and temperature to calculate the transport properties.

$$\dot{m} = f(\Delta p, \rho, T) \tag{37}$$

According to the scheme used to compute the flow rate (upstream, centered, order 1, 2 etc), the indices of the density, temperature and pressure differ. However, the general form is preferred to study the non linear behavior. Please note that implicit forms can raised depending on the scheme used. For clarity sake, a simplified centered scheme can be used in this section.

Grouping the left and right terms, we obtain:

$$V \cdot \rho \cdot \left(\chi_T \cdot \frac{\mathrm{d}p}{\mathrm{d}t} - \alpha \cdot \frac{\mathrm{d}T}{\mathrm{d}t} \right) = \sum f(\Delta p, \rho, T) \tag{38}$$

Let's now apply the assumptions of the different approaches to this equation.

Incompressible approach: Since the density is constant, the derivatives of the density with respect to pressure and temperature are zero. The density is always used to calculate the pressure losses but is no longer a function of the explicit variables, so it is removed from the function "f". The equation (Eq. 38) becomes a simple algebraic equation.

$$0 = \sum f(\Delta p, T) \tag{39}$$

Semi-incompressible approach: Since the density is only a function of temperature, it can be removed from the function "f". The equation (Eq. 38) can be simplified to:

$$\frac{\mathrm{d}T}{\mathrm{d}t} = \frac{\sum f(\Delta p, T)}{-\rho \cdot V \cdot \alpha} \tag{40}$$

We can clearly see with this approach that:

- The mass conservation equation becomes an equation for the time derivative of temperature
- The equation is linear and the derivative is explicit (does not depend on itself or on another time derivative)

Zero compressibility approach: The expression for density is exactly the same as for the semi-incompressible approach. The conclusions are therefore the same.

Neglected pressure work approach: The expression for density is exactly the same as for the semi-incompressible approach. The conclusions are therefore the same.

3.3.2 Conservation of energy

Let's apply the same process to the energy balance.

$$V \cdot \frac{\mathrm{d}(\rho \cdot u)}{\mathrm{d}t} = \sum_{boundaries} \dot{H} + \sum_{boundaries} \dot{Q} \tag{41}$$

For the energy balance, we will neglect external thermal inputs ($\sum \dot{Q}$). They do not change the conclusions.

The gross enthalpy flow $\dot{H}$ is the product of the mass flow rate and the specific enthalpy. From (Eq. 37), the mass flow rate can be expressed from the explicit variables:

$$\dot{H} = \dot{m} \cdot h = \sum f(\Delta p, \rho, T) \cdot h \tag{42}$$

According to the scheme used to compute the flow rate (upstream, centered, order 1, 2 etc), the indices of the density, temperature, pressure and enthalpy differ. However, the general form is preferred to study the non linear behavior. Please note that implicit forms can raised depending on the scheme used. For clarity sake, a simplified centered scheme can be used in this section.

Using Eq. 41 with the partial derivatives form and the Eq. 37:

$$\frac{\mathrm{d}p}{\mathrm{d}t} \cdot \rho \cdot V \cdot \left[\frac{\partial u}{\partial p} + u \cdot \chi_T \right] + \frac{\mathrm{d}T}{\mathrm{d}t} \cdot \rho \cdot V \cdot \left[\frac{\partial u}{\partial T} - u \cdot \alpha \right] = \sum (f(\Delta p, \rho, T) \cdot h) \tag{43}$$

Incompressible approach: Since density is constant and specific internal energy depends only on temperature, many partial derivatives become zero. Equation (43) becomes:

$$\frac{\mathrm{d}T}{\mathrm{d}t} = \frac{\sum (f(\Delta p, T) \cdot h)}{\rho \cdot V \cdot c_p} \tag{44}$$

- The energy conservation equation becomes an equation for the time derivative of temperature

- The equation is linear and the derivative is explicit (does not depend on itself). However, the right-hand term is also a function of pressure via the specific enthalpy formula.

Semi-incompressible approach: Since density and specific internal energy depend only on temperature, many partial derivatives become zero. Equation (43) becomes:

$$\frac{\mathrm{d}T}{\mathrm{d}t} = \frac{\sum(f(\Delta p, T)\cdot h)}{\rho \cdot V \cdot (c_p - \alpha \cdot u)} \tag{45}$$

The conclusions are identical to the incompressible approach. The only difference comes from the term in the denominator, which is corrected for the expansion effect.

Zero compressibility approach: The energy conservation equation (43) becomes:

$$\frac{\mathrm{d}p}{\mathrm{d}t}\cdot V\cdot(-\alpha\cdot T) + \frac{\mathrm{d}T}{\mathrm{d}t}\cdot\rho\cdot V\cdot\left[\left(c_p - \frac{p\cdot\alpha}{\rho}\right) - u\cdot\alpha\right] = \sum(f(\Delta p, T)\cdot h) \tag{46}$$

- The equations are linear and the derivative is explicit (does not depend on itself).
- The resolution of pressure is no longer an algebraic system but becomes an ordinary differential equation.

Zero compressibility + neglected pressure work approach: Since density and enthalpy depend only on temperature, the energy conservation equation (43) becomes:

$$\frac{\mathrm{d}p}{\mathrm{d}t}\cdot V\cdot\rho\cdot\left[\frac{\partial u}{\partial p}\right] + \frac{\mathrm{d}T}{\mathrm{d}t}\cdot V\cdot\left[\rho\cdot\frac{\partial u}{\partial T} + u\cdot\frac{\partial \rho}{\partial T}\right] = \sum f_2(\Delta p, T) \tag{47}$$

Note that for the formulation of internal energy, we cannot use equation (21) but (34), which derives from enthalpy without the pressure force work term. Moreover, the function f can be replace by f2 since the specific enthalpy depends only on temperature:

$$\frac{\mathrm{d}p}{\mathrm{d}t}\cdot(-V) + \frac{\mathrm{d}T}{\mathrm{d}t}\cdot\rho\cdot V\cdot\left[c_p - \frac{p\cdot\alpha}{\rho} - u\cdot\alpha\right] = \sum f_2(\Delta p, T) \tag{48}$$

3.4 Applicability of `TableBased` medium on these approaches

In this section, we will explain why the pre-existing generic `TableBased` medium of the MSL are not perfectly suited to our needs. All the four approaches presented in previous section cannot be reproduced with this medium by tuning the following parameters:

- singleState
- enthalpyOfT

For reminder, the definition of the singleState parameter is given in the MSL package `Media.UsersGuide.MediumUsage.Constants` and is as follows:

> singleState = true, if u and d are not a function of pressure, and thus only a function of a single thermal variable (temperature or enthalpy) and of Xi for a multiple substance medium. Usually, this flag is true for incompressible media. It is used in a model to determine whether 1+nXi (singleState=true) or 2+nXi (singleState=false) initial conditions have to be provided for a volume element that contains mass and energy balance.

For a purely incompressible approach, we must therefore have $singleState = true$. However, looking more closely at the system of equations, we find that $singleState = true$ is not always the right condition for the mass density and specific internal energy to depend only on temperature.

If $enthalpyOfT = false$, the function for calculating specific enthalpy (h_pT) becomes:

$$h = h_0 + \int_{T_0}^{T} cp\cdot dT + \frac{p - p_{ref}}{\rho}\cdot(1 - \alpha\cdot T) \tag{49}$$

If the density is constant (incompressible total), this expression simplifies to:

$$h = h_0 + \int_{T_0}^{T} cp\cdot dT + \frac{p - p_{ref}}{\rho} \tag{50}$$

The function for calculating specific internal energy with $singleState = true$ becomes:

$$u = h - \frac{p_{ref}}{\rho} \tag{51}$$

By substituting the expression for specific enthalpy, we obtain:

$$\begin{aligned} u &= h_0 + \int_{T_0}^{T} cp\cdot dT + \frac{p - p_{ref}}{\rho} - \frac{p_{ref}}{\rho} \\ u &= h_0 + \int_{T_0}^{T} cp\cdot dT + \frac{p}{\rho} \end{aligned} \tag{52}$$

We can clearly see that the pressure is always present in the expression for specific internal energy. On the contrary, if $singleState = false$, the function for calculating specific internal energy becomes:

$$\begin{aligned} u &= h_0 + \int_{T_0}^{T} cp\cdot dT + \frac{p - p_{ref}}{\rho} - \frac{p}{\rho} \\ u &= h_0 + \int_{T_0}^{T} cp\cdot dT - \frac{p_{ref}}{\rho} \end{aligned} \tag{53}$$

DOI 10.3384/ecp218313

In this case, specific internal energy and mass density are well independent of pressure. **However, we have the opposite behavior compared to what the user expects**. Moreover, this can cause problems with initialization because with singleState = false, the pressure is initialized in the capacitive components but has no inertial term.

If now $enthalpyOfT = true$ and $singleState = true$, the function to calculate specific enthalpy (h_T) becomes:

$$h = h_0 + \int_{T_0}^{T} cp \cdot dT \tag{54}$$

and the specific internal energy (u_T) becomes:

$$u = h_0 + \int_{T_0}^{T} cp \cdot dT - \frac{p_{ref}}{\rho} \tag{55}$$

Since the mass density is constant, the expression is independent of pressure.

The **semi-incompressible approach** could be modeled with the TableBased medium in the MSL. It would require forcing the constant $DensityOfT = false$ even though the density is a function of temperature. Indeed, the constantis defined as $DensityOfT = size(tableDensity, 1) > 1$. The parameters would then be:

- singleState = false
- enthalpyOfT = false
- densityOfT = false

The approach of **zero compressibility** cannot be modellable with the TableBased medium in the MSL even with the following parameters:

- singleState = false
- enthalpyOfT = false
- densityOfT = true

Introducing the (49) in the expression $u = h - \frac{p}{\rho}$, it can be remarked that there is the missing following term, which is a correction term of the specific heat capacity (check (29)) in the specific internal energy expression:

$$\int_{T_0}^{T} -\alpha \cdot \frac{p}{\rho} \cdot dT$$

Similarly, the approach of **zero compressibility + neglected pressure work** cannot be modeled with the TableBased medium in the MSL for the same reason than for the zero compressibility approach.

Based on the analysis operated in this section, our further developments must concern the modeling of a own liquid medium for the bisubtance model rather than using an existing medium from the Modelica Standard Library (MSL) in order to:

- Build expertise in medium development
- Have total control over the approximations made
- The TableBased medium in the MSL rebuilds polynomials from data points. It does not allow defining arbitrary relationships between variables.

4 Comparison of approaches applied to a flow in a vertical pipe

Let us now attempt to quantify the differences between the approaches by applying them to a simple case: the flow in a vertical adiabatic pipe. In this article, we will focus exclusively on the theoretical study. A further study is ongoing and will contain the practical comparison using an equivalent Modelica model, compiled and executed with OpenModelica.

Suppose a perfectly adiabatic vertical pipe filled with an inert liquid fluid (no heat production). Let's start by assuming that the fluid is static. We can generalize later. The conservation of energy gives us an invariance of the sum of enthalpy and potential energy of gravity.

$$dH + m \cdot g \cdot dz = 0 \tag{56}$$

Replacing dH by its exact expression:

$$C_p \cdot dT + V \cdot (1 - T \cdot \alpha) \cdot dp + m \cdot g \cdot dz = 0 \tag{57}$$

We can express dp in terms of dz using the hydrostatic relation:

$$dp = -\rho \cdot g \cdot dz \tag{58}$$

The conclusions remain valid even when introducing a term for pressure loss: $dp = -\rho\ g\ dz\ -\ dp_{loss}$

$$C_p \cdot dT + V \cdot (1 - T \cdot \alpha) \cdot (-\rho \cdot g \cdot dz) + m \cdot g \cdot dz = 0 \tag{59}$$

By simplifying the terms that cancel out and dividing by the mass ($m = \rho\ V$), we obtain:

$$c_p \cdot dT + T \cdot \alpha \cdot g \cdot dz = 0 \tag{60}$$

We can therefore determine the temperature variation as a function of height variation:

$$\frac{dT}{dz} = \frac{-\alpha \cdot T \cdot g}{c_p} \tag{61}$$

NB: With a mass density that depends on temperature, a temperature change induces a change in mass density. By mass conservation, in steady state, this induces a change in flow rate, hence kinetic energy. The energy balance should therefore be written:

$$H + \frac{1}{2} \cdot m \cdot V^2 + m \cdot g \cdot (z - z_0) = \text{constant} \tag{62}$$

We choose to neglect the effects of kinetic energy for simplicity.

For a real transformation with fluid displacement, we make a change of variable to make the pressure loss term appear in the hydrostatic equation (58). In practice, a pressure loss always has a negative value. However, we assume that an eventual flow would be in the "z" direction. This avoids introducing vectors and allows us to only look at values.

$$\frac{dT}{dz} = \frac{-\alpha \cdot T \cdot g}{c_p} + \frac{(1-\alpha \cdot T) \cdot dp_{loss}}{\rho \cdot c_p} \tag{63}$$

Let's now consider the case where the fluid is incompressible ($\alpha = 0$) :

$$\frac{dT}{dz} = \frac{dp_{loss}}{\rho \cdot c_p} \tag{64}$$

In this case, the temperature can only increase (since $dp_{loss} < 0$) and depends only on the pressure loss. We clearly see here the irreversibility of the conversion of mechanical pressure energy into thermal energy.

The bias on the temperature gradient induced by the assumption of incompressibility is therefore of the order of $\frac{\alpha \cdot T \cdot g}{c_p}$.

Let's now consider the case where the pressure term is negligible in the enthalpy, i.e., $dH(T) = C_p \cdot dT$:

$$C_p \cdot dT + m \cdot g \cdot dz = 0 \tag{65}$$

$$\frac{dT}{dz} = \frac{-g}{c_p} \tag{66}$$

In this case, the temperature no longer depends on the pressure loss. It depends only on height.

The ongoing comparison of the approaches with OpenModelica is using a "DynamicPipe" component from the Modelica Standard library (MSL) coupled with a "pressure" sink at the pipe outlet (top part of the vertical pipe). The inlet conditions studied are of kind source of mass (and the necessary potential variables) flow and source of pressure (and the necessary potential variables).

5 Conclusion

This paper has addressed two key challenges in thermal-hydraulic system modeling: (1) the creation of a generic non-miscible liquid-gas medium that integrates with standard Modelica component libraries, and (2) the investigation of different approaches to model liquid incompressibility through detailed mathematical analysis.

The developed `BisubLiqGas` medium effectively represents systems that transition between fully liquid, fully gaseous, or mixed states, maintaining compatibility with standard components. This allows simulation of important scenarios like draining or leak management in systems such as Molten Salt Reactors. The model's generic structure enables adaptation to various liquid-gas pairs beyond the initial molten salt/helium application.

Our detailed mathematical analysis of different incompressibility approaches revealed significant implications for equation structure and numerical performance:

- The pure incompressible approach, while offering the simplest equation structure (algebraic pressure resolution), cannot model natural convection effects due to its constant density assumption.
- The semi-incompressible approach enables temperature-dependent density variations essential for buoyancy-driven flows, converting mass conservation into an explicit differential equation for temperature.
- The zero compressibility approach provides greater thermodynamic rigor but transform pressure resolution from algebraic to differential equations, increasing system complexity.
- The neglected pressure work variant offers numerical simplification but introduces thermodynamic inconsistencies that may affect accuracy in systems with significant pressure variations.

The theoretical analysis of vertical pipe flow demonstrated quantifiable differences between approaches. For instance, the temperature gradient bias induced by incompressibility assumptions is of the order $\frac{\alpha \cdot T \cdot g}{c_p}$, while neglecting pressure work in enthalpy leads to temperature gradients that depend solely on elevation rather than pressure losses.

This analysis revealed fundamental limitations of existing MSL `TableBased` media for implementing these approaches, particularly the counterintuitive behavior of the `singleState` parameter and the inability to represent zero compressibility formulations correctly. This justifies the development of specialized media rather than relying on generic table-based implementations.

These findings highlight the inherent trade-offs between physical accuracy and numerical efficiency in fluid modeling. The choice of incompressibility approach should be guided by the specific requirements of the application: natural convection modeling necessitates at least the semi-incompressible approach, while systems without significant buoyancy effects may benefit from the numerical advantages of the simpler incompressible model.

Future work will focus on implementing and validating these theoretical approaches in practical Modelica simulations, quantifying their performance differences in large-scale system models, and extending the analysis to more

DOI 10.3384/ecp218313

complex flow configurations. Additionally, exploring more sophisticated mixing rules for transport properties in the non-miscible medium would enhance the model's accuracy for specialized applications.

Acknowledgements

We acknowledge Pierre Lassausset from NAAREA for bringing to their attention the first problem investigated in this paper: the development of a single liquid-gas medium compatible with existing `Fluid` components. We are grateful to BLUE CAPSULE, a start-up developing a natural circulation Sodium-cooled reactor concept, and to NAAREA, a start-up developing a Molten Salt Reactor (MSR) concept, for the opportunities to explore these topics and contribute to MODELICA development within the academic community.

References

Casella, Francesco and Alberto Leva (2024). "Modelling and Simulation of Thermal Power Generation Systems with Modelica: 20 Years of Experience with the ThermoPower Library". In: *IFAC-PapersOnLine* 58.13. 12th IFAC Symposium on Control of Power and Energy Systems - CPES 2024, pp. 227–234. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2024.07.487. URL: https://www.sciencedirect.com/science/article/pii/S240589632400586X.

Fritzson, Peter (2014). *Principles of Object-Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach*. Wiley-IEEE Press. ISBN: 978-1118859124.

Greenwood, M. Scott et al. (2023). *TRANSFORM - TRANsient Simulation Framework Of Reconfigurable Models*. Oak Ridge National Laboratory. URL: https://github.com/ORNL-Modelica/TRANSFORM-Library.

Modelica Association (2017-04). *Modelica – A Unified Object-Oriented Language for Systems Modeling. Language Specification Version 3.4*. Tech. rep. Linköping: Modelica Association. URL: https://www.modelica.org/documents/ModelicaSpec34.pdf.

Müller, Dirk et al. (2016). "AixLib – An Open-Source Modelica Library within the IEA EBC Annex 60 Framework". In: *Proceedings of BauSIM 2016*. Aachen, Germany.

Serp, J. et al. (2014). "The molten salt reactor (MSR) in Generation IV: Overview and perspectives". In: *Progress in Nuclear Energy* 77, pp. 308–319. DOI: 10.1016/j.pnucene.2014.02.014.

Wetter, Michael et al. (2014). "Modelica Buildings library". In: *Journal of Building Performance Simulation* 7.4, pp. 253–270. DOI: 10.1080/19401493.2013.765506.

Zimmer, Dirk, Michael Meißner, and Niels Weber (2022). "The DLR ThermoFluid Stream Library". In: *Electronics* 11.22. ISSN: 2079-9292. DOI: 10.3390/electronics11223790. URL: https://www.mdpi.com/2079-9292/11/22/3790.

DOI
10.3384/ecp218313

Centrifugal Pump Model of the DLR Thermofluid Stream Library

Raphael Gebhart[1] Martin Düsing[2] Niels Weber[1] Franciscus L. J. van der Linden[1]

[1]Institute of System Architectures in Aeronautics, German Aerospace Center, Münchener Strasse 20, 82234 Weßling, Germany, {raphael.gebhart,franciscus.vanderlinden,niels.weber}@dlr.de

[2]HELLA GmbH & Co. KGaA, Beckumer Str. 130, 59552 Lippstadt, Germany, martin.duesing@forvia.com

Abstract

Modeling pumps in 1D flows in Modelica is not new. In this paper a new approach is presented that can be used for general centrifugal pumps as well as centrifugal pumps based on measurement data for the DLR Thermofluid Stream (TFS) Library. The presented model uses affinity laws for good scaling. This approach allows to predict the pump behavior also for pumps where no measurement data exist based on data from similar pumps. The models are designed with focus on numerically robustness and focus on normal pump operation. They avoid nonlinear equation systems and can be initialized at zero mass flow rate and/or zero speed. Furthermore care has been taken that extrapolation of the pump data outside the data range is robust without interpolation artifacts. A pump surge example demonstrates that the TFS can handle instationary abnormal pump behavior like surge limit cycles by default without any further modifications needed.

Keywords: Centrifugal Pump; Hydraulic Transients; Thermal Modeling; Thermofluid Stream; TFS

1 Introduction

The pump models available in the Thermofluid Stream (TFS) Library (Zimmer 2020; Zimmer, Meißner, and Weber 2022) have been used for simple modeling tasks. However, in order to create more realistic pumps or pumps based on real measurement data, we have found that current models do not meet all the requirements needed by industry. Therefore, new pump models have been developed that are more realistic. These models are in contrast to the models from Modelica.Fluid, since they avoid division by zero. Furthermore they avoid using polynomials of high degree for interpolation, which can cause oscillation at the edges of the interpolation interval known as *Runge's phenomenon*, see (Trefethen 2019). There exists extensive literature on pump modeling, well known german-language books are e.g. (Gülich 2010) or (Sigloch 2013), and english-language books are e.g. (Stepanoff 1957) or (Karassik et al. 2007). Furthermore it might be worth mentioning (Chaudhry 2014) focusing on hydraulic transients. In Modelica pump models are part of fluid libraries as Modelica.Fluid (Casella et al. 2006), the Buildings Library (Wetter et al. 2014) or TIL Library (Richter 2008). Modelica.Fluid uses polynomial interpolation for head characteristic and either fixed efficiency or quadratic power characteristic. For the Buildings Library one can use spline interpolation of measurement data. The model incorporates several numerical workarounds to avoid singularities and ensure a unique solution (Wetter 2013). Alternatively a method based on the Euler Number can be used that requires only reference values at the point of best efficiency as inputs (Fu, Blum, and Wetter 2023). The TIL Library uses a quadratic head characteristic $h \sim \dot{V}|\dot{V}|$ that has an inflection point at zero flow and uses an empirical relation for the loss power.

First, subsection 2.1 introduces the centrifugal pump model. Subsequently in subsection 2.2 measurements of a sample development pump from the automotive supplier HELLA GmbH & Co. KGaA are used, to validate the pump model, see an example in Figure 1. These pumps, which are developed in several power classes from 80 W to 400 W, are used in battery electric cars to cool and heat several components (e.g. battery, electrical machine, cabin). In most cooling systems the pumps are integrated into a Coolant Control Hub (CCH) together with valves. The fluid used in these systems are mixtures of water and glyocol. In addition, the models have been validated against data from pumps available in the Modelica Buildings library (Wetter et al. 2014). In subsection 2.3 it is shown that centrifugal pump characteristics have great similarities. Hence realistic pump characteristics can be set up with minimum knowledge. Furthermore in section 3, a pump surge example demonstrates that the TFS can handle instationary abnormal pump behavior like surge limit cycles by default without any further modifica-

tions needed and without any numerical modifications of the pump characteristics necessary.



Figure 1. Sample automotive development pump (copyright HELLA GmbH & Co. KGaA).

2 Centrifugal pump model

2.1 Basic equations

The centrifugal pump model is based on affinity laws that govern the scaling behavior of head $h \sim \omega^2$, discharge $\dot{V} \sim \omega$, and power $P \sim \rho\omega^3$ with respect to speed ω and fluid density ρ. These relationships, commonly referred to as homologous relationships, are detailed in any textbook on pumps e.g. (Stepanoff 1957; Karassik et al. 2007; Gülich 2010; Sigloch 2013). The model targets normal pump operating conditions characterized by $\dot{V}, \omega, h, P \geq 0$, as measurements for abnormal pump operating conditions are scarce and normal pump operating conditions are of primary interest. The model uses quadratic polynomials for the approximation of homologeous head h/ω^2 and homologeous power $P/(\rho\omega^3)$ as a function of homologeous discharge $\dot{V}/\omega$:

$$\frac{h}{\omega^2} = \tilde{c}_{h,0} + \tilde{c}_{h,1}\frac{\dot{V}}{\omega} + \tilde{c}_{h,2}\left(\frac{\dot{V}}{\omega}\right)^2, \quad (1)$$

$$\frac{P}{\rho\omega^3} = \frac{\tau\omega}{\rho\omega^3} = \tilde{c}_{P,0} + \tilde{c}_{P,1}\frac{\dot{V}}{\omega} + \tilde{c}_{P,2}\left(\frac{\dot{V}}{\omega}\right)^2, \quad (2)$$

where τ denotes the torque, and $\tilde{\boldsymbol{c}}_h = (\tilde{c}_{h,0}, \tilde{c}_{h,1}, \tilde{c}_{h,2})^\mathsf{T}$ and $\tilde{\boldsymbol{c}}_P = (\tilde{c}_{P,0}, \tilde{c}_{P,1}, \tilde{c}_{P,2})^\mathsf{T}$ are non-dimensionless coefficients. The head characteristic of a centrifugal pump is based on the linear Euler characteristic and extended by accounting for friction and shock losses, which can be well approximated by quadratic terms for centrifugal pumps, see e.g. (Kallesøe 2005, eq. (3.19)), (Sigloch 2013, fig. 9.5), (Gülich 2010, fig. 4.3), (Stepanoff 1957, fig. 9.8). Similarly, the torque characteristic is also found to follow a quadratic relationship (Kallesøe 2005, eq. (3.24)).

Equation 1 and Equation 2 can be rewritten in terms of normalized variables as:

$$h_\mathrm{n} = c_{h,0}\omega_\mathrm{n}^2 + c_{h,1}\omega_\mathrm{n}\dot{V}_\mathrm{n} + c_{h,2}\dot{V}_\mathrm{n}^2, \quad (3)$$

$$P_\mathrm{n} = \omega_\mathrm{n}\tau_\mathrm{n} = \omega_\mathrm{n}\rho_\mathrm{n}\left(c_{P,0}\omega_\mathrm{n}^2 + c_{P,1}\omega_\mathrm{n}\dot{V}_\mathrm{n} + c_{P,2}\dot{V}_\mathrm{n}^2\right), \quad (4)$$

where $h_\mathrm{n} = h/h_\mathrm{ref}$, $\dot{V}_\mathrm{n} = \dot{V}/\dot{V}_\mathrm{ref}$, $\omega_\mathrm{n} = \omega/\omega_\mathrm{ref}$, $P_\mathrm{n} = P/P_\mathrm{ref}$, $\tau_\mathrm{n} = \tau/\tau_\mathrm{ref}$, and $\rho_\mathrm{n} = \rho/\rho_\mathrm{ref}$ are the normalized head, discharge, speed, power, torque, and fluid density, respectively, h_ref, $\dot{V}_\mathrm{ref}$, ω_ref, $P_\mathrm{ref} = \tau_\mathrm{ref}\omega_\mathrm{ref}$, and ρ_ref are the corresponding reference values and $\boldsymbol{c}_h = (c_{h,0}, c_{h,1}, c_{h,2})^\mathsf{T}$ and $\boldsymbol{c}_P = (c_{P,0}, c_{P,1}, c_{P,2})^\mathsf{T}$ are dimensionless coefficients.

From a mathematical perspective, reference values can be chosen arbitrarily. In practice, however, they are chosen at the point of best efficiency, also referred to as the rated conditions (Chaudhry 2014, sec. 4.3), (Stepanoff 1957). This normalization facilitates the comparison of different pumps, as similar pumps will exhibit similar normalized characteristics. The effect of impeller diameter D on performance, described by head $h \sim D^2$, discharge $\dot{V} \sim D^3$, and power $P \sim D^5$, is implicitly accounted for through these reference quantities. Therefor pump scaling can be accomplished by adapting the reference values accordingly.

An additional advantage of Equation 3 and Equation 4 is the avoidance of division by zero at zero discharge $\dot{V} = 0$ and/or zero speed $\omega = 0$, which is essential for initializing simulations from rest. This contrasts with models within `Modelica.Fluid`, as noted in (Wetter 2013). Furthermore, using polynomials of degree higher than two would either violate the affinity laws or necessitate additional numerical handling to prevent division by zero.

It is also worth noting that, unlike in (Wetter 2013), Equation 3 is not constrained to be monotonically decreasing. Although the presence of a maximum can lead to dynamic instability to be avoided in practice, it is a physically observed phenomenon for centrifugal pumps. While steady-state solvers may fail to converge at this point, the TFS library does not assume steady-state conditions. Instead, it incorporates the fluid inertia by default and is capable of capturing transient behaviors such as surge limit cycles. Consequently, the numerical workarounds described in (Wetter 2013) become unnecessary. Nevertheless, Equation 3 and Equation 4 require modification to qualitatively account for reverse operation with discharge $\dot{V} < 0$, as discussed in section 3. In order to quantitatively cover all four quadrants of pump operation, it might be worthwhile to integrate the approach presented in (Chaudhry 2014, sec. 4.3) into the TFS in the future. Finally, pump efficiency is defined as $\eta = \rho g h \dot{V}/P$, where g is the standard gravitational acceleration.

DOI 10.3384/ecp218325

2.2 Approximation based on measurement data

The six coefficients of the quadratic head and power approximation polynomials $\boldsymbol{c}_h = (c_{h,0}, c_{h,1}, c_{h,2})^\intercal$ and $\boldsymbol{c}_P = (c_{P,0}, c_{P,1}, c_{P,2})^\intercal$ can each be determined by solving a quadratic polynomial regression, i.e. solving the linear equation systems $\boldsymbol{A}^\intercal\boldsymbol{A}\boldsymbol{c}_h = \boldsymbol{A}^\intercal\boldsymbol{b}_h$ and $\boldsymbol{A}^\intercal\boldsymbol{A}\boldsymbol{c}_P = \boldsymbol{A}^\intercal\boldsymbol{b}_P$ for the coefficients $\boldsymbol{c}_h$ and $\boldsymbol{c}_P$ respectively, where the matrix elemets $A_{ij} = (\dot{V}_i/(\dot{V}_{\mathrm{ref}}\omega_{\mathrm{n},i}))^{j-1}$ and the vector elements $b_{h,i} = h_i/(h_{\mathrm{ref}}\omega_{\mathrm{n},i}^2)$ and $b_{P,i} = P_i/(P_{\mathrm{ref}}\omega_{\mathrm{n},i}^3)$. Thereby h_i, $\dot{V}_i$, ω_i and P_i are measurements of head, discharge, speed and power respectively and $\omega_{\mathrm{n},i} = \omega_i/\omega_{\mathrm{ref}}$ is the normalized measured speed. If only measurements at reference speed ω_{ref} are used, the normalized speed $\omega_{\mathrm{n},i} = 1$. The reference quantities h_{ref}, $\dot{V}_{\mathrm{ref}}$ and P_{ref} can e.g. be defined at maximum measured efficiency $\eta_{\mathrm{ref}} = \max \rho g h_i \dot{V}_i / P_i$, which has advantages that are shown subsequently. In addition reference quantities improve the condition number of the matrix $\boldsymbol{A}$, since coefficients $\boldsymbol{c}_h$ and $\boldsymbol{c}_P$ are then dimensionless and approximately of the same magnitude.

Figure 2 shows measurements and approximation polynomials of a sample development pump from the automotive supplier HELLA GmbH & Co. KGaA. It shows head h, efficiency η and power P in dependency of discharge $\dot{V}$ and normalized head h/ω_{n}^2, efficiency η and normalized power P/ω_{n}^3 in dependency of normalized discharge $\dot{V}/\omega_{\mathrm{n}}$. The reference speed was arbitrarily chosen as $\omega_{\mathrm{ref}} = 5440\,\mathrm{rpm}$ and the reference density $\rho_{\mathrm{ref}} = 1000\,\mathrm{kg/m^3}$. Different colors refer to different speeds ω and measurement data are highlighted with markers. Approximation 1 (solid) is based on measurements at reference speed $\omega_{\mathrm{ref}} = 5440\,\mathrm{rpm}$ only. In contrast approximation 2 (dashed) is based on measurements at all eight speeds $\omega = 2040\,\mathrm{rpm}$ to $6800\,\mathrm{rpm}$. Figure 2 shows that a transformation based on affinity laws projects the head, efficiency and power maps well onto single curves, i.e. the homologeous quantities can be well expressed as a function of homologeous discharge $\dot{V}/\omega_{\mathrm{n}}$. This is especially true for the head h. In contrast applying affinity laws for power P overestimates the efficiency η for low speeds, but the results might still be accurate enough if off design power at low speed is of minor importance. Also note that the available torque data at those low speeds had only one significant digit.

If reference power P_{ref} is defined as $P_{\mathrm{ref}} = \rho_{\mathrm{ref}} g h_{\mathrm{ref}} \dot{V}_{\mathrm{ref}}/\eta_{\mathrm{ref}}$ one can derive that the efficiency:

$$\eta = \eta_{\mathrm{ref}} \frac{x + c_h x^2 - (1 + c_h) x^3}{c_{P,1} + c_{P,2} x + c_{P,3} x^2}, \tag{5}$$

where $x = \dot{V}_{\mathrm{n}}/\omega_{\mathrm{n}}$ is the homologeous discharge. Hence applying affinity laws yields efficiency $\eta \sim \dot{V}/\omega$, as already shown in Figure 2. Consequently according to affinity laws efficiency $\eta \sim \dot{V}/\sqrt{h}$, since head $h \sim \omega^2$. Therefor curves of constant efficiency η are parabolas $h = c(\eta)\dot{V}^2$, where $c(\eta)$ is a parameter, see e.g. (Sigloch 2013, sec. 9.2).

Figure 3 shows the approximated map of a sample development pump from the automotive supplier HELLA GmbH & Co. KGaA based on measurements at reference speed $\omega_{\mathrm{ref}} = 5440\,\mathrm{rpm}$ only (equivalent to approximation 1 in Figure 2) and extended using affinity laws. Solid lines thereby refer to the same curves of constant speed ω as displayed in Figure 2. The reference speed line $\omega = \omega_{\mathrm{ref}}$ is displayed as thick solid. Dashed lines refer to parabolas of constant efficiency η and numbers 0.2, 0.5, 0.8, 0.9, 0.95 and 1 refer to the ratio of efficiency to peak efficiency η/η_{ref}. The peak efficiency curve $\eta = \eta_{\mathrm{ref}}$ is displayed as thick dashed.

Even if the dashed parabolas of constant efficiency in Figure 3 represent the principal pump behavior, it is well known that the efficiency decreases for off design speeds $\omega > \omega_{\mathrm{ref}}$ and $\omega < \omega_{\mathrm{ref}}$, yielding the well known pump map shape, see e.g. (Sigloch 2013, fig. 9-14). This is not yet considered by the model, but by means of a simple modification the model could in the future be improved by a efficiency reduction coefficient $c(\omega) \leq 1$ reducing the peak efficiency of individual speed curves $\eta_{\mathrm{ref}}(\omega) = c(\omega)\eta_{\mathrm{ref}}(\omega_{\mathrm{ref}})$.

Figure 2 furthermore shows, that for (radial flow) centrifugal pumps quadratic polynomials are suitable for the approximation. However note, that quadratic polynomials may not be suitable to approximate pumps with higher specific speed, e.g. mixed flow or axial flow pumps, since their characteristics may differ considerably, see e.g. (Sigloch 2013, fig. 9-16). Specific speed is thereby defined as (Chaudhry 2014):

$$\omega_{\mathrm{specific}} = \omega \sqrt{\frac{\dot{V}}{\sqrt{gh}^3}}. \tag{6}$$

Note that the sample development pump from the automotive supplier HELLA GmbH & Co. KGaA has a specific speed of 0.406.

Figure 4 shows mean relative errors of the approximations as shown in Figure 2 for series of measurements at constant speed ω. Colors correspond to head h (blue), power P (red) and efficiency η (black). As in Figure 2 approximation 1 (solid) is also in Figure 4 based on measurements at reference speed $\omega_{\mathrm{ref}} = 5440\,\mathrm{rpm}$ only and approximation 2 (dashed) is based on measurements at all eight speeds $\omega = 2040\,\mathrm{rpm}$ to $6800\,\mathrm{rpm}$. Figure 4 shows that the mean relative error regarding head h is for all speeds about one order of

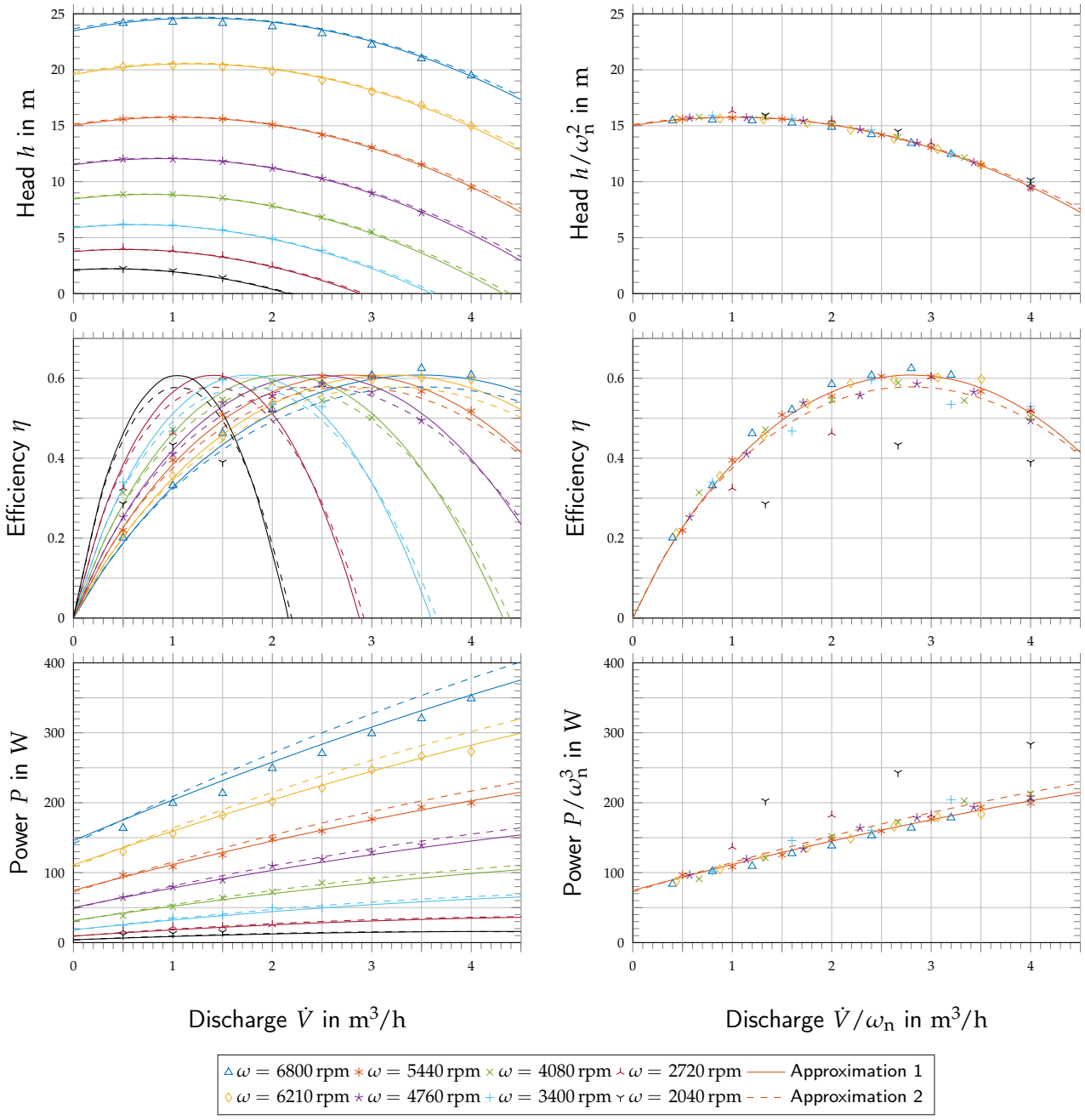


Figure 2. Measurements and approximation polynomials of a sample development pump from the automotive supplier HELLA GmbH & Co. KGaA.

magnitude smaller than the mean relative error regarding power P and efficiency η. The strongest outlier regarding head h is only 6.0 % and 4.0 % off for approximation 1 and 2 respectively, which is remarkable considering that the available head data at low speeds had only 2 significant digits. Including measurements at all eight speeds changes the mean error on average only slightly from $\mathrm{err}_{h,\mathrm{ap1}} = 1.5\,\%$ to $\mathrm{err}_{h,\mathrm{ap2}} = 1.3\,\%$ regarding head h, from $\mathrm{err}_{P,\mathrm{ap1}} = 8.2\,\%$ to $\mathrm{err}_{P,\mathrm{ap2}} = 8.7\,\%$ regarding power P, and from $\mathrm{err}_{\eta,\mathrm{ap1}} = 9.4\,\%$ to $\mathrm{err}_{\eta,\mathrm{ap2}} = 7.7\,\%$ regarding efficiency η, where the subscripts $(\bullet)_{\mathrm{ap1}}$ and $(\bullet)_{\mathrm{ap2}}$ refer to the first and second approximation respectively. This demonstrates that the approximation can not really be improved by including measurements at different speeds. However both approximations yield significant errors concerning power and efficiency of up to 40 % at low speeds, which highlights the need to extend the model as previously mentioned, if off design power and efficiency at low speeds matter.

2.3 Comparison of centrifugal pumps

A convenient way to obtain the six head and power coefficients $\boldsymbol{c}_h$ and $\boldsymbol{c}_P$ can be achieved by defining:

DOI 10.3384/ecp218325

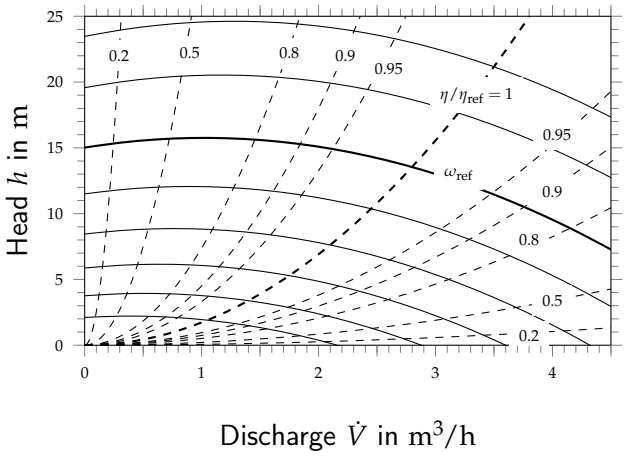


Figure 3. Map of a sample development pump from the automotive supplier HELLA GmbH & Co. KGaA.

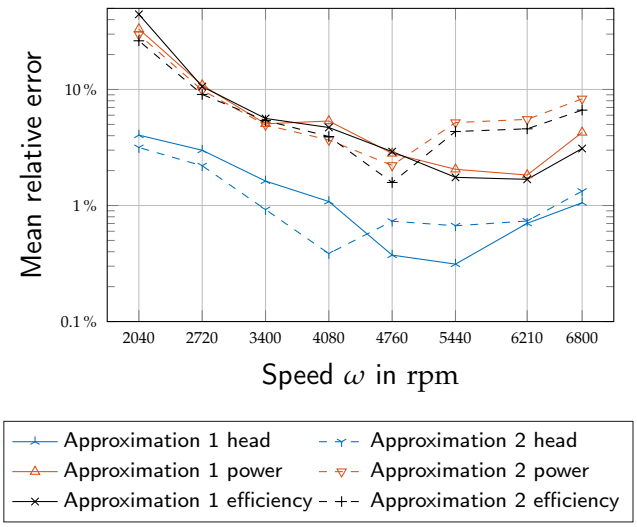


Figure 4. Mean relative errors of the approximations as shown in Figure 2 for series of measurements at constant speed ω.

1. the peak efficiency η_{ref}
2. the head at peak efficiency h_{ref}
3. the discharge at peak efficiency $\dot{V}_{\text{ref}}$
4. the normalized head at zero discharge $h_{0,\text{n}}$
5. the normalized discharge at zero head $\dot{V}_{0,\text{n}}$
6. the normalized power at zero discharge $P_{0,\text{n}}$

Items 2 to 5 provide three equations for the head coefficients $\boldsymbol{c}_h$, which yield:

$$c_{h,0} = h_{0,\text{n}}\,, \tag{7}$$

$$c_{h,1} = \frac{\dot{V}_{0,\text{n}}}{\dot{V}_{0,\text{n}} - 1} - h_{0,\text{n}} \frac{\dot{V}_{0,\text{n}} + 1}{\dot{V}_{0,\text{n}}}\,, \tag{8}$$

$$c_{h,2} = \frac{h_{0,\text{n}}}{\dot{V}_{0,\text{n}}} - \frac{1}{\dot{V}_{0,\text{n}} - 1}\,, \tag{9}$$

Table 1. Parameters of approximation 1 (ap1) and approximation 2 (ap2) of the sample development pump from the automotive supplier HELLA GmbH & Co. KGaA. as shown in Figure 2. The reference speed $\omega_{\text{ref}} = 5440\,\text{rpm}$ and the reference density $\rho_{\text{ref}} = 1000\,\text{kg/m}^3$.

	h_{ref} in m	$\dot{V}_{\text{ref}}$ in m³/h	η_{ref} in %	$h_{0,\text{n}}$	$\dot{V}_{0,\text{n}}$	$P_{0,\text{n}}$
ap1	13.5	2.80	60.8	1.11	2.06	0.440
ap2	13.6	2.80	57.9	1.11	2.09	0.403

Table 2. Mean, standard deviation, minimum and maximum values of normalized head at zero discharge $h_{\text{n},0}$, normalized discharge at zero head $\dot{V}_{\text{n},0}$ and normalized power at zero flow $P_{\text{n},0}$ for 18 WILO pumps available in the Modelica Buildings Library (Wetter et al. 2014) .

	mean	std	min	max
$h_{0,\text{n}}$	1.273	±0.128	1.101	1.516
$\dot{V}_{0,\text{n}}$	1.946	±0.087	1.784	2.090
$P_{0,\text{n}}$	0.499	±0.099	0.372	0.677

and items 1 and 6 provide three more equations for the power coefficients $\boldsymbol{c}_P$, which yield:

$$c_{P,0} = P_{0,\text{n}}\,, \tag{10}$$

$$c_{P,1} = -2P_{0,\text{n}} + c_{h,0} - c_{h,2}\,, \tag{11}$$

$$c_{P,2} = P_{0,\text{n}} + c_{h,1} + 2c_{h,2}\,. \tag{12}$$

Table 1 summarizes the parameters of approximation 1 (ap1) and approximation 2 (ap2) of the sample development pump from the automotive supplier HELLA GmbH & Co. KGaA. as shown in Figure 2. Those six coefficients (plus reference speed $\omega_{\text{ref}} = 5440\,\text{rpm}$ and the reference density $\rho_{\text{ref}} = 1000\,\text{kg/m}^3$) are sufficient to fully describe the pump characteristic.

The benefit of this approach is, that head at design point h_{ref} and discharge at design point $\dot{V}_{\text{ref}}$ may vary for different pumps according to their requirements. For example the design points of the 18 WILO pumps available in the Modelica Buildings Library differ significantly: their peak head h_{ref} ranges from 1.4 m up to 22 m, their peak discharge ranges $\dot{V}_{\text{ref}}$ from 2.9 m³/h up to 73 m³/h, their peak power P_{ref} ranges from 27 W up to 4.2 kW and their peak efficiency η_{ref} ranges from 24 % up to 77 %. However the three coefficients: 1) normalized head at zero discharge $h_{\text{n},0}$, normalized discharge at zero head $\dot{V}_{\text{n},0}$ and normalized power at zero flow $P_{\text{n},0}$ differ much less as listed in Table 2. Hence normalized pump characteristics differ only to a medium degree as shown in Figure 5 for 18 WILO pumps available in the Modelica Buildings Library (Wetter et al. 2014). Colors in Figure 5 refer to different pumps and the pump characteristics based on mean parameters as

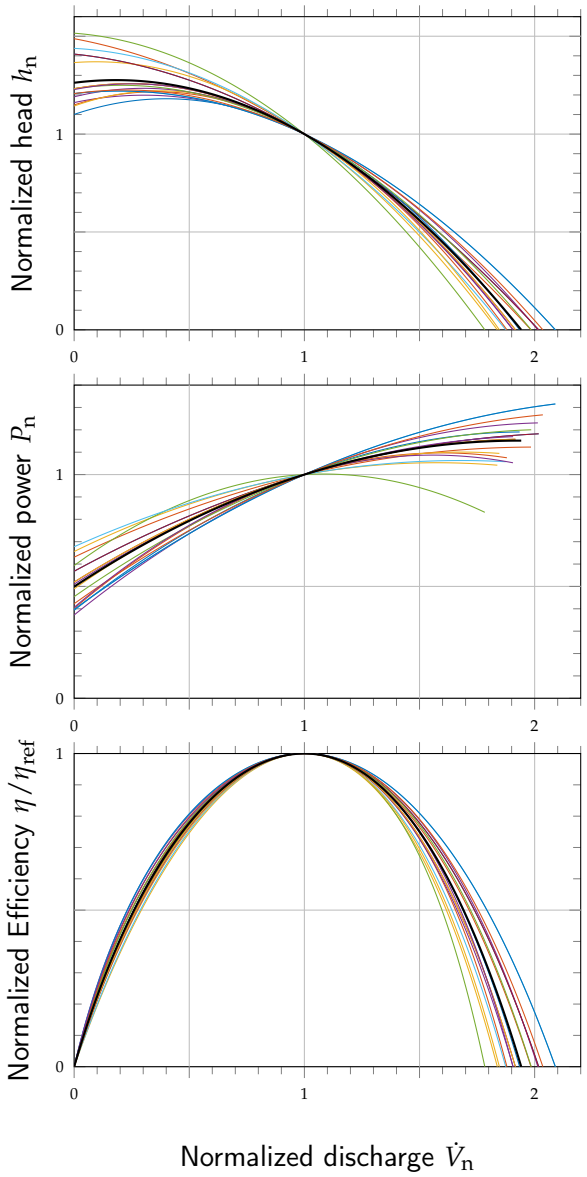


Figure 5. Comparison of 18 WILO pumps available in the Modelica Building Library (Wetter et al. 2014).

listed in Table 2 are marked thick black. The similarity is also based on the fact that the specific speed only varies from 0.42 to 1.21.

Note that (Sigloch 2013, sec. 9.2) also mentions a range of normalized power at zero flow of $P_{0,n} = 0.4$ to 0.6. The peak efficiency η_{ref} is also rather easy to predict where smaller and/or cheaper pumps usually reach lower peak efficiencies and larger, more efficient pumps can reach up to 90 % peak efficiency.

3 Example: Instationary behavior

To extend the pump map to negative discharge $\dot{V} < 0$, we adopt the head characteristic of Equation 3 by replacing the square term $\dot{V}_n^2$ by $\dot{V}_n|\dot{V}_n|$:

$$h_n = c_{h,0}\omega_n^2 + c_{h,1}\omega_n\dot{V}_n + c_{h,2}\dot{V}_n|\dot{V}_n|. \qquad (13)$$

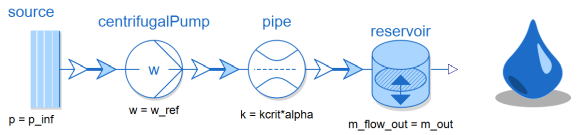


Figure 6. Exemplary system to investigate instationary pump behavior.

Note that this modification is a rather simple qualitative extension for negative discharge $\dot{V} < 0$ that is still able to predict the principle physical behavior as the quadratic term originates from friction that is $\sim \dot{V}|\dot{V}|$, see (Sigloch 2013, sec. 9.2). In order to quantitatively cover all four quadrants of pump operation, it might be worthwhile to integrate the approach presented in (Chaudhry 2014, sec. 4.3) into the TFS in the future.

Figure 6 shows an exemplary system to investigate instationary pump behavior. An open reservoir with a constant discharge `m_flow_out` $= \rho\dot{V}_{out}$ is fed through a flow resistance by a centrifugal pump at constant speed `w_ref` $= \omega_{ref}$. Depending on the constant discharge $\dot{V}_{out}$ and on the flow resistance the system may yield a limit cycle known as surge, such that the system does not reach a steady-state (Sigloch 2013, sec. 9.2). In this case the system can only be simulated when considering dynamic effects.

In the following we use head at zero discharge as new reference head $\hat{h}_{ref}$ and discharge at zero head as new reference discharge $\hat{V}_{ref}$:

$$\hat{h}_{ref} = h_{ref}c_{h,0}, \qquad (14)$$

$$\hat{V}_{ref} = \frac{\dot{V}_{ref}}{2c_{h,2}}\left(-c_{h,1} - \sqrt{c_{h,1}^2 - 4c_{h,0}c_{h,2}}\right), \qquad (15)$$

which simplifies Equation 13:

$$\hat{h}_n = \omega_n^2 + \hat{c}\omega_n\hat{\dot{V}}_n - (1+\hat{c})\hat{\dot{V}}_n|\hat{\dot{V}}_n|, \qquad (16)$$

where $\hat{h}_n = h/\hat{h}_{ref}$ and $\hat{\dot{V}}_n = \dot{V}/\hat{\dot{V}}_{ref}$ are normalized head and discharge and $\hat{c}$ is a coefficient:

$$\hat{c} = c_{h,1}\frac{h_{ref}\hat{\dot{V}}_{ref}}{\hat{h}_{ref}\dot{V}_{ref}}. \qquad (17)$$

The limit cycle can only occur when the discharge $\dot{V}_{out}$ is less than the critical discharge at maximum head $\dot{V}_{out} \leq \dot{V}_{crit}$. The corresponding critical normalized discharge can be derived based on the condition $\mathrm{d}\hat{h}_n/\mathrm{d}\hat{\dot{V}}_n = 0$ at reference speed $\omega_n = 1$ as:

$$\hat{\dot{V}}_{n,crit} = \frac{\hat{c}}{2(1+\hat{c})}. \qquad (18)$$

We consider a constant reservoir discharge $\dot{V}_{out} = 1/2\dot{V}_{crit}$ such that surge may occur depending on the system characteristics.

DOI 10.3384/ecp218325

Surge occurs, if the pump- and system characteristics do not intersect anymore. In the limiting case that means that pump- and system characteristics are tangential for the constant reservoir discharge $\dot{V}_{\text{out}}$:

$$\left.\frac{\mathrm{d}\hat{h}_{\text{p,n}}}{\mathrm{d}\hat{V}_{\text{n}}}\right|_{\hat{V}_{\text{out,n}}} = \left.\frac{\mathrm{d}\hat{h}_{\text{sys,n}}}{\mathrm{d}\hat{V}_{\text{n}}}\right|_{\hat{V}_{\text{out,n}}}, \tag{19}$$

where $\hat{h}_{\text{p,n}} = h_{\text{p}}/\hat{h}_{\text{ref}}$, $\hat{h}_{\text{sys,n}} = h_{\text{sys}}/\hat{h}_{\text{ref}}$ and $\hat{V}_{\text{out,n}} = \dot{V}_{\text{out}}/\hat{V}_{\text{ref}}$ are the normalized pump head, system head, and reservoir discharge.

To be able to derive analytical results later on, we assume a constant friction factor k for the system characteristics. Furthermore the normalized friction factor $\hat{k}$ is introduced:

$$\Delta p_{\text{sys}} = k\dot{m}\,|\dot{m}|\,, \tag{20}$$

$$\implies \hat{h}_{\text{sys,n}} = \hat{k}\hat{V}_{\text{n}}\left|\hat{V}_{\text{n}}\right|, \quad \hat{k} = k\frac{\rho\hat{V}_{\text{ref}}^2}{g\hat{h}_{\text{ref}}}. \tag{21}$$

Then Equation 19 yields the critical normalized friction factor $\hat{k}_{\text{crit}}$:

$$\hat{k}_{\text{crit}} = \frac{\hat{c}}{2|\hat{V}_{\text{out,n}}|} - (1+\hat{c}), \tag{22}$$

such that a normalized friction factor $\hat{k} \leq \hat{k}_{\text{crit}}$ yields surge.

Figure 7 shows simulation results of the exemplary system (Figure 6) for five different friction factors $k = \alpha k_{\text{crit}}$ with coefficient $\alpha = 0$, 0.5, 0.98, 1.02 and 2 (different colors). The ambient pressure `p_inf` $= p_\infty = 1\,\text{bar}$, the cross sectional area of the reservoir $A = 100\,\text{cm}^2$, the initial height of the reservoir $h_{\text{R},0} = 13\,\text{m}$, the initial pump discharge $\dot{V}(0) = 0\,\text{m}^3/\text{s}$, the density and reference density are $\rho = \rho_{\text{ref}} = 1000\,\text{kg}/\text{m}^3$ and the overall inertance $L = (4 \times 0.01)/\text{m}$. The constant reservoir discharge $\dot{V}_{\text{out}} = 0.5\dot{V}_{\text{crit}} = 0.512\,\text{m}^3/\text{h}$ yields a critical friction factor `kcrit` $= k_{\text{crit}} = 8.92 \times 10^4\,\text{Pa}\,\text{s}^2/\text{kg}^2$. The example is a second order system of nonlinear differential equations:

$$L\rho\frac{\mathrm{d}\dot{V}}{\mathrm{d}t} = \Delta p_{\text{p}} - (\Delta p_{\text{sys}} + \rho g h_{\text{R}}), \tag{23}$$

$$A\frac{\mathrm{d}h_{\text{R}}}{\mathrm{d}t} = \dot{V} - \dot{V}_{\text{out}}. \tag{24}$$

Note that the implicit part is linear (it is even diagonal and regular for $L, A \neq 0$), so it is a trivial task to solve the system for its states $\boldsymbol{x} = (\dot{V}, h_{\text{R}})^{\intercal}$, which is generally the case for the TFS.

Figure 7 shows that discharge $\dot{V}$ increases quickly from rest and reaches a quasi-static equilibrium shortly after initialization, since the flow rate dynamics limited by the inertance L are a lot faster than the reservoir dynamics and hence often the influence of the flow rate dynamics can be neglected and the system is quasi-static with respect to the flow rate dynamics.

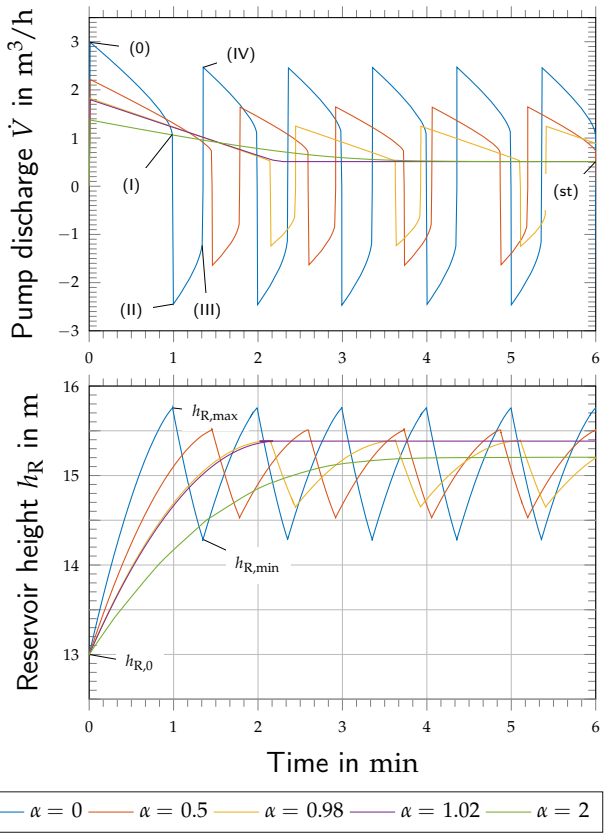


Figure 7. Instationary pump simulation for different pipe characteristics.

The initial quasi-static discharge $\dot{V}_{0+}$ depends on the system characteristics and hence on the coefficient α. The higher the coefficient α, i.e. the more friction, the lower the initial quasi-static discharge $\dot{V}_{0+}$. Afterwards the reservoir height h_{R} increases (time $t \leq (1 \text{ to } 2)\,\text{min}$) and the discharge $\dot{V}$ decreases. For coefficients $\alpha = 1.02$ and 2 the pump discharge $\dot{V}$ decreases until it balances the reservoir discharge $\dot{V}_{\text{out}}$ and the system reaches steady-state (st). The larger the coefficient α, i.e. the more friction, the lower is the steady-state reservoir height $h_{\text{R,st}}$.

In contrast for coefficients $\alpha = 0$, 0.5 and 0.98 the system yields a limit cycle, with critical points (I), (II), (III) and (IV). Note that the critical discharges $\dot{V}_{\text{I}}$ to $\dot{V}_{\text{IV}}$, the critical minimum and maximum reservoir heights $h_{\text{R,min}}$ and $h_{\text{R,max}}$, the periods of the positive and negative limit cycle T_+ and T_- and the overall period $T = T_+ + T_-$ depend on the coefficient α.

Figure 8 illustrates the quasi-static system behavior, defined by the intersection of the system characteristics (dotted) and the pump characteristics (solid). The three subplots and their respec-

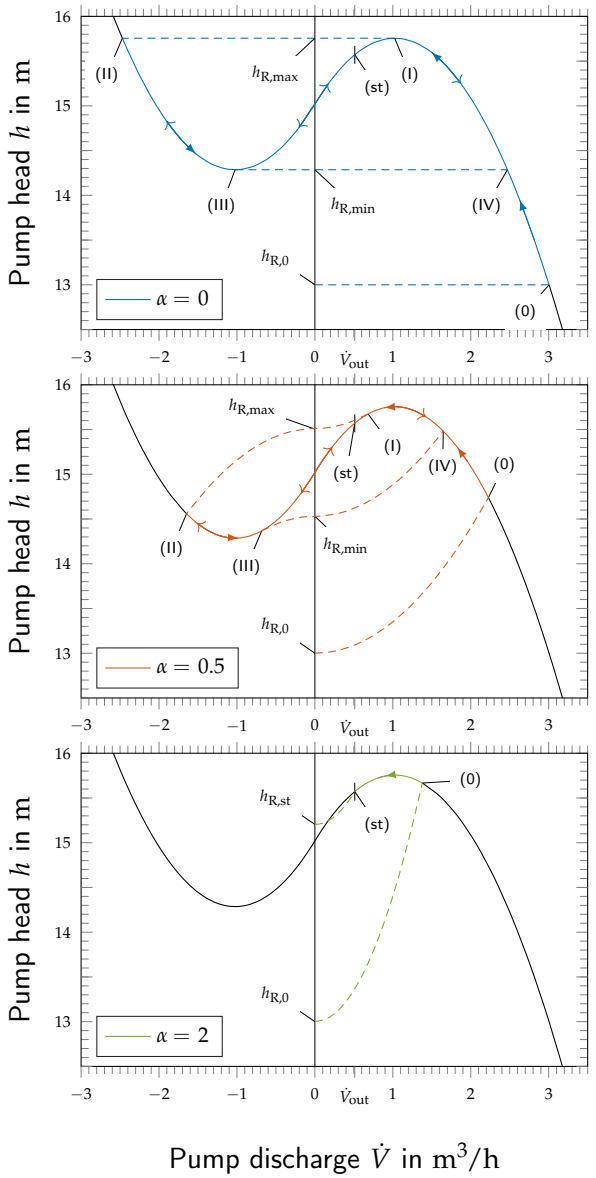


Figure 8. System behaviour for different pipe characteristics.

tive colors correspond to the three values of the coefficient $\alpha = 0$, 0.5 and 2. The y-axis intercept of the system characteristics represents the reservoir height h_R, and the curvature is proportional to the coefficient α. In contrast, all three subplots share the same pump characteristic since the pump speed ω is identical.

Initially, it can be observed that different values of α lead to different quasi-static initial discharges (0), resulting in different initial net inflows to the reservoir $\dot{V}_{0+} - \dot{V}_{out}$. Consequently, the reservoir height h_R increases more rapidly for $\alpha = 0$ compared to $\alpha = 2$. As the reservoir height h_R increases, the system characteristics shift vertically. The new intersection between the shifted system characteristics and the pump characteristics results in a decreasing pump discharge $\dot{V}$, thereby slowing the increase of the reservoir height h_R. This quasi-static evolution is indicated by filled arrows (⟶).

For $\alpha = 2$, the system eventually reaches the steady-state point (st), where the net inflow $\dot{V} - \dot{V}_{out}$ becomes zero, and the reservoir height h_R remains constant. This point is stable because any small deviation would still lead to an intersection between system and pump characteristics, generating a restoring effect that drives the system back to steady state.

In contrast, for $\alpha = 0$ and 0.5, the system reaches the critical point (I) before achieving steady state (st). At this point, the net reservoir inflow $\dot{V} - \dot{V}_{out}$ remains positive, and the reservoir height h_R continues to rise. However, beyond the critical point (I), no quasi-static intersection exists. The reservoir pressure decelerates the fluid, reducing the pump discharge $\dot{V}$, which in turn further decelerates and eventually reverses the flow. A new quasi-static equilibrium is reached only at the reversed pump discharge (II). During the rapid transition from (I) to (II), the pump (solid) and system (dashed) characteristics remain valid, and the pressure difference between them causes a sharp deceleration of the fluid. Meanwhile, the reservoir height remains approximately constant $h_R = h_{R,max}$. This fast transition is marked with arrows (⟶).

At point (II), the net inflow $\dot{V} - \dot{V}_{out}$ becomes negative, causing the reservoir height h_R to decrease. The intersection between the shifted system characteristics and the pump characteristics now evolves quasi-statically again (⟶). However, before reaching steady state (st), the system encounters the next critical point (III). At this point, the reservoir height h_R becomes too low to sustain reversed flow through the pump. The fluid is then accelerated by the pump, increasing the discharge $\dot{V}$, which further accelerates the fluid until a new quasi-static intersection is reached at point (IV). During the rapid transition from (III) to (IV), the pump (solid) and system (dashed) characteristics are again satisfied. The resulting pressure difference drives the flow reversal, while the reservoir height remains approximately constant $h_R = h_{R,min}$. This fast transition is indicated by arrows (⟶). The limit cycle then repeats, preventing the system from reaching the steady-state point (st).

3.1 Analytic verification

The behavior is well known, see e.g. (Sigloch 2013, sec. 9.2). To confirm the previous simulation results, analytic results are derived in the following.

First note that the critical discharges $\dot{V}_I$ to $\dot{V}_{IV}$ and the maximum and minimum reservoir heights $h_{R,max}$ and $h_{R,min}$ can be determined analytically by evaluating Equation 19 for given

normalized friction factor $\hat{k}$, which yields:

$$\hat{V}_{\mathrm{I,n}} = \frac{\hat{c}}{2(1+\hat{c}+\hat{k})}, \tag{25}$$

$$\hat{h}_{\mathrm{R,max,n}} = 1 + \hat{c}\hat{V}_{\mathrm{I,n}} - (1+\hat{c}+\hat{k})\hat{V}_{\mathrm{I,n}}^2, \tag{26}$$

$$\hat{V}_{\mathrm{II,n}} = \frac{\hat{c} + \sqrt{\hat{c}^2 - 4(1+\hat{c}+\hat{k})\left(1-\hat{h}_{\mathrm{R,max,n}}\right)}}{-2(1+\hat{c}+\hat{k})}, \tag{27}$$

$$\hat{V}_{\mathrm{III,n}} = -\hat{V}_{\mathrm{I,n}}, \tag{28}$$

$$\hat{V}_{\mathrm{IV,n}} = -\hat{V}_{\mathrm{II,n}}, \tag{29}$$

$$\hat{h}_{\mathrm{R,min,n}} = 1 - (\hat{h}_{\mathrm{R,max,n}} - 1). \tag{30}$$

Solving the second order system of nonlinear differential equations (Equation 23 and Equation 24) analytically is not trivial, but neglecting the flow rate dynamics, i.e. for zero interance $L = 0$, Equation 23 becomes analytic. Then one combine Equation 23 and Equation 24 and derive the first order nonlinear differential equation for the normalized reservoir height $\hat{h}_{\mathrm{R,n}}$:

$$\frac{\mathrm{d}\hat{h}_{\mathrm{R,n}}}{\mathrm{d}\tau} = -\hat{V}_{\mathrm{out,n}} \pm \hat{V}_{\mathrm{I,n}}\left(1 + \sqrt{1 \pm \gamma(1-\hat{h}_{\mathrm{R,n}})}\right), \tag{31}$$

where $\tau = t/t_{\mathrm{ref}}$ is the normalized time, $t_{\mathrm{ref}} = A\hat{h}_{\mathrm{ref}}/\hat{V}_{\mathrm{ref}}$ is the reference time, $\gamma = 4(1+\hat{c}+\hat{k})/\hat{c}^2$ is a coefficient and $(\pm)$ refers to the positive branch $\dot{V} > 0$ and negative branch $\dot{V} < 0$ of the limit cycle respectively. The first order differential Equation 31 can be solved analytically for each branch of the limit cycle, and one can obtain an implicit solution of the normalized reservoir height $\hat{h}_{\mathrm{R,n}}$ depending on the normalized time τ:

$$\tau = f_{\pm}(\hat{h}_{\mathrm{R,n}}) - f_{\pm}(\hat{h}_{\mathrm{R,0,n}}), \tag{32}$$

$$f_{\pm} = -\frac{2}{\hat{V}_{\mathrm{I,n}}^2\gamma}\Big(\hat{V}_{\mathrm{I,n}}g_{\pm} + \\ + (\pm\hat{V}_{\mathrm{out,n}} - \hat{V}_{\mathrm{I,n}})\ln\left|\mp\hat{V}_{\mathrm{out,n}} + \hat{V}_{\mathrm{I,n}}(1+g_{\pm})\right|\Big), \tag{33}$$

$$g_{\pm} = \sqrt{1 \pm \gamma(1-\hat{h}_{\mathrm{R,n}})}, \tag{34}$$

where f and g are auxiliary variables and $\hat{h}_{\mathrm{R,0,n}}$ is the initial normalized reservoir height. This also enables the calculation of the periods corresponding to the positive and negative branches of the limit cycle $T_{\pm}$:

$$T_+/t_{\mathrm{ref}} = f_+(\hat{h}_{\mathrm{R,max,n}}) - f_+(\hat{h}_{\mathrm{R,min,n}}), \tag{35}$$

$$T_-/t_{\mathrm{ref}} = f_-(\hat{h}_{\mathrm{R,min,n}}) - f_+(\hat{h}_{\mathrm{R,max,n}}), \tag{36}$$

and the overall period of the limit cycle $T = T_+ + T_-$, where $g_+(\hat{h}_{\mathrm{R,max,n}}) = g_-(\hat{h}_{\mathrm{R,min,n}}) = \sqrt{2}$ and $g_+(\hat{h}_{\mathrm{R,min,n}}) = g_-(\hat{h}_{\mathrm{R,max,n}}) = 0$.

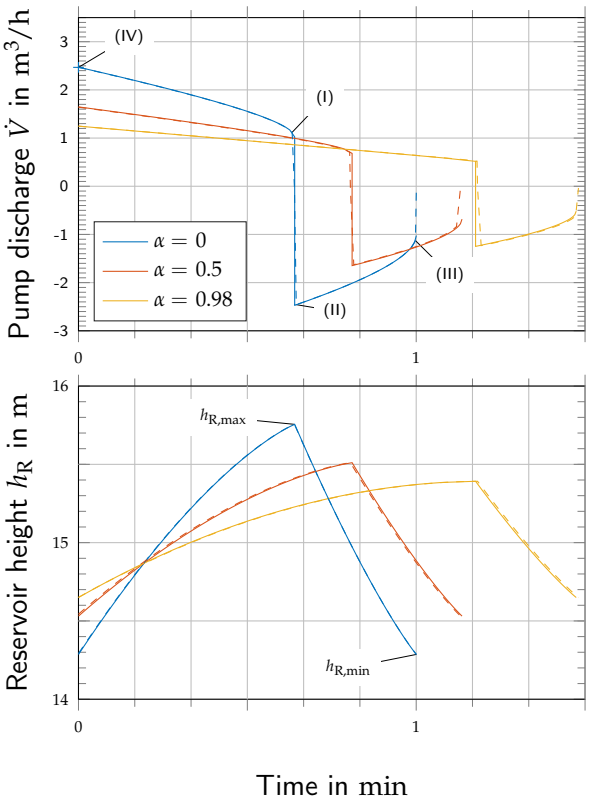


Figure 9. Comparison of analytical and dynamic simulation results for one limit cycle.

Figure 9 compares the analytic solution based on eqs. (32) to (34) (solid lines) with the numerical simulation results (dashed lines) of the first limit cycle as shown in Figure 7 that starts for $\alpha = 0$ at about 1 min 20 s, for $\alpha = 0.5$ at about 1 min 50 s and for $\alpha = 0.98$ at about 2 min 30 s. Colors refer to coefficients $\alpha = 0$, 0.5 and 0.98. Both results agree very well, i.e. Figure 9 verifies the analytic solution and the simulation model.

Figure 9 demonstrates, that the effect of the fluid inertia L is small, however considering its effect is crucial to solve the system. It is well known that nonlinear equation solvers may fail at the limit points (I) and (III), since a slight change in reservoir height $\dot{h}_{\mathrm{R}}$ causes a large change in discharge $\dot{V}$ if Equation 23 is solved algebraically ($L = 0$). There is no certainty that the solver of the nonlinear algebraic equation will converge at this step change in discharge. This is especially true for larger algebraic systems of nonlinear equations. When considering the effect of the fluid inertia the nonlinear algebraic equation turns into the differential equation. The implicit part of the system of differential equations is linear and it is a trivial task to solve for its states. From a physical point of view the dynamic equilibrium always exists and the implicit time integration method, which also has to solve a nonlinear equation system at every time step, is guided into the correct direction in

which the system will evolve. To be able to use an explicit time integration scheme the inertance L and the time step size have to match. In the future it might be promising to increase the inertance L up to a certain limit to enable real time simulation.

The example showed, that the simulation of limit cycles are possible with the TFS by default, since it takes the fluid inertia into account and avoids the common oversimplification of requiring quasi-static behavior. This actually simplifies the component models, since numerical modifications, as presented in (Wetter 2013), are no longer necessary. Note that while analytic calculations may help to verify simulation models for simple examples, modifications are very likely to cause analytic calculations to be neither possible nor feasible anymore. In these cases numerical simulation can still provide the opportunity to gain insight into system dynamics, if the simulation models proofed to be based on the correct physics.

4 Conclusion and Outlook

The proposed model for a centrifugal pump has proven to be a numerically robust model. The possibility to work with real pump behavior allows for instationary simulations. This is especially important when fault scenarios must be simulated. This also allows for a robust initialization of the models, as the model behavior follows the behavior from the real world. Fitting the pump data is simple, as no special care has to be taken to avoid instationary solutions, as the model can handle these well. An example of an instationary pump model has been validated against the theoretical solutions with an extremely well correlation. Furthermore, the models have been validated against data from measurements. The models have been validated for pump head, power and efficiency over a wide range of pump speeds: The pump head fit at all speeds stays very good (lower than 1 % relative error for speeds around ω_{ref}, lower than 3 % relative error at $\omega < \omega_{\text{ref}}/2$). The pump power fits the reference data well around ω_{ref} with errors of approximately 2 %. Even at $\omega = \omega_{\text{ref}}/2$ the error is smaller than 10 %. However at speeds below $\omega < \omega_{\text{ref}}/2$ the error quickly grows to around 40 %. A similar behavior can be seen for the pump efficiency: at $\omega_{\text{ref}}/2 < \omega < 1.25\omega_{\text{ref}}$ the relative error stays below 5.5 %, quickly growing to 45 % below half the reference speed. Hence it might be worth considering a decreasing peak efficiency at off design speeds as an extensions to similarity laws to obtain the well known pump characteristics. To further improve the model and quantitatively cover all four quadrants of pump operations, the approach presented in (Chaudhry 2014, sec. 4.3) is planned to be integrated into the TFS in the future.

Acknowledgements

This work was carried out as part of the European project **TheMa4HERA**. The project is supported by the Clean Aviation Joint Undertaking and its members. Funded by the European Union under Grant Agreement No. 101102008.



References

Casella, Francesco et al. (2006). "The Modelica Fluid and Media library for modeling of incompressible and compressible thermo-fluid pipe networks". In: *Proceedings of the 5th international modelica conference*, pp. 631–640.

Chaudhry, M. Hanif (2014). *Applied Hydraulic Transients*. 3rd ed. Springer New York. DOI: https://doi.org/10.1007/978-1-4614-8538-4.

Fu, Hongxiang, David Blum, and Michael Wetter (2023). "Fan and Pump Efficiency in Modelica based on the Euler Number". In.

Gülich, Johann Friedrich (2010). *Kreiselpumpen: Handbuch für Entwicklung, Anlagenplanung und Betrieb*. 3rd ed. Springer. DOI: 10.1007/978-3-642-05479-2.

Kallesøe, Carsten (2005). "Fault detection and isolation in centrifugal pumps". PhD thesis. Denmark: Aalborg University. URL: https://vbn.aau.dk/en/publications/fault-detection-and-isolation-in-centrifugal-pumps.

Karassik, Igor J. et al., eds. (2007). *Pump Handbook*. 4th ed. McGraw-Hill. ISBN: 978-0071460446.

Richter, Christoph C (2008). "Proposal of new object-oriented equation-based model libraries for thermodynamic systems". PhD thesis. Braunschweig, Techn. Univ.

Sigloch, Herbert (2013). *Strömungsmaschinen: Grundlagen und Anwendungen*. 8th ed. Carl Hanser Verlag GmbH Co KG. ISBN: 978-3-446-47677-6.

Stepanoff, Alexey Joakim (1957). *Centrifugal and Axial Flow Pumps: Theory, Design, and Application*. 2nd ed. John Wiley & Sons.

Trefethen, Lloyd N. (2019). "Chapter 13. Equispaced Points, Runge Phenomenon". In: *Approximation Theory and Approximation Practice, Extended Edition*. SIAM. Chap. 13, pp. 95–102. DOI: 10.1137/1.9781611975949.ch13.

Wetter, Michael (2013). "Fan and pump model that has a unique solution for any pressure boundary condition and control signal". In: *Building Simulation 2013*. Vol. 13. IBPSA, pp. 3505–3512.

Wetter, Michael et al. (2014). "Modelica Buildings library". In: *Journal of Building Performance Simulation* 7.4, pp. 253–270. DOI: 10.1080/19401493.2013.765506.

Zimmer, Dirk (2020). "Robust object-oriented formulation of directed thermofluid stream networks". In: *Mathematical and Computer Modelling of Dynamical Systems* 26.3, pp. 204–233. DOI: 10.1080/13873954.2020.1757726.

Zimmer, Dirk, Michael Meißner, and Niels Weber (2022). "The DLR ThermoFluid Stream Library". In: *Electronics* 11.22. ISSN: 2079-9292. DOI: 10.3390/electronics11223790.

Frost/Defrost Models for Air-Source Heat Pumps with Retained Water Refreezing Considered

Jiacheng Ma[1] Matthis Thorade[2]

[1]Modelon Inc., USA, jiacheng.ma@modelon.com
[2]Modelon Deutschland GmbH, Germany, matthis.thorade@modelon.com

Abstract

Cyclic frosting and defrosting operations constitute a common characteristic of air-source heat pumps in cold climates during winter. Simulation models that can capture simultaneous heat and mass transfer phenomena associated with frost/defrost behaviors and their impact on the overall heat pump system performance are of critical importance to improved controls of heat delivery and frost mitigation. This paper presents a novel frost formulation using an enthalpy method to systematically capture all phase-change behaviors including frost formation and melting, retained water refreezing and melting, and water drainage during cyclic frosting and defrosting operations. A Fuzzy modeling approach is proposed to smoothly switch source terms when evaluating the dynamics of frost and water mediums for numerical robustness. The proposed frost/defrost model is incorporated into a flat-tube outdoor heat exchanger model of an automotive heat pump system model to investigate system responses under cyclic operations of frosting and reverse-cycle defrosting.

Keywords: heat pump, reverse-cycle defrost, retained water, transient simulation, mode switch

1 Introduction

Cyclic frost accumulation and removal on the outdoor coil surface of air-source heat pumps are common in winter operations. As the system performance degrades due to blockage of free air flow passages and increased thermal resistance between coil walls and air streams, periodic defrost cycles are necessary to ensure consistent heat delivery and safe operations of the system. In practice, reverse-cycle defrost is among the most common strategies to implement by reversing the thermodynamic cycle to melt the accumulated frost. After that, the melted water can be evaporated as the coil continues being heated and can simultaneously drain from the coil surfaces due to gravity. Once the defrost cycle completes according to feedback controls or simply time-based controls, the refrigerant flow is reversed again to restore nominal heating operations. However, in case of an insufficient defrost operation, a portion of the melted liquid water is retained on the coil surface owing to surface tensions, which subsequently refreezes as solid ice when the system switches back to the heating mode where the outdoor coil operates as an evaporator. As a consequence, the next heating cycle can suffer from more significant performance degradation due to the presence of the ice layer as an additional thermal resistance compared to the previous heating cycle, which can lead to a shorter heating period before a necessary defrost cycle (Zhang, Li, and Hrnjak 2024). The issue is particularly important to operations of microchannel heat exchangers commonly adopted in automobile applications (including electric vehicles) whose performance is more sensitive to air flow blockage and thermal insulation due to the compact configuration.

Simulation tools that can capture the complicated heat and mass transfer phenomena of cyclic frosting-defrosting operations can be critically useful for the design and evaluation of improved defrost controls for air-source heat pumps that can mitigate frost/ice on the outdoor coil without excessively heating the coil during defrost cycles. Almost all existing frost/defrost models for heat pump systems focused on switches between frost formation and melting behaviors, omitting the impacts of retained water refreezing and melting on transients of cyclic frosting and defrosting (Qiao, Aute, and Radermacher 2018; Ma et al. 2023; Alam et al. 2024). Westhaeuser et al. (2023) derived a numerical model that accounted for frost formation and melting as well as retained water statuses in a lumped control volume. However, no general switching scheme was implemented for transitions between stages of frost and water processes, which hinders its applicability to describe heat pump transients under realistic mode switches between frosting and defrosting.

This paper presents a novel frost model that systematically captures all phase-change behaviors of frost and retained water/ice in a multi-phase multi-substance control volume using an enthalpy method. A Fuzzy modeling approach is proposed to smoothly switch source terms of governing equations to evaluate dynamics of the control volume based on underlying physical processes. The derived model is incorporated into a discretized flat-tube outdoor heat exchanger model of an automotive heat pump system. The system model is then simulated to investigate transients under cyclic operations of frosting/heating and reverse-cycle defrosting with retained water behaviors taken into consideration.

The remainder of this paper is organized as follows:

Section 2 describes model development for frost and retained water processes; Section 3 presents a discretized outdoor heat exchanger model incorporated with frost behaviors, and other component models to complete a heat pump system model; simulation results of the developed system model under cyclic frosting and defrosting are reported in Section 4, followed by conclusions summarized in Section 5.

2 Frost/defrost model

Frost formation and melting involve simultaneous heat and mass transfer. A common practice to model both processes relies on evaluating mass and energy balances of a frost control volume to characterize its dynamics which can be represented by a frost layer thickness and a lumped density (D. Kim, Braun, and Ramaraj 2018). While a control volume for water/ice can be constructed in a similar manner and interconnected with the frost volume to capture retained water/ice behaviors, significant model complexities are introduced as different phase presences and interactions between these control volumes need to be captured when the operating mode changes. Figure 1 depicts typical processes that occur on coil wall surfaces under cyclic frosting and defrosting, that consist of

- Process 1: frost formation on dry coils;
- Process 2: frost melting and water drainage;
- Process 3: retained water vaporization and drainage;
- Process 4: retained water refreezing and frost formation on ice.

It is important to note that transitions between these processes can follow different orders. Clearly, each stage during the process corresponds to different heat and mass transfer descriptions for control volumes and interfaces. Moreover, additional iteration variables (e.g., interface temperature or heat flux) can be introduced when modeling frost and water/ice mediums with individual control volumes. As a consequence, the overall model robustness deteriorates.

This work aims to develop a frost model with retained water/ice considered in a uniform model structure. A lumped film control volume that integrates frost, liquid water, and solid ice is developed using an enthalpy method (Alebrahim and Sherif 2002). The film dynamics under various operating modes and phase presences are captured by switching source terms of mass and energy balances.

2.1 A lumped film volume model

An advantage of the enthalpy method to model multiple mediums with phase change is that phase presences can be determined using the total enthalpy and mass fractions of each medium, eliminating the evaluation of internal heat transfer between them explicitly. Consider a one-dimensional lumped film control volume composed of frost and retained water/ice

$$m_{film} = m_f + m_{wi}. \tag{1}$$

The total enthalpy of the film is defined as

$$H_{film} = m_f h_f + m_{wi} h_{wi} \tag{2}$$

which is the sum of frost enthalpy and retained water/ice enthalpy. Carrying out an energy conservation analysis of the film volume yields

$$\frac{dH_{film}}{dt} = \dot{H}_{in} - \dot{H}_{out} + \dot{Q} \tag{3}$$

where the inlet enthalpy flow $\dot{H}_{in}$ corresponds to water vapor desublimation during frost formation, the exit enthalpy flow $\dot{H}_{out}$ corresponds to melted water drainage and vaporization during frost melting, $\dot{Q}$ represents heat transfer rate (convection or conduction) with ambient air and coil metal walls. It is important to note that the enthalpy flow source terms are only active corresponding to a specific operating mode (e.g, frost formation, frost melting), and the heat transfer rate changes signs when the heat flow directions reverse under mode switches.

Assume that the mediums of the film volume are in thermal equilibrium and their temperature gradients are negligible. As a result, a lumped temperature can be determined from the enthalpy. Note that frost is a mixture of dry air and ice. Equation 2 can be rewritten as

$$H_{film} = m_{ice} h_{ice} + m_{air} h_{air} + m_{wi} h_{wi} \tag{4}$$

where m_{ice} and m_{air} denote mass of ice and air of the frost layer, respectively. Apparently, the determination of the film temperature depends on operating modes. In case of frost formation, the retained water on the coil surface must be frozen before frost deposition happens, since a necessary condition of frost formation is that the surface temperature is below the water freezing point T_0 (e.g., 273.15 K). This physical insight indicates that the film undergoes frost formation process when the condition $H_{film} < H_{film,s0}$ holds where $H_{film,s0}$ represents the enthalpy when all retained water turns into solid ice at T_0 and is computed by

$$H_{film,s0} = m_{ice} h_{ice,0} + m_{air} h_{air,0} + m_{wi} h_{ice,0} \tag{5}$$

where $h_{ice,0}$ and $h_{air,0}$ represent specific enthalpies at the freezing temperature. During frost formation, the film enthalpy is proportional to temperature changes with a lumped specific heat capacity since there is no internal phase change inside the volume. As a result, the film temperature is obtained by

$$T_{film} = \frac{H_{film} - H_{film,s0}}{m_{film} c_{p,film}} + T_0 \tag{6}$$

DOI 10.3384/ecp218335

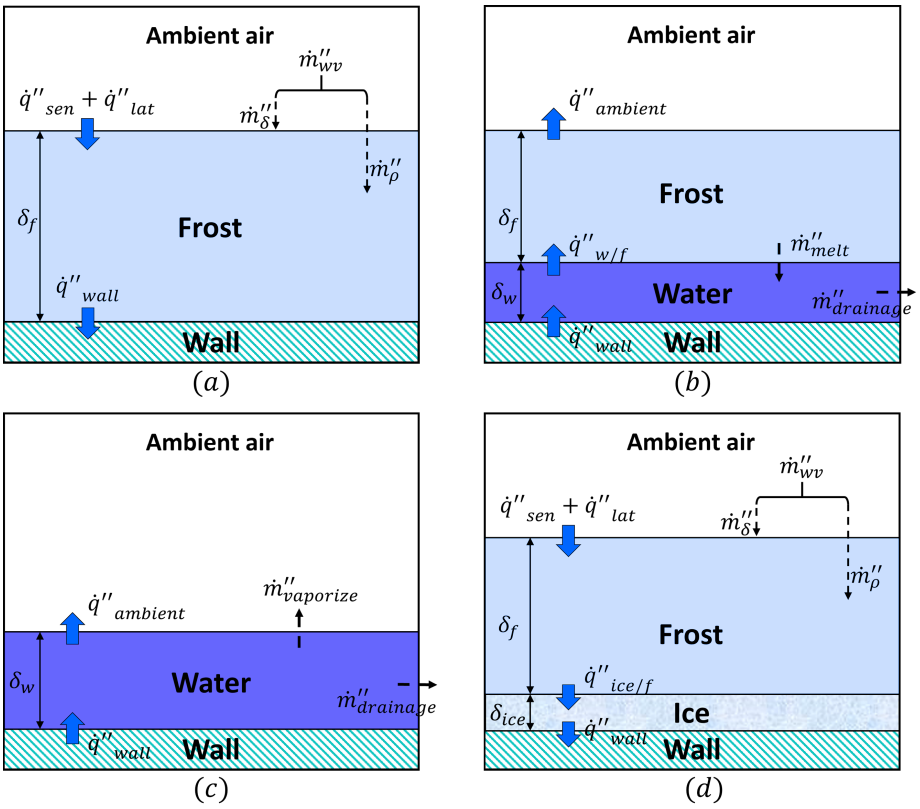


Figure 1. Frost and water/ice processes during cyclic frosting and defrosting: (a) Process 1: frost formation on dry coils; (b) Process 2: frost melting and water drainage; (c) Process 3: retained water vaporization and drainage; (d) Process 4: retained water refreezing and frost formation on ice.

where $c_{p,film}$ is a mass-weighted property and is determined by

$$c_{p,film} = \frac{c_{p,ice}m_{ice} + c_{p,air}m_{air} + c_{p,wi}m_{wi}}{m_{film}}. \tag{7}$$

Similarly, an enthalpy condition $H_{film} > H_{film,l0}$ can be employed to indicate the conclusion of the frost melting process as well as the switch to retained water drainage and vaporization, as shown in Figure 1 (c) where

$$H_{film,l0} = m_{ice}h_{w,0} + m_{air}h_{air,0} + m_{wi}h_{w,0} \tag{8}$$

defines a film enthalpy at T_0 when solely liquid water is present in the film volume. In this case, further heating of the film leads to a temperature above the freezing point, which can be determined using Equation 6 with $H_{film,l0}$ as a reference.

Clearly, the film undergoes phase changes associated with either the frost layer or the retained water/ice when $H_{film,s0} < H_{film} < H_{film,l0}$ is met, with a temperature at T_0. These processes include frost melting, and retained water refreezing or melting.

In summary, the film temperature can be retrieved from its enthalpy and mass composition at any time instance as

$$T_{film} = \begin{cases} \frac{H_{film} - H_{film,s0}}{m_{film}c_{p,film}} + T_0 & \text{if } H_{film} < H_{film,s0}, \\ \frac{H_{film} - H_{film,l0}}{m_{film}c_{p,film}} + T_0 & \text{if } H_{film} > H_{film,l0}, \\ T_0 & \text{otherwise.} \end{cases} \tag{9}$$

In terms of mass conservation, analyses are carried out for the frost layer and the retained water/ice layer separately. The frost mass is characterized by its thickness and density $m_f = \delta_f \rho_f A_s$ as mass transfer contributes to its growth and densification simultaneously. The mass conservation can then be formed as

$$\frac{d\delta_f}{dt} = \frac{\dot{m}_{\delta,in} - \dot{m}_{\delta,out}}{\rho_f A_s}, \tag{10}$$

$$\frac{d\rho_f}{dt} = \frac{\dot{m}_{\rho,in} - \dot{m}_{\rho,out}}{\delta_f A_s} \tag{11}$$

with a decomposition of the source mass flow rates into portions that correspond to mass transfer at the frost surface and within the frost layer. Finally, a mass balance of the retained water/ice layer is formed as

$$\frac{dm_{wi}}{dt} = \dot{m}_{wi,in} - \dot{m}_{wi,out}. \tag{12}$$

With the model setup presented above, transient behaviors of the lumped film can be described by its enthalpy, the frost layer thickness and density, and the retained water/ice mass. Other properties of the film are retrieved from these state variables. For example, an average film density is computed by

$$\rho_{film} = \frac{m_{film}}{\delta_f + m_{wi}/\rho_{wi}} \tag{13}$$

where ρ_{wi} represents water/ice density depending on the operating mode.

2.2 Switched source terms

2.2.1 Heat and mass transfer

Evaluations of the source terms that appear in the governing equations depend on operating modes and phase presences of the film. In the current work, coupled behaviors of the frost layer and water/ice layer are described in the following stages based on phase presences, which cover scenarios associated with cyclic frosting and defrosting operations of heat pumps

- stage 1: frost formation on retained solid ice, corresponding to Process 4 shown in Figure 1 (d);
- stage 2: melting of the retained ice and frost, corresponding to Process 2 shown in Figure 1 (b);
- stage 3: retained liquid water drainage and vaporization, corresponding to Process 3 shown in Figure 1 (c);
- stage 4: dry coil with no frost or retained water/ice.

To derive source terms of heat and mass transfer under frost formation at stage 1, an important assumption is that water vapor is saturated at the frost surface. The water vapor mass flow rate from air streams $\dot{m}_{wv}$, which is obtained by mass transfer calculations (see Eq. (41)), is divided into a portion that diffuses into the frost layer and another that deposits on the frost surface. The mass flow rate of water vapor that diffuses into the frost layer and contributes to its densification over the surface area A_s can be obtained according to Fick's law at the frost surface

$$\dot{m}_{\rho,in} = D_{eff} A_s \left.\frac{\partial \rho_{wv}}{\partial x}\right|_{x=\delta_f} \tag{14}$$

and is approximated in a finite-difference fashion across the film volume

$$\dot{m}_{\rho,in} \approx D_{eff} A_s \frac{\rho_{wv}(T_{fs}) - \rho_{wv}(T_{wall})}{\delta_f} \tag{15}$$

where $\rho_{wv}(T_{fs})$ and $\rho_{wv}(T_{wall})$ denote saturated water vapor density at the film surface and wall temperatures, respectively. The effective diffusion coefficient D_{eff} is evaluated using the following correlation (Breque and Nemer 2016)

$$D_{eff} = \frac{\varepsilon}{1 - 0.58(1-\varepsilon)} \tag{16}$$

where ε represents the frost porosity. Given a total water vapor mass flow rate from air streams $\dot{m}_{wv}$, the portion that deposits on the frost surface can be obtained by

$$\dot{m}_{\delta,in} = \dot{m}_{wv} - \dot{m}_{\rho,in}. \tag{17}$$

Since there is no mass flow leaving the frost layer during this stage, Equation 10 and 11 can be evaluated with $\dot{m}_{\delta,in}$ and $\dot{m}_{\rho,in}$, respectively. Furthermore, the retained water/ice layer yields no mass transfer, which leads to the mass m_{wi} remaining constant.

Heat transfer between the film volume and air streams consists of sensible and latent heat due to water vapor desublimation. Carrying out an energy balance at the film surface leads to

$$\dot{Q}_a = A_s k_{film} \frac{T_{fs} - T_{film}}{\delta_{film}/2} \tag{18}$$

which can be used to obtain the film surface temperature T_{fs}, given that the total heat transfer (include sensible and latent) from air streams $\dot{Q}_a$ (obtained by Eq. (43)) depends on the surface temperature as well. k_{film} is the film thermal conductivity which is determined in a density-weighted manner

$$k_{fim} = k_f + (k_{wi} - k_f) \frac{\rho_{film} - \rho_f}{\rho_{wi} - \rho_f}. \tag{19}$$

A correlation reported in (K.-S. Lee, W.-S. Kim, and T.-H. Lee 1997) is adopted to evaluate the frost thermal conductivity as a function of the frost density. Similarly, heat transfer from wall surfaces is calculated by

$$\dot{Q}_{wall} = A_s k_{film} \frac{T_{wall} - T_{film}}{\delta_{film}/2}. \tag{20}$$

As a result, the energy conservation in Equation 3 becomes

$$\frac{dH_{film}}{dt} = \dot{Q}_a + \dot{Q}_{wall} \tag{21}$$

for this stage. Note that the inlet enthalpy flow carried by water vapor is negligible compared to sensible and latent heat which is determined in an air flow model presented in the next section.

At stage 2, the retained ice is melted, followed by the frost melting process. To calculate the melting rate, a quasi-steady-state energy balance is employed

$$\dot{m}_{melt} = \frac{\dot{Q}_a + \dot{Q}_{wall}}{\Delta h_{sl}} \tag{22}$$

where Δh_{sl} represents the water heat of fusion. Typically, outdoor heat exchanger fans are off under defrost mode when the ambient temperature is below the freezing point and the ambient air cannot be employed to melt frost through convection. Therefore, the only mass transfer source of the frost layer is the melted water $\dot{m}_{\delta,out} = \dot{m}_{melt}$, which flows into the retained water/ice layer and implies $\dot{m}_{wi,in} = \dot{m}_{melt}$. Furthermore, it is assumed that the frost density remains constant during frost melting. The energy conservation shown in Equation 21 also applies to this stage.

DOI 10.3384/ecp218335

After frost is melted and with liquid water retained on coil surfaces, stage 3 focuses on transients of the retained water. Two primary processes are liquid drainage owing to gravity and vaporization as heating coil surfaces leads to mass transfer. It is challenging to quantify the water drainage rate physically since it relies on a number of factors such as surface orientation, wettability, etc. In the current work, a first-order model

$$\dot{m}_{drain} = \frac{m_{wi}}{t_{drain}} \tag{23}$$

is applied to describe the process (Westhaeuser et al. 2023). The drainage flow rate is proportional to the total retained water mass with a time constant t_{drain} which needs to be calibrated for a specific application.

The mass flow rate of vaporized water is determined by

$$\dot{m}_{vap} = c_{vap} A_s \left(\rho_{wv}(T_{film}) - \rho_{wv}(T_a)\right) \tag{24}$$

where c_{vap}[m/s] is a vaporization coefficient (Qiao, Aute, and Radermacher 2018), $\rho_{wv}(T_{film})$ is the saturated water vapor density evaluated at the film temperature, and $\rho_{wv}(T_a)$ is the water vapor density at the ambient temperature. Since both drainage and vaporization move water out of the film control volume, the final form of mass balance for this stage is

$$\frac{dm_{wi}}{dt} = -\dot{m}_{drain} - \dot{m}_{vap}. \tag{25}$$

Correspondingly, enthalpy flows that result from the retained water mass transfer are evaluated in energy conservation. The outlet enthalpy flow is determined by

$$\dot{H}_{out} = \dot{m}_{drain} h_{film} + \dot{m}_{vap} \Delta h_{lv} \tag{26}$$

where h_{film} denotes the film specific enthalpy and Δh_{lv} denotes the water heat of vaporization.

In case the retained water is completely removed from coil surfaces before a switch back to frosting conditions, heat is dissipated to the ambient air. This is indicated by m_{wi} below a small threshold. At stage 4, the film energy balance is describe by Equation 21 without mass transfer.

2.2.2 Switching algorithm

Source terms associated with all stages are computed at each time step. However, only those relevant to the current stage should be applied in evaluating the governing equations. A robust switching algorithm is crucial for realizing the proposed film model. In the current work, the T-S Fuzzy modeling approach (Takagi and Sugeno 1985) is adopted for this purpose.

A set of switching rules is first introduced based on characteristics of all possible stages described in Section 2.2.1. Apparently, the wall temperature constitutes a candidate for differentiating the frost formation process from other melting and dry heating stages. Specifically, stage 1 is active when the wall temperature is below the freezing point temperature T_0. Otherwise, one of the stages during defrosting is active. With frost presence, the film volume undergoes stage 2 until all the frost is melted. Numerically, this can be determined when the frost layer thickness is above a small threshold (for model robustness concerns). Similarly, the retained water presence is indicated by the retained water/ice layer mass, that also determines whether stage 3 or stage 4 is active. In summary, wall temperature T_{wall}, frost layer thickness δ_f, and retained water/ice layer mass m_{wi} are identified as linguistic variables to construct Fuzzy rules. For each variable, two Fuzzy numbers are created with unique membership functions. For instance, Figure 2 depicts membership functions associated with Fuzzy numbers N and P for the linguistic variable T_{wall}. It can be seen that values of both membership functions change around the freezing point temperature T_0 which corresponds to rules that depend on whether T_{wall} is above or below T_0. It is important to note that the piecewise membership functions are continuous and differentiable to avoid generating events during simulations. Fuzzy numbers associated with δ_f and m_{wi} are acquired similar to those shown in Figure 2 where membership functions associated with Fuzzy numbers N and P change values between 0 and 1 around a threshold. In this case, $\delta_{f,\varepsilon} = 0.01$mm and $m_{wi,\varepsilon} = 1e^{-5}$kg are applied as thresholds.

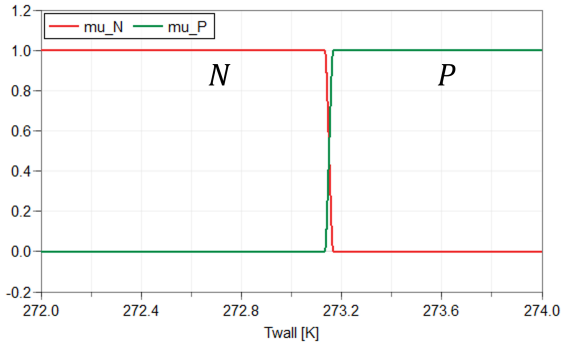


Figure 2. Membership functions associated with Fuzzy numbers for the linguistic variable T_{wall}.

After that, the following set of Fuzzy rules can be formed using Fuzzy numbers

1. stage 1 IF T_{wall} is N;
2. stage 2 IF T_{wall} is P AND δ_f is P;
3. stage 3 IF T_{wall} is P AND δ_f is N AND m_{wi} is P;
4. stage 4 IF T_{wall} is P AND δ_f is N AND m_{wi} is N,

to identify continuous mode switches under cyclic operations. Finally, numerical weights are assigned to source terms associated with each stage to realize mode switches between these stages through a defuzzification process re-

vealed below

$$\omega_1 = \mu_N(T_{wall}), \tag{27}$$
$$\omega_2 = \mu_P(T_{wall}) \times \mu_P(\delta_f), \tag{28}$$
$$\omega_3 = \mu_P(T_{wall}) \times \mu_N(\delta_f) \times \mu_P(m_{wi}), \tag{29}$$
$$\omega_4 = \mu_P(T_{wall}) \times \mu_N(\delta_f) \times \mu_N(m_{wi}). \tag{30}$$

which translates the Fuzzy rules to numerical outputs using membership function values. A weight for each stage is then obtained through normalization

$$\gamma_i = \frac{\omega_i}{\sum_{i=1}^{4} \omega_i}. \tag{31}$$

The final set of governing equations for the film model is presented as

$$\frac{dH_{film}}{dt} = \dot{Q}_a + \dot{Q}_{wall} - \gamma_3(\dot{m}_{drain} h_{film} + \dot{m}_{vap} \Delta h_{lv}) \tag{32}$$
$$\frac{d\delta_f}{dt} = \frac{\gamma_1 \dot{m}_{\delta,in} - \gamma_2 \dot{m}_{melt}}{\rho_f A_s} \tag{33}$$
$$\frac{d\rho_f}{dt} = \frac{\gamma_1 \dot{m}_{\rho,in} + (\gamma_3 + \gamma_4)(\rho_{reset} - \rho_f)/t_{reset}}{\delta_f A_s} \tag{34}$$
$$\frac{dm_{wi}}{dt} = \gamma_2 \dot{m}_{melt} - \gamma_3(\dot{m}_{drain} + \dot{m}_{vap}). \tag{35}$$

Note that in Equation 34 the second source term is a numerical treatment using a first-order filter to reset the frost density to an initial value for the next frosting cycle after all the frost is melted. It is necessary for simulations of cyclic frost behaviors since the assumption of a constant frost density is applied to model the frost melting process (stage 2). Alternatively, this can be realized using the reinit() method of the Modelica language.

3 Reversible heat pump system model

The developed film volume model captures transient behaviors of frost and retained water/ice on coil surfaces. Incorporating the film model into an outdoor coil model for a reversible heat pump enables investigation of system performance under cyclic frosting and defrosting operations. This section describes a discretized outdoor heat exchanger model, and then presents other component models as well as system model integration.

3.1 Flat-tube heat exchanger model

An outdoor flat-tube heat exchanger with 6 refrigerant flow passes arranged in 2 rows is discretized by dividing each refrigerant flow pass into two volumes and dividing the air flow path by tube rows. As a result, the heat exchanger is divided into a total number of 12 control volumes. Figure 3 depicts the heat exchanger configuration and discretization for a tube row.

Within each control volume, the reversible refrigerant dynamics are described by the following mass and energy

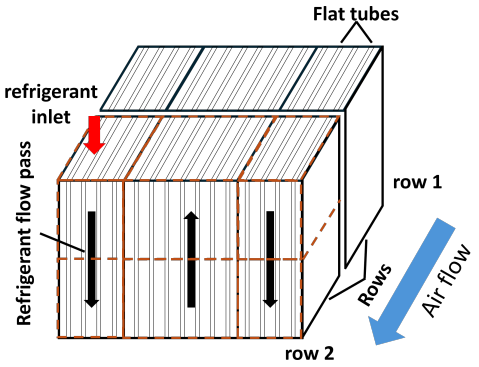


Figure 3. Flat-tube heat exchanger discretization.

balances

$$\frac{d(\rho_r V)}{dt} = \dot{m}_{in} - \dot{m}_{out} \tag{36}$$
$$\frac{d(\rho_r V u_r)}{dt} = \dot{m}_{in} h_{in} - \dot{m}_{out} h_{out} - \dot{Q}_r \tag{37}$$

along with a static momentum balance that correlates the refrigerant pressure drop with wall friction. The heat transfer rate to metal walls is obtained by

$$\dot{Q}_r = \alpha_r A_{wall}(T_r - T_{wall}) \tag{38}$$

where α_r is the heat transfer coefficient evaluated by an empirical correlation. The refrigerant pressure and enthalpy are selected as state variables, and all other thermophysical properties (e.g., T_r, u_r) are retrieved based on them with lookup table interpolation property models. It is important to note that the refrigerant flow descriptions are valid for arbitrary flow directions and handle reverse flow in a numerically sound way. Since the focus of this paper is on frost/defrost model formulations, readers are referred to (Ma 2024) for two-phase flow modeling details.

Metal walls of flat tubes and fins are modeled as lumped capacitance

$$C_{wall} \frac{dT_{wall}}{dt} = \dot{Q}_r - \dot{Q}_{wall} \tag{39}$$

where $\dot{Q}_{wall}$ is the heat transfer to the film and is defined in Equation 20.

The air-side heat and mass transfer over the film are modeled with a quasi-steady-state assumption due to their fast responses compared to the refrigerant flow or frost. A mass balance is formed as

$$\dot{m}_{in} - \dot{m}_{out} - \dot{m}_{cond} = 0 \tag{40}$$

where $\dot{m}_{cond}$ represents the water vapor mass transfer through condensation or desublimation, and is obtained by

$$\dot{m}_{cond} = \beta A_s(\omega_{in} - \omega_{sat}(T_{fs})) \tag{41}$$

DOI
10.3384/ecp218335

where β is a mass transfer coefficient evaluated using the heat and mass transfer analogy with Lewis number $Le = 0.89$, ω_{in} represents the inlet air humidity ratio, and $\omega_{sat}(T_{fs})$ represents the saturated air humidity ratio evaluated at the film surface temperature. The energy balance is formed as

$$\dot{H}_{in} - \dot{H}_{out} - \dot{Q}_a = 0 \tag{42}$$

where $\dot{Q}_a$ represents the total heat transfer rate that consists of sensible and latent parts, and is calculated by

$$\dot{Q}_a = \alpha_a A_s (T_a - T_{fs}) + \dot{m}_{cond} \Delta h_{lat}. \tag{43}$$

Here α_a represents the air-side heat transfer coefficient and Δh_{lat} is the latent of either vaporization or sublimation depending on the surface temperature. Note that the outlet enthalpy flow $\dot{H}_{out}$ includes those of the air stream as well as the condensed water. A lumped air-side pressure loss is calculated based on the outlet air of the heat exchanger instead of air streams in each channel.

3.2 Heat pump system model integration

The developed flat-tube outdoor heat exchanger model incorporated with frost/defrost is subsequently integrated into an automotive air-source heat pump system model. The heat pump can operate in heating or cooling/defrost modes by controlling a reversing valve. R1234yf is utilized as the working fluid. The system adopts a variable-speed compressor and an electronic expansion valve (EXV) as the expansion device. Moreover, an indoor flat-tube heat exchanger that is similar to the outdoor one but with different geometries is employed.

Modeling the indoor heat exchanger follows the same approach for the refrigerant flow, metal walls, and air flows as presented before. The only distinction compared to the outdoor heat exchanger model is the absence of the film model since no frost/defrost phenomenon occurs on the indoor side.

A semi-empirical compressor model is employed to calculate the compressor mass flow rate as well as discharge enthalpy using volumetric and isentropic efficiencies which are correlated as functions of discharge and suction pressures and the compressor speed. Furthermore, the shaft power is determined using the shaft torque and speed.

The EXV model follows a standard throttling process for both flow directions. The mass flow rate across the EXV is computed by

$$\dot{m} = K_v \sqrt{\rho_{in} |p_{in} - p_{out}|} \tag{44}$$

where K_v is a flow coefficient determined by lookup tables when the valve position changes to capture the valve characteristics under varying operating conditions.

The outdoor heat exchanger fan is modeled using performance tables to calculate the nominal fan power as well as volumetric flow rate that the fan delivers based on the pressure rise across the fan, which is equalized to the pressure drop of the heat exchanger. Then the actual performance of the fan is obtained by scaling nominal values with the fan speed.

The suction accumulator captures the refrigerant mass and energy conservations, similar to the heat exchanger model. However, the refrigerant exit quality is determined as a function of the liquid level.

Actuation signals for the heat pump system model include the compressor speed, outdoor fan speed, EXV opening, indoor air flow rate, and the reversing valve opening which controls the refrigerant flow direction and the system operating mode. A schematic of the heat pump system model is shown in Figure 4. Besides the major components described above, the system model also include connecting pipes, sensors, and controls. All component models are implemented using Modelon's Air Conditioning Library (`https://modelon.com/library/air-conditioning-library/`).

4 Simulation results

The developed heat pump system model is simulated to investigate system performance under cyclic frosting and reverse-cycle defrosting operations and to demonstrate capabilities of the developed film model in capturing frost and retained water/ice behaviors. Simulations are carried out using the Modelica tool Dymola 2024x with the Dassl solver and a tolerance of $1e^{-4}$.

The system is initialized at an off condition where the refrigerant pressure is equalized across the system. Vapor refrigerant resides in the indoor heat exchanger while two-phase refrigerant is stored in the outdoor heat exchanger which is in thermal equilibrium with the ambient. Upon a heating cycle startup, the compressor speed, outdoor fan speed, and indoor air mass flow rate are ramped up to their nominal values while the EXV opening is fixed. After the compressor speed settles, superheat control is activated to regulate the EXV opening using a PI controller, as shown in Figure 4. After a period of operation under frosting conditions, the system switches to a defrost cycle to remove the accumulated frost and restore system performance. A number of actuations are considered during mode switching. The EXV is fully opened while the compressor operates at a lower speed than the nominal value during defrost. In the meantime, the reversing valve is energized to redirect the refrigerant flow. On the air side, the outdoor fan is shut off. However, the indoor air mass flow rate remains the same as before in order to heat the refrigerant that flows back to the compressor in the reversed direction. Similarly, when the system switches back to the heating mode, the reversing valve is energized again and other actuators follow the same operations as heating cycles. It should be noted that vehicle driving cycles that have a significant impact on air flows are not considered in the simulated conditions. Consequently, air flow rates through both coils are determined by the given boundary

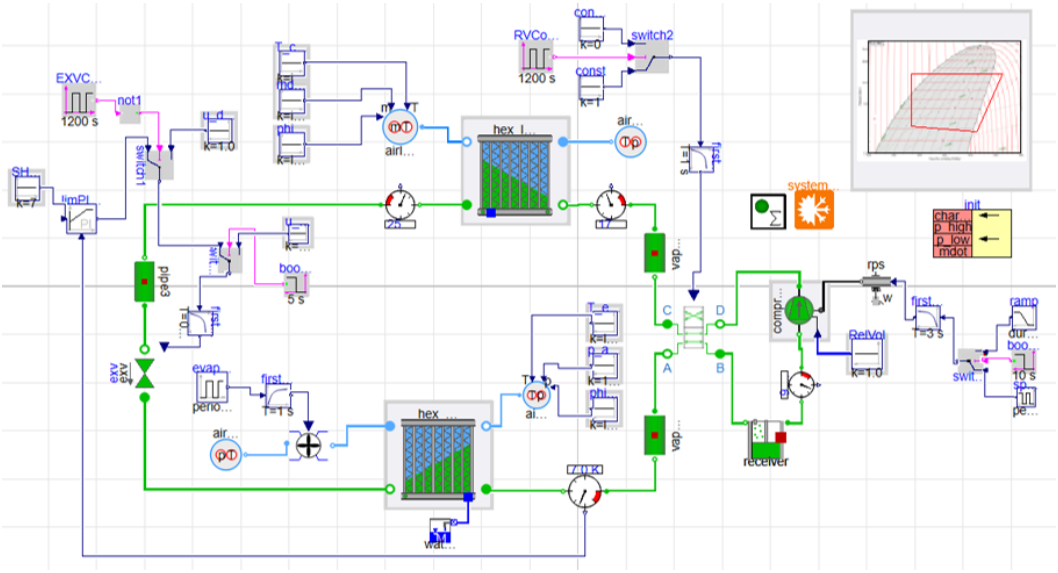


Figure 4. Heat pump system model schematic.

conditions.

In the current work, the indoor air condition is set to 20 C° with a relative humidity of 50%, while the outdoor ambient condition is 2 C° with a relative humidity of 70%. A simulation concerning 25-minute operations is carried out where the system operates in heating mode for 20 minutes before switching to a short defrost cycle, which lasts for 1 minute. After that, the system switches back to the heating model and runs for 4 minutes.

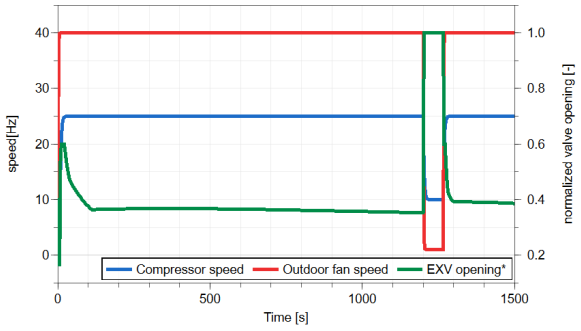


Figure 5. Actuation signals of the compressor and outdoor fan speeds and normalized EXV opening.

Figure 5 reveals the compressor speed, outdoor heat exchanger fan speed, and EXV opening corresponding to the operating characteristics described above. Figure 6 reports discharge and suction pressures. It can be seen that the discharge pressure settles at around 14 bar while the suction pressure at 2 bar after a startup period of 100 seconds. Then a gradual decrease of the discharge pressure can be observed during the heating cycle. This is due to frost formation on the outdoor heat exchanger that deteriorates heat transfer and leads to a slight decrease of the evaporating temperature. A dramatic drop of the discharge pressure at 1200 seconds is attributed to the actuation of the reversing valve and opening of the EXV. Ater the refrigerant flow is reversed, pressure difference across the compressor starts building up again similar to a startup operation until the reversing valve is energized to conclude the defrost cycle, which is followed by another heating cycle with notable performance degradation due to frost accumulation.

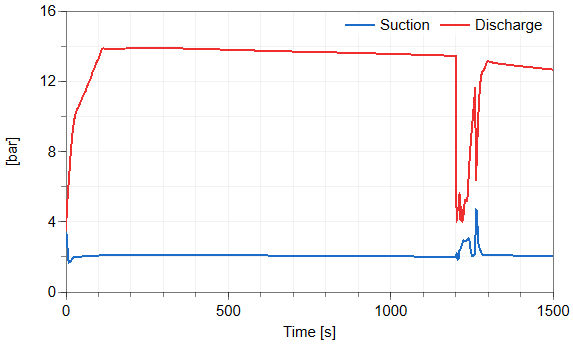


Figure 6. Simulation results of discharge and suction pressures.

The same trend can be observed in the indoor heating capacity and the compressor power, as shown in Figure 7. Clearly, the frost buildup leads to a reduction of the air-side capacity during the heating cycle. However, as a result the compressor power also decreases due to a smaller pressure ratio. During the defrost cycle, assuming that no additional heat source (e.g., electric heater) is available, the indoor capacity changes sign since the indoor heat exchanger operates as an evaporator, which indicates that the heat pump system absorbs heat from the indoor environment and can cause thermal discomfort.

Figure 8 examines evolution of the frost mass and the retained water/ice mass throughout cyclic operations, as a major contribution of the present work is to derive frost models with retained water/ice behaviors taken into con-

DOI
10.3384/ecp218335

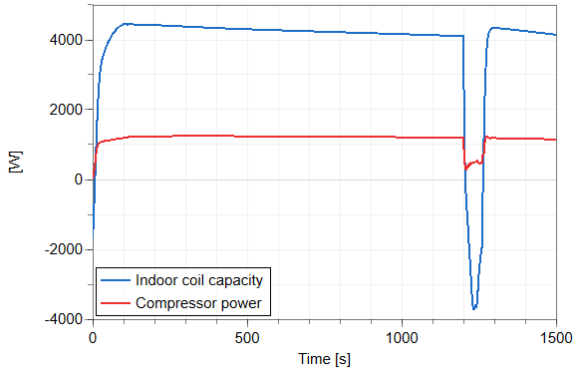


Figure 7. Simulation results of indoor air-side capacity and compressor power.

sideration. The film mass as the sum of them is also shown. The frost mass keeps increasing upon system startup, while no mass transfer associated with the retained water/ice occurs during this period. The accumulated frost is melted during the defrost cycle, which leads to an increased retained water mass. Meanwhile, the retained water drains away and vaporizes as coil wall surfaces are heated. However, the system switches back to the heating mode before a complete removal of the retained water. Consequently, the retained water refreezes on coil surfaces as frost starts buildup in the next heating cycle. This additional thermal resistance added on the heat exchanger surface and blockage of free air flow passages can accelerate system performance deterioration in the next heating cycle. These results demonstrate that the proposed model is capable of capturing the retained water refreezing phenomenon under cyclic operations.

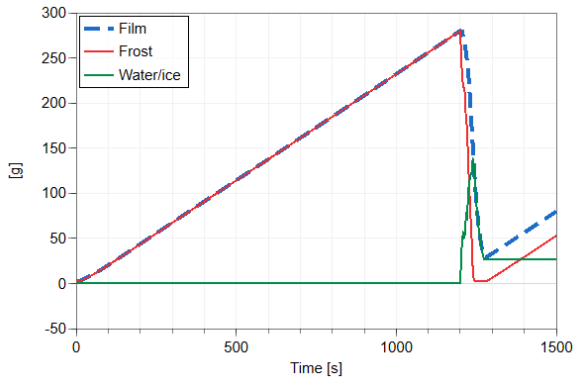


Figure 8. Simulation results of frost and retained water/ice mass.

Furthermore, Figure 9 reveals non-uniform film thicknesses across the outdoor heat exchanger. Film thicknesses of control volumes at the refrigerant flow inlet (the first control volume of tube row 2), in the middle of the refrigerant path (the first control volume of tube row 1), and at the refrigerant flow outlet (the last control volume of tube row 1) are shown. It is important to note that the refrigerant enters the heat exchanger at tube row 2, as tube row 1 is facing inlet air flows (see Figure 3). It can be seen that more frost accumulates in the control volume of tube row 1 than tube row 2. This is because the moist air first flows through tube row 1 where moisture content is greater than that of tube row 2 since a portion of water vapor carried in the air deposits on tube row 1. In addition, no frost behavior is observed in the control volume at the heat exchanger outlet due to the fact that the superheated refrigerant brings the coil surface temperature above the local dew point temperature of the moist air that results in dry coil operations there.

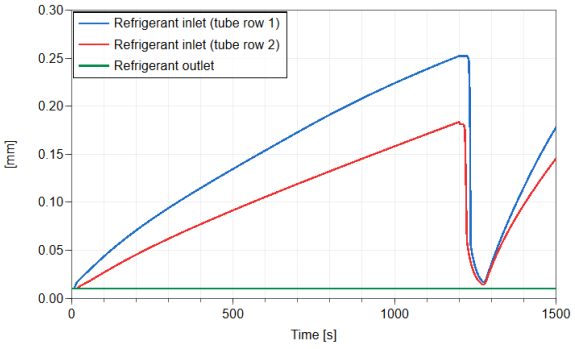


Figure 9. Simulation results of discretized film thicknesses of selected control volumes.

5 Conclusions

This paper presents a novel film volume model that integrates frost and retained water/ice behaviors using an enthalpy method. The film model captures possible operating modes that involve frost formation and melting, and retained water/ice refreezing and melting under cyclic frosting and defrosting operations. The advantage of using the enthalpy method to model these phase change behaviors is that it features a uniform model structure with switched source terms for various operating modes. A switching algorithm based on the Fuzzy modeling approach is proposed to avoid discontinuities under mode switches. The developed film volume model is incorporated into a discretized outdoor flat-tube heat exchanger model which is then coupled to other component models of an automotive heat pump system. The system model is simulated to investigate system responses under cyclic heating and reverse-cycle defrosting operations from a startup. Simulation results indicate that the developed model is capable of capturing the complicated cyclic transients of the refrigerant, air, frost, and retained water/ice in case of an insufficient defrost cycle. The proposed modeling approach can be extremely useful for improved defrost control designs and evaluations of air-source heat pumps as well as low-temperature refrigeration applications.

Future research efforts can focus on experimentally validating the model in scenarios of retained water/ice presences and utilizing the model to investigate the overall im-

pact of retained water refreezing and melting on system performance over multiple defrost cycles to aid performance improvements. Furthermore, the proposed modeling approach can be useful to investigate control strategies that optimize operating efficiencies while minimizing disruptions to the vehicle interior heating under frost conditions.

Nomenclature

Symbols

α	Heat transfer coefficient $[\mathrm{W}/(\mathrm{m}^2\,\mathrm{K})]$
β	Mass transfer coefficient $[\mathrm{kg}/(\mathrm{m}^2\,\mathrm{s})]$
δ	Thickness [m]
$\dot{H}$	Enthalpy flow rate [W]
$\dot{m}$	Mass transfer rate [kg/s]
$\dot{Q}$	Heat transfer rate [W]
ε	Frost porosity $[-]$
ρ	Density $[\mathrm{kg}/\mathrm{m}^3]$
A	Area $[\mathrm{m}^2]$
C	Heat capacity $[\mathrm{J}/\mathrm{K}]$
H	Enthalpy [J]
h	Specific enthalpy $[\mathrm{J}/\mathrm{kg}]$
M	Mass [kg]
T	Temperature [K]
u	Specific internal energy $[\mathrm{J}/\mathrm{kg}]$
V	Volume $[\mathrm{m}^3]$

subscript

a	Air
f	Frost
in	Inlet
out	Outlet
r	Refrigerant
vap	Vaporized
w	Liquid water
$wall$	Coil metal wall
wi	Retained water/ice
wv	Water vapor

References

Alam, Imran et al. (2024). "Improving Heat Pump System Performance Using a Novel Frost/Defrost Model". In: *20th International Refrigeration and Air Conditioning Conference at Purdue*.

Alebrahim, AM and SA Sherif (2002). "Electric defrosting analysis of a finned-tube evaporator coil using the enthalpy method". In: *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 216.6, pp. 655–673.

Breque, Florent and Maroun Nemer (2016). "Frosting modeling on a cold flat plate: Comparison of the different assumptions and impacts on frost growth predictions". In: *International Journal of Refrigeration* 69, pp. 340–360.

Kim, Donghun, James E Braun, and Sugirdhalakshmi Ramaraj (2018). "Computationally efficient modeling strategy for evaporator performance under frost conditions". In: *International Journal of Refrigeration* 96, pp. 88–99.

Lee, Kwan-Soo, Woo-Seung Kim, and Tae-Hee Lee (1997). "A one-dimensional model for frost formation on a cold flat surface". In: *International Journal of Heat and Mass Transfer* 40.18, pp. 4359–4365.

Ma, Jiacheng (2024). "Development, Calibration and Validation of A Dynamic Modeling Framework for Air-Source Heat Pumps Under Cycling of Frosting and Reverse-Cycle Defrosting". PhD thesis. Purdue University Graduate School.

Ma, Jiacheng et al. (2023). "Development and validation of a dynamic modeling framework for air-source heat pumps under cycling of frosting and reverse-cycle defrosting". In: *Energy* 272, p. 127030.

Qiao, Hongtao, Vikrant Aute, and Reinhard Radermacher (2018). "Modeling of transient characteristics of an air source heat pump with vapor injection during reverse-cycle defrosting". In: *International journal of Refrigeration* 88, pp. 24–34.

Takagi, Tomohiro and Michio Sugeno (1985). "Fuzzy identification of systems and its applications to modeling and control". In: *IEEE transactions on systems, man, and cybernetics* 1, pp. 116–132.

Westhaeuser, Jochen et al. (2023). "Model for cyclic frosting and defrosting of flat tube heat exchangers: Theoretical analysis and experimental validation". In: *Applied Thermal Engineering* 225, p. 120140.

Zhang, Wenying, Wenzhe Li, and Pega Hrnjak (2024). "Performance of an automotive reversible CO2 heat pump system during periodic frosting-defrosting cycles". In: *Applied Thermal Engineering* 236, p. 121892.

DOI 10.3384/ecp218335

A Dynamic Analysis of Refrigerant Mass in Vapor Compression Cycles

Scott A. Bortoff, Vedang M. Deshpande, Christopher R. Laughman and Hongtao Qiao [1]

[1]Mitsubishi Electric Research Laboratories, Cambridge, MA, USA {bortoff, deshpande, laughman, qiao}@merl.com

Abstract

Numerical simulation of a thermofluid vapor compression cycle (VCC) model in Modelica, for example, can exhibit a variation in the total fluid (refrigerant) mass. This paper provides a dynamic analysis of a commonly used VCC model, identifies and analyzes the root cause of this variation, and proposes a number of remedies. The cause lies within the dynamic equations that result from application of the principle of mass conservation. In many common formulations, these equations express the conservation of mass as one or more differential equations that equate the time derivative of mass to zero. The resulting set of n ordinary differential equations (and a number of auxiliary algebraic equations) include the time derivative of a mass constraint function, but not the actual mass constraint function itself. As a result, this modeling formulation has the following properties: (1) equilibrium solutions of the system are neither isolated, nor exponentially stable; (2) a linearization about any equilibrium solution has at least one eigenvalue equal to zero, making an equilibrium solution stable, but not exponentially stable; (3) for a VCC model formulated using two fluid states per control volume, a one-dimensional equilibrium manifold exists containing all of the equilibrium solutions, and is parameterized by the total fluid mass; (4) an $(n-1)$ dimensional, stable, invariant manifold exists transverse to the equilibrium manifold, defined by the mass constraint function, and on which *analytic* solutions to the model evolve and the total fluid mass remains constant; and (5) *numerical* solutions may drift off of this manifold, resulting in an observed drift of fluid mass. These properties have consequences for simulation, control design, numerical model reduction, and state estimation. A number of methods to stabilize the mass constraint are proposed and a number of examples that illustrate the behavior, analysis and remedies are provided.

Keywords: vapor compression cycle, mass conservation, simulation, dynamic analysis

1 Introduction

Vapor compression cycles (VCCs) are the primary means of moving heat in almost all modern air conditioning systems. Physically, VCCs consist of one or more hierarchically organized loops of modular components such as heat exchangers, pipes, valves and compressors. The physics of fluid mechanics, usually including phase change and heat transfer is nonlinear and multi-scale in time and space, resulting in models that consist of large sets of coupled, nonlinear, numerically stiff but sparse differential algebraic equations. As such, the Modelica modeling language has proven to be well suited to this application, with its object oriented structure for organization and tools that support simulation of large sparse DAEs, and has generally found successful application in product development throughout the industry, which is continuously under pressures to innovate.

Often VCC model users need to run simulations over relatively long time horizons. For example, it is important to ascertain the annual performance of a building air conditioner, requiring many simulations over a one year time scale. Standard drive cycles in automotive applications are also relatively long time scale simulations. Several authors have reported that VCC models formulated under standard assumptions (conservation equations) do not conserve refrigerant mass, and that this has adverse practical consequences, especially for long time scale simulations. The refrigerant mass, or charge, seems to drift over the course of a simulation. Some choices of state can help alleviate the problem (Laughman and Qiao, 2015, 2016), although not entirely correct it. Practical methods of resetting states (Anderson et al., 2023) can help to reduce the "refrigerant mass drift," but require model modification and possibly manual intervention.

These results address the symptom of refrigerant mass drift in numerical simulations, but not its root cause. In this paper, we analyze a common model formulation of a VCC, identify mathematically the root cause of refrigerant mass drift, and describe two methods that can remedy the situation. The dynamic analysis results in an improved understanding of the structure and behavior of the resulting system of DAEs, with some perhaps surprising consequences and implications. For example, we show that equilibrium solutions to the systems of DAEs are stable, but not exponentially stable, which is often assumed to the the case. In fact, there is an equilibrium manifold of stable solutions to the set of DAEs. The equilibrium manifold is parameterized by the total refrigerant mass, and unavoidable numerical errors incurred during a numerical simulation will result in mass drift along the manifold.

This has implications for simulation, model-based control system design, and state estimation.

Fortunately, the structure suggests several remedies, which may be implemented as modifications to standard modeling constructs and result in exponentially stable models that do not exhibit refrigerant mass variation. This paper is organized as follows. We begin Section 2 with a VCC dynamic model resulting from application of standard mass, energy and momentum conservation assumptions. We analyze this model geometrically, proving existence of an integral submanifold of dimension one less than the state dimension, and existence of a one-dimensional equilibrium manifold. We show how this structure implies drift specifically in the refrigerant mass in a numerical simulation, illustrated by a simple VCC example. We then present two remedies in Section 3, with simulation examples. In Section 4 we also show how this behavior can manifest in a mechanical system model, which aids in understanding the situation, and shows that the results are general in nature and not restricted to only thermofluid cycles. We close in Section 5 by summarizing the results and stating a conjecture regarding more complex VCCs.

2 VCC Dynamics

A dynamic model of the thermofluid physics of a VCC generally begins from mathematical statements of distributed, one-dimensional mass, energy and momentum balance, such as

$$\frac{\partial(\rho A)}{\partial t}+\frac{\partial(\rho A v)}{\partial x}=0 \tag{1a}$$

$$\frac{\partial(\rho v A)}{\partial t}+\frac{\partial(\rho v^2 A)}{\partial x}=-A\frac{\partial P}{\partial x}-F_f \tag{1b}$$

$$\frac{\partial(\rho u A)}{\partial t}+\frac{\partial(\rho v h A)}{\partial x}=vA\frac{\partial P}{\partial x}+vF_f+\frac{\partial Q}{\partial x}, \tag{1c}$$

where P is a pressure, v is a velocity, u is a specific internal energy, ρ is a density, Q is a heat transfer rate, A is a cross-sectional area, V is a volume, F_f is a frictional pressure drop, x is a scalar spatial coordinate and t is time. These may be discretized in the spatial coordinate x resulting in a set of ordinary differential equations (ODEs)

$$\frac{d(\rho_j V_j)}{dt}=\dot{m}_k-\dot{m}_{k+1} \tag{2a}$$

$$\frac{d(\dot{m}_i l)}{dt}=\rho_j v_j^2 A_j-\rho_{j+1}v_{j+1}^2A_{j+1}+\frac{A_j+A_{j+1}}{2}\left(P_{j+1}-P_j\right)+F_{f,i} \tag{2b}$$

$$\frac{d(\rho_j u_j A_j)}{dt}=\dot{H}_k-\dot{H}_{k+1}+v_j jA_j(P_{j+1}-P_j)+vF_{f,i}+\dot{Q}_j, \tag{2c}$$

where the indices j and k correspond to gridding methods, and $\dot{m}_k$ denotes the mass flow rate at the boundaries of a finite control volume. See (Pat, 1980; Tummescheit, 2002; Laughman and Qiao, 2015, 2016) for details of modeling procedures and Modelica realizations.

State selection is dependent on the particulars of an application, but generally two states are selected for each control volume: A mixture variable, and another variable. Herein we choose pressure P and specific enthalpy h, which are common choices for VCC applications. However, the main results are not dependent on this specific choice. For compactness, define the full state $x=[P_1,h_1,P_2,h_2,\ldots,P_m,h_m]\in\mathbb{R}^n$ where m is the number of control volumes and $n=2m$, and express (2) as

$$\dot{x}=f(x,u), \tag{3}$$

where $u\in\mathbb{R}^p$ is a vector of control inputs, such as a compressor speed or an expansion valve opening. For simplicity, this formulation ignores states resulting from consideration of heat transfer to metal. These would have no bearing on the main results. Note that a Modelica model may not explicitly compute f, but rather it may express the dynamics in an implicit form. However, f exists mathematically and will make use of this fact.

In (2) the total refrigerant mass M is an explicit function of state:

$$M=g(x):=\sum_{i=1}^{m}=\rho_i(x_{2i},x_{2i+1})V_i. \tag{4}$$

Note that neither g nor M appear in (2). However, (2a) is a statement that $\dot{g}=0$. In typical use, g is used to set the initial condition $x(t_0)=x_0$ to correspond to a total refrigerant mass M_0 using

$$M_0=g(x_0). \tag{5}$$

2.1 Main Results

Since $\dot{g}=0$, we have

$$\dot{g}=\frac{\partial g}{\partial x}\cdot\dot{x}=\frac{\partial g}{\partial x}\cdot f=0. \tag{6}$$

Therefore, at any equilibrium solution x_e satisfying $f(x_e,u_e)=0$, we have

$$\frac{\partial g}{\partial x}|_{\substack{x=x_e\\u=u_e}}\cdot f(x_e,u_e)=0. \tag{7}$$

Expanding f into a Taylor's series about (x_e,u_e), we have

$$\frac{\partial g}{\partial x}|_{\substack{x=x_e\\u=u_e}}\cdot A=0, \tag{8}$$

where A is the Jacobian of f with respect to x, i.e., $A=\frac{\partial f}{\partial x}|_{\substack{x=x_e\\u=u_e}}$. Therefore

$$U_0:=\frac{\partial g}{\partial x}|_{\substack{x=x_e\\u=u_e}} \tag{9}$$

is the left eigenvector corresponding to a zero eigenvalue of A. Define V_0 to be the right eigenvector of A corresponding to the zero eigenvalue of A. Since A has one

DOI 10.3384/ecp218345

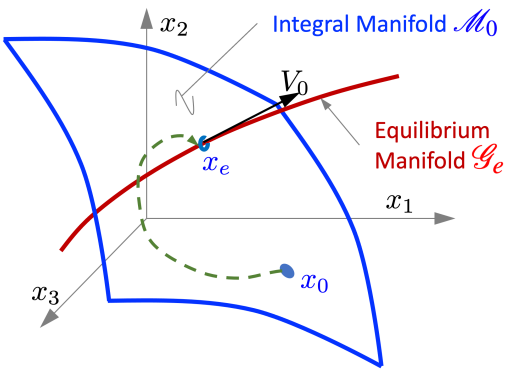


Figure 1. Integral manifold $\mathscr{M}_0$, equilibrium manifold $\mathscr{G}_e$, and a solution trajectory (green) for initial condition $x_0 \in \mathscr{M}_0$, converging to the stable equilibrium $x_e \in \mathscr{M}_0 \cap \mathscr{G}_e$.

zero eigenvalue (and the remaining $n-1$ eigenvalues are assumed to lie in the open left half plane), the equilibrium solution x_e is stable, but not exponentially stable. Further, x_e is not isolated, since $\tilde{x}_e = x_e \pm \varepsilon \cdot V_0$ is also an equilibrium for any $\varepsilon > 0$, by definition of the eigenvalue and eigenvector. Therefore, the linearized system

$$\delta\dot{x} = A\delta x + B\delta u$$

has an equilibrium subspace defined by the span of V_0, and therefore the nonlinear system (2) has a one-dimensional equilibrium manifold $\mathscr{G}_e$, which is tangent to V_0 at x_e, defined by those states x_e satisfying

$$0 = f(x_e, u_e) \tag{10a}$$
$$0 = g(x_e) - M_e \tag{10b}$$

which is parameterized by the refrigerant mass M_e (and also u_e). Finally, as as consequence, the nonlinear system has an $n-1$ dimensional integral manifold $\mathscr{M}_0$, implied by the fact that f lies within the tangent space of the $n-1$ dimensional submanifold defined by $dg = 0$, where d is the gradient operator. (An *integral manifold* $\mathscr{M}_0$ is invariant to the dynamics, meaning $x_0 \in \mathscr{M}_0 \rightarrow x(t) \in \mathscr{M}_0$ for all $t \geq t_0$.)

The situation is diagrammed in Figure 1, where an initial condition x_0 is defined to satisfy (10), the analytic solution $x(t)$ evolves on the $n-1$ dimensional integral manifold $\mathscr{M}_0$, which contains x_0, converging to the equilibrium solution x_e which lies in the intersection of $\mathscr{M}_0$ and $\mathscr{G}_e$. The equilibrium x_e is not exponentially stable because the eigenvector V_0 corresponding to the the zero eigenvalue is tangent to $\mathscr{G}_e$. As such, there is a family of $n-1$ dimensional integral manifolds $\mathscr{M}_k$, all transverse to $\mathscr{G}_e$, defined by the initial condition x_0. (These foliations are not shown in Figure 1 for clarity.) All of these results follow from the fact that A has one eigenvalue equal to zero, basic linear algebra and elementary nonlinear differential equation theory found in reference texts such as (Vidyasagar, 1993), (Khalil, 2002) and (Isidori, 1989).

2.2 Some Implications

This geometric structure has implications for numerical simulation, control design, model reduction and state estimation applications. Analytically, a solution $x(t)$ evolves on the $n-1$ dimensional integral manifold $\mathscr{M}_0$, and the effective dimension of the state vector is $n-1$, given an initial condition x_0, which determines the refrigerant mass M_0. However, a numerical simulation evolves in the full n-dimensional state space. Inevitable numerical errors at each discrete time step of any solver algorithm will result in an approximate solution that is not invariant to $\mathscr{M}_0$, i.e., the solution will drift off $\mathscr{M}_0$, onto some other $\mathscr{M}_k$. This drift is in exactly the direction of varying the refrigerant mass. Further, since the dynamics in the $\mathscr{G}_e$ direction are not exponentially stable, these errors are not corrected step to step. In effect, the drift is like a one dimensional random walk. The constraint $0 = g(x) - M_0$ is not stabilized.

In fact, there is a hidden state in the dynamics, corresponding to the direction of dg and locally associated to the eigenvector V_0. In a spectrally decomposed coordinate system this state is an open loop integrator driven by "noise" generated by numerical simulation errors, step to step. (The random walk analogy is very close, except the VCC simulation is deterministic.) For some simulations, this "noise" may be biased, which may be caused by simulation errors near the saturation curve, where the change in density is extreme, and in this case, a numerical simulation may seem to leak its entire charge of refrigerant.

It has been stated by some authors that the "mass is not a state" in such models, but this is not entirely correct. It is not an explicit element of x. However, the refrigerant mass is a state of the system, appearing as an open loop integrator in the spectrally decomposed coordinates. Since the x coordinates are used for simulation, the mass state is hidden, mixed among all the other states. This implies that any time domain simulation will exhibit drift in the refrigerant mass, regardless of the coordinates used, and regardless of the integration algorithm used. Some choice of coordinates (Laughman and Qiao, 2015, 2016) and also measures taken to enforce the mass constraint (Anderson et al., 2023) may result in a reduced magnitude of drift, owing to improved error control, but do not cause the mass constraint to be enforced.

For control, it is common to make use of model linearizations. Care must be taken in reducing the order of such models. Hankel norm methods which remove modes that are both weakly controllable and weakly observable, work only for exponentially stable systems. Matlab provides convenient tools for such purpose, but these will first extract the unstable (and weakly stable) part of the system by spectral decomposition, reduce the stable part, and then put the unstable part back, assuming that is what the user wants! Of course, for these applications, the user needs to remove the mode corresponding to the zero eigenvector first, because this mode is a modeling artifact. The true state dimension is $n-1 = 2m-1$.

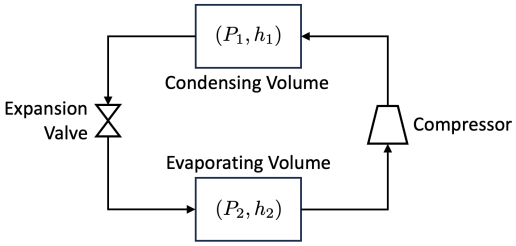


Figure 2. Two-volume vapor compression cycle.

2.3 An Example

Consider the simple two-volume VCC diagrammed in Figure 2, consisting of two control volumes labeled "condensing" and "evaporating" each with pressure-enthalpy states (P_1, h_1) and (P_2, h_2), respectively, a compressor and an expansion valve. The latter are modeled as algebraic functions relating the inlet and outlet states. Here the control input $u = [u_1, u_2]$ with the compressor speed u_1 and expansion valve count u_2. The state vector is $x = [P_1, h_1, P_2, h_2]$, and the dynamics are

$$\dot{x} = f(x,u), \quad x(t_0) = x_0 \tag{11a}$$
$$M = g(x) = \rho_1(P_1,h_1)V_1 + \rho_2(P_2,h_2)V_2\,. \tag{11b}$$

Linearizing the dynamics about an equilibrium solution confirms that 3 eigenvalues are in the open left half plane, and one is at the origin, with its left eigenvector U_0 aligned with the gradient of g. A simulation result with initial condition $P_1 = 2.6\,\text{MPa}$, $h_1 = 420\,\text{kJ/kg}$, $P_2 = 1\,\text{MPa}$ and $h_2 = 230\,\text{kJ/kg}$, constant values of the compressor speed (30Hz) and expansion valve (100 counts), and constant boundary conditions results in the transient response plotted in Figure 3. The refrigerant mass drifts for the 15s, coming to rest after the transient has transpired. Depending on the application, this amount of drift may or may not be acceptable. Importantly, it is entirely driven by numerical solver error, more or less the result of integrating a random signal.

2.4 VCC Models in (P,h,ρ) Coordinates

Formulating the VCC dynamics in (P,h,ρ) coordinates has been suggested as an approach to mitigate mass drift (Laughman and Qiao, 2015, 2016). These coordinates were derived by first expressing the dynamics in (P,ρ)-coordinates, and then adding a dynamic equation for h. Following this approach, assume the change of coordinates from (P,h) to (P,ρ) is well-defined in a region of interest. In the (P,ρ)-coordinates, denote the state vector as $y = [P_1,\ldots,P_m,\rho_1,\ldots,\rho_m]^T$. All the results from Section 2.1 also carry over to these coordinates. Next consider a linearized model around any equilibrium solution $\bar{y} \in \mathbb{R}^{2m}$, ignoring for simplicity the effect of any inputs,

$$\delta\dot{y} = A\delta y + B\delta z \tag{12a}$$
$$\delta z = C\delta y, \tag{12b}$$

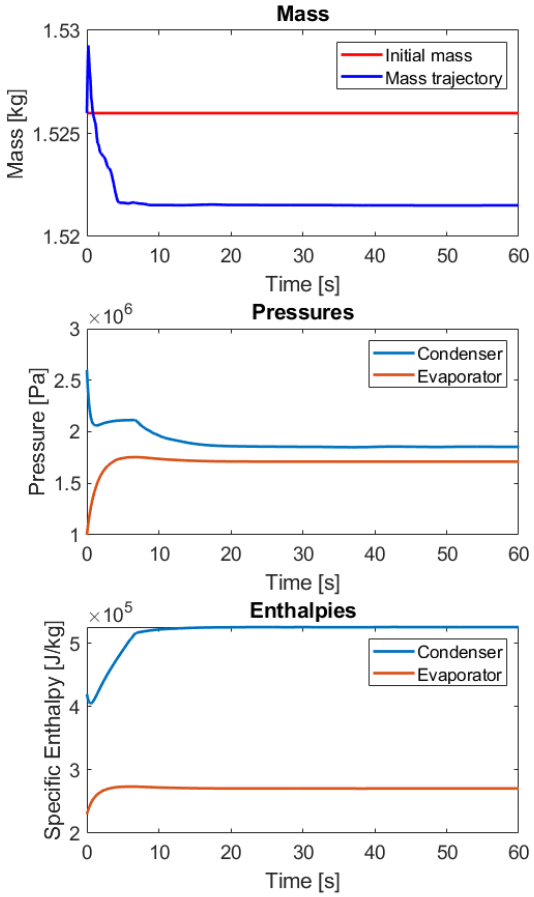


Figure 3. Refrigerant mass drift during initial transient.

where $\delta y = y - \bar{y}$, and $z \in \mathbb{R}^m$ is the enthalpy vector h, $\bar{z} \in \mathbb{R}^m$ is the equilibrium enthalpy vector, and $\delta z = z - \bar{z}$. The linearized enthalpy term δz is explicitly pulled out and expressed as an output function because (P,h,ρ)-coordinates use it as a third state in each control volume. Note that the linearized dynamics in (P,ρ) coordinates are $\delta\dot{y} = (A + BC)\delta y$.

In this notation, the corresponding linearization in (P,h,ρ) coordinates is

$$\delta\dot{y} = A\delta y + B\delta z \tag{13a}$$
$$\delta\dot{z} = C\delta\dot{y} = CA\delta y + CB\delta z. \tag{13b}$$

Note that the δz in (13b) is used on the right hand side of (13a), which means that in the original (P,h,ρ)-coordinate formulation, the h states, computed as solutions to m ODEs, are used in the $2m$ ODEs defining $\dot{P}$ and $\dot{\rho}$. Rewriting (13) in matrix form

$$\begin{bmatrix} \delta\dot{y} \\ \delta\dot{z} \end{bmatrix} = \begin{bmatrix} A & B \\ CA & CB \end{bmatrix} \begin{bmatrix} \delta y \\ \delta z \end{bmatrix}, \tag{14}$$

makes it clear that the system has m eigenvalues at zero, since the last m rows are linearly dependent on the first $2m$. As such a system expressed in (P,h,ρ)-coordinates will have a linearization with $m+1$ eigenvalues at the origin,

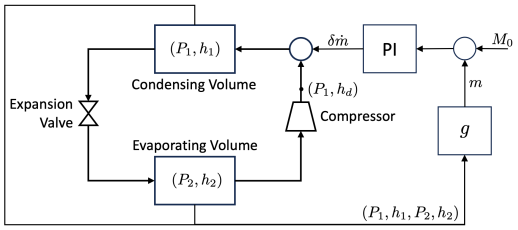


Figure 4. Two-volume vapor compression cycle.

and the nonlinear system will have an equilibrium manifold of dimension $m+1$ parameterized by the mass, and also the error between the m values of enthalpy h computed by the additional m differential equations and the algebraic constraint $h = h(P,\rho)$ that was differentiated. The improved accuracy in mass conservation reported in (Laughman and Qiao, 2015, 2016) is likely due to better solver error control in the density dynamics, but drift in a higher-dimensional manifold resulting in non-physical results is possible in long duration simulations.

3 Drifting Mass Remedies

We present two remedies. First, we can exponentially stabilize the total amount of mass in a cycle using a virtual feedback loop with integral action. Consider again the simple two-volume VCC diagrammed in Figure 2, now with a virtual mass feedback loop which acts to regulate the total amount of mass to the constant M_0, shown in Figure 4. This method requires minimal modification of the original model to allow for an additional virtual control input. The virtual input might be an added mass flow rate signal at the discharge port of the compressor, as shown in Figure 4, or alternatively it might be an added enthalpy signal at the discharge port of the compressor.

The PI feedback exponentially stabilizes the total mass, which can be understood by the Root Locus diagrammed in Figure 5. The "plant" here is the system from the virtual input to the measured mass output, which is $m = g(x)$. The plant includes a single pole at the origin, as discussed in Section 2, and $n-1$ poles in the open left half plane, where $n = 4$ for this example. The feedback loop including the PI compensator adds a second pole at the origin, and a zero in the open left half plane, at $s = -0.2$ for this example. Closing the feedback loop with positive gain causes the double poles at the origin to break away and move into the open left half plane, exponentially stabilizing the equilibrium. The action of feedback may change the dynamics of (3) but only slightly, because the dynamics on the integral manifold is invariant and the dynamics on the integral manifold are relatively insensitive to the virtual mass input. This is clear on Figure 5, where the 3 stable poles move very slightly from open to closed loop (the closed loop location is plotted over the open loop in the figure). The integral manifold $\mathscr{M}_0$ is not changed by the PI feedback.

Figure 6 shows the result of a simulation in which the

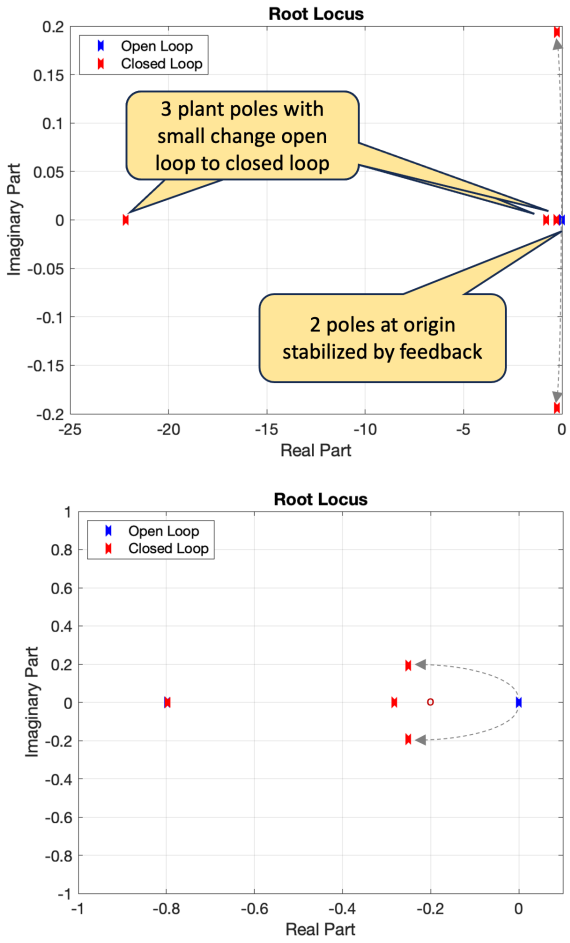


Figure 5. Root locus for the 2-volume system with virtual mass PI feedback (top), magnified to show poles and zeros near the origin (bottom). The PI zero is placed at $s = -0.2$. The 3 stable plant poles are relatively insensitive to the feedback, without visible displacement in the figure. The two unstable poles at the origin are moved to complex conjugate locations by the action of the virtual feedback loop.

virtual mass feedback is active, for the same initial conditions as in Figure 3. The feedback actively stabilizes the total mass and ensures that the equilibrium solution x_e is isolated and exponentially stable.

3.1 Stabilizing the constraint

Alternatively, following Gear (Gear, 1988; Brenan et al., 1996), a second method to stabilize the mass constraint is as follows. The modeled system (3) does not include the refrigerant mass constraint $0 = g(x) - M_0$. But Gear shows that the augmented system

$$\dot{x} = f(x,u) + \mu \cdot dg(x) \tag{15a}$$
$$0 = g(x) - M_0, \tag{15b}$$

where

$$dg = \frac{\partial g}{\partial x} \tag{16}$$

and μ is scalar Lagrange multiplier state, enforces the mass constraint explicitly and has an identical analytic so-

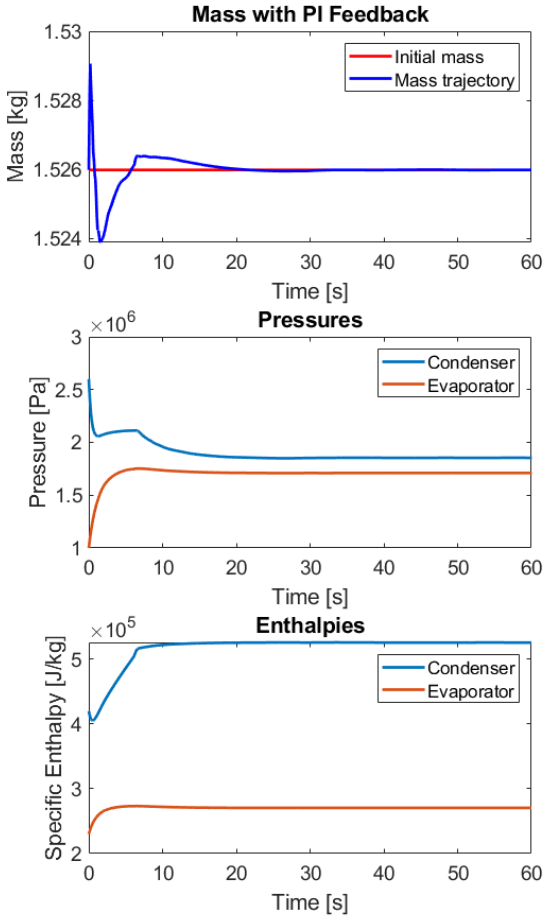


Figure 6. Refrigerant mass under the action of virtual PI feedback control for same initial condition as Figure 3.

lution as (3) with $\mu = 0$. Numerical solutions of (15) will therefore not suffer from drift from $\mathscr{M}_0$. However, the index of (15) is 2, so this system cannot be simulated directly using standard solvers.

The index can be reduced using the available methods provided in Modelica tools such as Dymola, but this could result in a complex and highly coupled model equations, since the gradient term dg contains all of the states and as a practical matter, most VCC models (3) are large in dimension to begin with. Alternatively, we may reduce the index manually using Baumgarte's method (Baumgarte, 1983). This involves replacing the constraint equation (15b) with a linear combination of g and $\dot{g}$, so that the new constraint equation is Hurwitz (roots in the open left half plane),

$$\dot{x} = f(x,u) + \mu \cdot dg(x) \tag{17a}$$
$$0 = \dot{g}(x) + \alpha_0 \cdot (g(x) - M_0), \tag{17b}$$

where the constant $\alpha_0 > 0$. System (17) is index 1 and may be simulated directly using standard solver algorithms. Note that $\dot{g}$ in (17b) is $dg \cdot (f + \mu \cdot dg)$ which not the same as $\dot{g}$ in Section 2, because the definitions of x and $\dot{x}$, respectively, have changed by the addition of the $\mu \cdot dg$ term in (17a).

To implement this scheme, the original model (3) needs modification by adding a single equation (17b) and adding

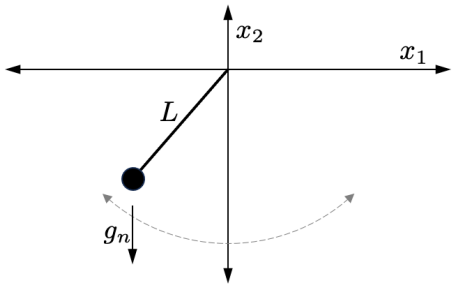


Figure 7. Cartesian pendulum.

the output injection terms $\mu \cdot dg$ to each of the state equations. This would require modification to the base classes of the modeling package, since they were probably not intended for this purpose, but the changes are relatively minor. System (17) may be less computationally complex compared to that obtained by reducing the index of (15) from 2 to 1 using the standard method of substitution of variables (or "dummy derivatives"). The method increases the dimension of the set of equations from n to $n+1$. But as a result, the constraint is stabilized, and the resulting equilibrium solution x_e is unique, isolated and exponentially stable. Numerical solutions will exhibit a small amount of error in the refrigerant mass during a simulation (as do any of the other states), on the order of the solver tolerance, but will not drift far from the initial condition because of the stabilization.

Implementing this approach for VCC models requires modifications to the base classes, which remains part of future work. Nevertheless, we illustrate the method using a simpler example in the following section.

4 Toy Pendulum Example

To show these results are not restricted to only VCC models, the dynamic analysis and remedies can be illustrated using the the quintessential Modelica example: The Cartesian pendulum, diagrammed in Figure 7, modeled as

$$\dot{x}_1 = v_1 \tag{18a}$$
$$\dot{x}_2 = v_2 \tag{18b}$$
$$\dot{v}_1 = -\lambda \cdot x_1 - b \cdot v_1 \tag{18c}$$
$$\dot{v}_2 = -\lambda \cdot x_2 - b \cdot v_2 - g_n \tag{18d}$$
$$0 = g(x_1, x_2) = x_1^2 + x_2^2 - L^2, \tag{18e}$$

where (x_1, x_2) is the Cartesian location of the bob, (v_1, v_2) is its velocity, λ is the Lagrange multiplier, g_n is the acceleration due to gravity, $b = 0.1$ is a damping coefficient introduced to stabilize the pendant position, and length $L = 1$. Realizing this model in Modelica, compiling in Dymola and simulating for $x_1(0) = 0.1$ gives the results in Figure 8, where Dymola reduces the index (from 3) and dynamically selects two states, here x_1 and v_1. There is of course no drift because the dimension of the dynamic states equals the number of degrees of freedom.

DOI
10.3384/ecp218345

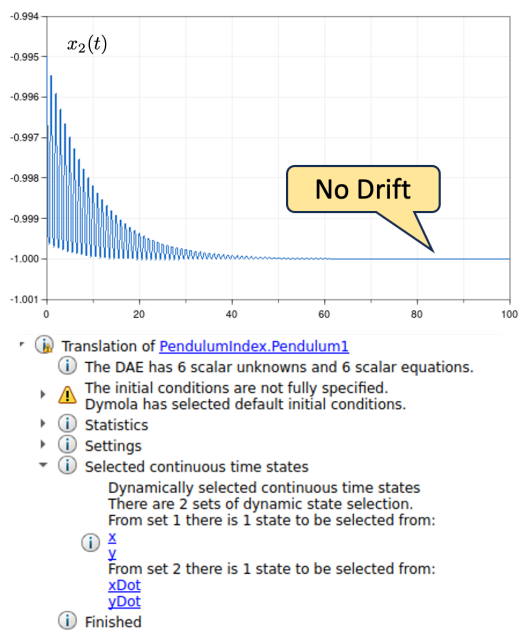


Figure 8. State selection results and simulation trajectory $x_2(t)$ of model (18).

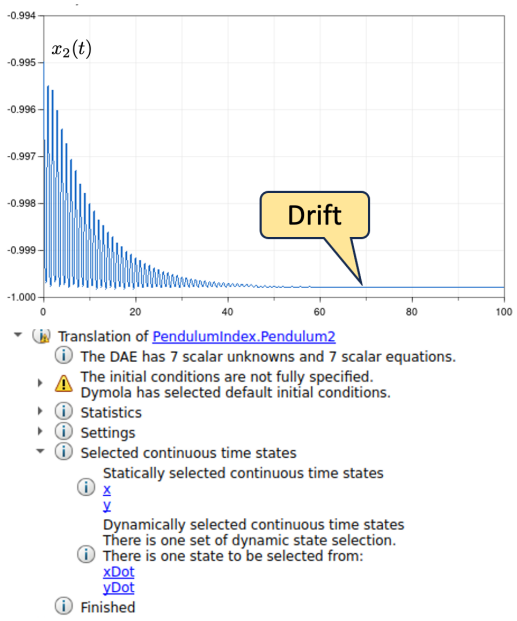


Figure 9. State selection results and simulation trajectory of $x_2(t)$ of model (19).

Now consider the system with the constraint (18e) replaced with its derivative,

$$\dot{x}_1 = v_1 \tag{19a}$$
$$\dot{x}_2 = v_2 \tag{19b}$$
$$\dot{v}_1 = -\lambda \cdot x_1 - b \cdot v_1 \tag{19c}$$
$$\dot{v}_2 = -\lambda \cdot x_2 - b \cdot v_2 - g_n \tag{19d}$$
$$0 = \dot{g}(x_1, x_2) = 2x_1 v_1 + 2x_2 v_2. \tag{19e}$$

This is analogous to VCC model with (19e) being similar to the mass flow rate constraint (2a). Realizing this model in Modelica, compiling in Dymola and simulating for $x_1(0) = 0.1$ gives the results in Figure 9, where Dymola reduces the index (from 2) and dynamically selects *three* states, here x_1, v_1 and x_2. Linearizing at the pendant equilibrium solution and computing eigenvalues results in the two stable eigenvalues that we expect, and a third eigenvalue at zero, associated with an eigenvector in the x_2-direction, precisely the direction of the missing constraint $g(x_1, x_2) = 0$ at the pendant equilibrium $(0, -1)$. The numerical solution is observed to drift along the vertical x_2 direction in Figure 9, away from the point $(0, -1)$.

Finally we augment (19) using Gear's method (Gear, 1988). The position constraint $0 = g(x_1, x_2)$ is added to the set, and the Lagrange multiplier μ multiplied by the gradient of g is added to the right-hand side of the ODEs (19a)-(19d), resulting in an index-2 DAE. Then we manually reduce the index from two to one using Baumgarte's method (Baumgarte, 1983), replacing the constraint equation $g = 0$ with $\dot{g} + \alpha_0 \cdot g = 0$, where $\alpha_0 > 0$, ($\alpha_0 = 1$ for convenience) resulting in the index-1 DAE

$$\dot{x}_1 = v_1 + \mu \cdot 2x_1 \tag{20a}$$
$$\dot{x}_2 = v_2 + \mu \cdot 2x_2 \tag{20b}$$
$$\dot{v}_1 = -\lambda \cdot x_1 - b \cdot v_1 \tag{20c}$$
$$\dot{v}_2 = -\lambda \cdot x_2 - b \cdot v_2 - g_n \tag{20d}$$
$$0 = 2x_1 v_1 + 2x_2 v_2 \tag{20e}$$
$$0 = \dot{g} + \alpha_0 \cdot (x_1^2 + x_2^2 - L^2). \tag{20f}$$

Again, there is a subtle but important point to consider: The $\dot{g}$ in (20f) is not the same as the $\dot{g}$ in (19e), because $\dot{x}$ has changed from the second to the third example. Realizing this in Modelica, compiling in Dymola and simulating for $x_1(0) = 0.1$ gives the results shown in Figure 10. Interestingly, Dymola reduces the index to 0, and dynamically selects three states, here x_1, v_1 and x_2, the same states selected for the second case. Linearizing the system at the equilibrium solution results in the two stable eigenvalues associated with the pendulum that we expect, and a third exponentially stable eigenvalue at $-\alpha_0$, associated with the stabilized constraint equation $\dot{g} + \alpha_0 \cdot g = 0$. The equilibrium solution is now exponentially stable. Although a numerical simulation of this system will suffer from a small amount of numerical error in the constraint ($g = 0$ is not exactly enforced), caused by Baumgarte's method (Bortoff, 2019) this can be controlled through the solver tolerance, and importantly the solution will not exhibit constraint drift, as shown in Figure 10.

5 Conclusion

Numerical simulations of some dynamic models of vapor compression cycles based on standard conservation equations may not conserve refrigerant mass, resulting in sim-

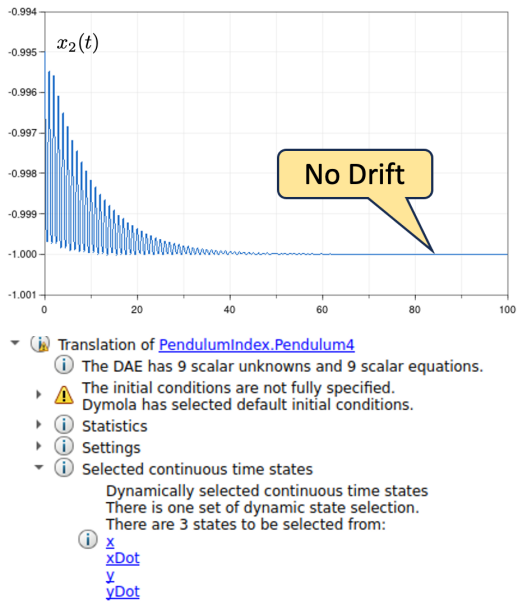


Figure 10. State selection results and simulation trajectory of $x_2(t)$ of model (20).

ulation failures especially for relatively long simulation times. This paper provides a dynamic analysis to ascertain mathematically the root cause of so-called refrigerant mass drift, resulting in an improved understanding of the dynamic model structure and behavior. The key result is that models derived using conventional assumptions and using conventional coordinates (two fluid states per control volume) are not exponentially stable, as is often assumed, and the n-dimensional state space may be decomposed into an $n-1$ dimensional integral manifold, and a transverse 1-dimensional equilibrium manifold. Numerical solutions will inevitably drift in the direction of the equilibrium manifold, manifesting as a varying refrigerant mass. Models that make use of higher dimensions such as (P,h,ρ) for each control volume, will exhibit higher dimensional equilibrium manifolds that may result in numerical drift in these spaces, leading to unphysical results. Two methods to remedy are proposed and shown to be effective with a thermofluid VCC example and also a classic pendulum example.

References

Numerical Heat Transfer and Fluid Flow. Hemisphere Publishing Co., 1980.

Daniel Anderson, John Batteh, Matthis Thorade, and Lixiang Li. Mass conservation in vapor compression cycles: A method for ensuring consistency with redundant dynamic states. In *Proceedings of the* 15th *International Modelica Conference*, pages 693–702, 2023.

J. W. Baumgarte. Stabilization of constraints and integrals of motion in dynamic systems. *Computer Methods in Applied Mechanics and Engineering*, 1:1–16, 1972.

J. W. Baumgarte. A new method of stabilization for holonomic constraints. *ASME Journal of Applied Mechanics*, 50:869–870, 1983.

Scott A. Bortoff. Using Baumgarte's method for index reduction in Modelica. In *Proceedings of the 13th International Modelica Conference*, pages 333–342, March 2019.

K. E. Brenan, S. L. Cambell, and L. R. Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. SIAM, 1996.

C. W. Gear. Differential-algebraic equation index transformation. *SIAM Journal on Scientific and Statistical Computing*, 9(1), 1988.

Alberto Isidori. *Nonlinear Control Systems*. Springer-Verlag, 1989.

Hassan K. Khalil. *Nonlinear Systems: Third Edition*. Prentice Hall, 2002.

Chris R. Laughman and Hongtao Qiao. On the influence of state selection on mass conservation in dynamic vapor compression cycle models. *Mathematical and Computer Modeling of Dynamical Systems*, 23(1):262–283, Dec. 2016.

Christopher R. Laughman and Hongtao Qiao. Mass conserving models of vapor compression cycles. In *Proceedings of the* 11th *International Modelica Conference*, pages 759–767, Versailles, France, 2015.

Hubertus Tummescheit. *Design and Implementation of Object-Oriented Model Libraries using Modelica*. PhD thesis, Lund Institute of Technology, 2002.

M. Vidyasagar. *Nonlinear Systems Analysis: Second Edition*. Prentice-Hall, 1993.

DOI
10.3384/ecp218345

Modelling and Impact of Hydraulic Short Circuit Operation in Pumped Hydro Energy Storage

Inga Beyers[1] Lukas Krebeck[1] Astrid Bensmann[1] Richard Hanke-Rauschenbach[1]

[1]Leibniz University Hannover, Institute of Electric Power Systems, Appelstraße 9A, 30167 Hannover, Germany

Abstract

This paper investigates the hydraulic short circuit (HSC) operation of pumped hydro energy storage systems, specifically how it can be modelled and its impact on energy capacity, efficiency and operating range. Hydraulic short circuit operation enables power modulation during charging for fixed-speed pumped hydro systems, thereby allowing them to operate more flexibly. In this work, a pumped hydro storage system with 140.5 MW nominal charging power and 10 h nominal charging time is analysed. The storage is simulated via a dynamic, system-level model composed of detailed physical component models to accurately depict the HSC operating behaviour. HSC enables part-load operation from 39 MW to 107 MW but with significantly lower overall system efficiencies. The HSC operating limits are governed by the turbine's minimum flow and maximum power. The custom-built OpenModelica library employed for this analysis can be downloaded from `https://github.com/ibeyers/pumped-storage-lib`.

Keywords: energy storage, pumped hydro, hydraulic short circuit, flexibility, off-design operation

1 Introduction

This paper focuses on the hydraulic short circuit operation mode of pumped hydro energy storage systems. The role and operation of pumped hydro energy storage in the European electricity system have changed following the liberalisation of energy markets and the increasing integration of fluctuating renewable energy sources. Moving away from traditional arbitrage operation, pumped hydro systems now often participate in multiple markets and generate additional revenue from ancillary services, such as frequency regulation (Pérez-Díaz et al. 2015). As a consequence, the operational flexibility of pumped hydro storage systems has become a key focus area in the field of pumped hydro research and practice (Giese and Magnoli 2019). This is an issue because the operational flexibility of traditional fixed-speed pumped hydro is severely limited on the charging side. Since the power consumption of a fixed-speed pump is governed by the system head, deliberate power modulation is not possible during charging (Valavi and Nysveen 2018)[1]. Pumped hydro operators increasingly retrofit existing systems to enable more operational flexibility, while flexibility is already a central design consideration for new systems (Vagoni et al. 2024).

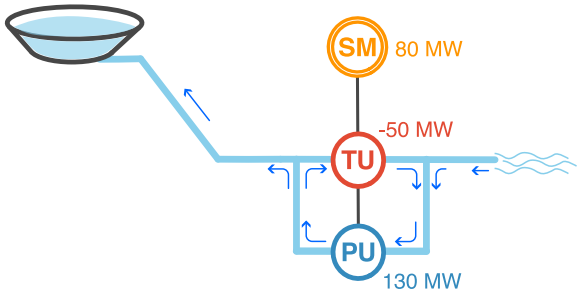


Figure 1. Illustration of hydraulic short circuit operation in a ternary machine set (pump, turbine and synchronous machine). During charging, a part of the flow is recirculated to the turbine, which delivers power and therefore reduces the system's net power consumption.

One approach to counteract this inflexibility and enable deliberate power modulation during charging is the so-called "hydraulic short circuit" (HSC) operation. Here, a portion of the pumping flow is redirected to a turbine, whose power delivery can be adjusted via the guide vane opening. This allows the system's net power consumption to be regulated by adjusting the turbine power delivery. This principle is illustrated in Figure 1. This mode requires at least two hydraulic machines, either in a single ternary machine set or two pump-turbines, with one operating as a pump and the other as a turbine. The advantage of HSC is that it can be retrofitted with relatively low capital expenditure, while the disadvantage is the low system efficiency (Vagoni et al. 2024). We recently investigated the operational flexibility and performance of different pumped hydro configurations, including HSC operation (Beyers, Krebeck, et al. 2025). In this work, we build on that previous work but focus on three aspects of hydraulic short circuit not yet addressed in the literature.

The first aspect is the implementation of the library, which was developed to simulate pumped hydro systems and HSC operation in the OpenModelica environment (Fritzson et al. 2020). Pumped storage research is

[1]Strictly speaking, it is not impossible but rather not advisable. Theoretically, pump throttling can be used to achieve a degree of freedom in pump operation (Alizadeh Bidgoli, Yang, and Ahmadian 2020). However, the resulting damage to the pump is unacceptable and pump throttling, therefore, does not feature in the continuous operation of pumped hydro storage systems

often conducted using proprietary software, and the results are then shared without a deeper insight into the underlying methodology and implementation, see, e.g. Ruppert et al. (2017) and Geiger and Riedelbauch (2023). Open source modelling can therefore contribute to open research and accelerate R&D efforts in this field. A well-executed open hydropower library for Modelica (OpenHPL) developed by Vytvytskyi (2019) exists; however, it currently cannot model the bidirectional operation of pumped hydro. Therefore, a custom library was developed[2]. The second aspect is an elaboration of the flow regimes that ensue during HSC operation. So far, the HSC mechanism in the entire hydraulic network has not been demonstrated in detail in the literature. By understanding the HSC mechanism, the impact of HSC operation on energy capacity, efficiency and operating range can be demonstrated very clearly. The third aspect is extending HSC with a thermal analysis. Practical experience at the Frades II pumped hydro plant has shown that hydraulic short circuit operation can lead to inadmissibly high water temperatures since the losses of the hydraulic machines heat up the circulating water (Pinto et al. 2024). This could significantly impact continuous operation and make HSC less suitable for retrofits of certain pumped hydro system types. To the authors' best knowledge, this aspect has never been formally investigated and therefore, warrants a first exploration.

Our paper will, therefore, deliver the following three contributions:

1. Focus on implementation aspects of modelling HSC.
2. A more detailed elaboration of flow regimes during HSC operation.
3. An investigation of non-isothermal HSC.

This paper is structured as follows: Section 2 details the system under consideration. Section 3 presents an overview of the methodology, including the implementation in a custom-built OpenModelica library. Section 4 presents the results and Section 5 concludes this work.

2 System under consideration

The system analysed in this work is a synthetic pumped hydro system. This is done to avoid a case-specific bias and instead develop general-purpose insights. The hydraulic scenario is taken from previous work (Beyers, Krebeck, et al. 2025). Key characteristics are:

- Upper reservoir with a geodetic head $\overline{h}_{\mathrm{geo}}$ of the reservoir water level ranging from 480 m (empty) to 520 m (full).
- Circular upper reservoir with tapered walls, with an opening angle of 30°.
- Upper reservoir sized so that 100 MW can be delivered for 10 h.
- Lower reservoir with an infinite capacity, such as a river or lake, which acts as a boundary with a constant head.
- Total waterway distance of 2.4 km.

The 500 m average geodetic head is explicitly chosen to keep this a generic scenario where all technical configurations are viable.

Within this hydraulic scenario, we consider a single ternary machine set with a dedicated pump and a Francis turbine. The machine set shaft is connected to a salient-pole synchronous machine. A block transformer steps up the synchronous machine voltage to the grid voltage (or steps down, depending on the direction of power flow). The hydraulic and electrical machines are sized to deliver 100 MW nominal discharging power and consume 140.5 MW nominal charging power at the grid connection point with a power factor $\cos(\varphi)$ of 0.9. In contrast to the previous paper, the machines here are sized for a symmetric charging and discharging duration of 10 h, resulting in higher charging power. Key parameters are summarised in Table 1.

Table 1. Selected parameters of hydraulic and electrical machines.

Component	*Parameter*	*Value*
Transformer	Nom. apparent power $S_{\mathrm{tr}}^{\mathrm{nom}}$	156 MVA
	Nom. high voltage $U_{\mathrm{tr,hv}}^{\mathrm{nom}}$	220 kV
Synchronous Machine	Nom. apparent power $S_{\mathrm{sm}}^{\mathrm{nom}}$	156 MVA
	Nom. stator voltage $U_{\mathrm{sm,st}}^{\mathrm{nom}}$	10.5 kV
	Nom. rotor speed $n_{\mathrm{sm,ro}}^{\mathrm{nom}}$	500 rpm
Turbine	Number of stages N_{s}	1
	Specific speed of stage $n_{\mathrm{q,tu}}^{\mathrm{nom}}$	24.0
Pump	Number of stages N_{s}	2
	Specific speed of stage $n_{\mathrm{q,pu}}^{\mathrm{nom}}$	41.0

3 Methodology

This section is divided into three parts: Section 3.1 describes the general modelling approach, Section 3.2 presents an overview of the components, and Section 3.3 discusses select aspects of the pumped storage library.

3.1 Modelling approach

In this work, we employ a system-level model formulation to account for all machines and their interactions up to the grid connection point. The model combines dynamic components and quasi-stationary components. The components are connected to each other to form a system model. Because the focus is on continuous operation, the system's dynamics are governed by the reservoir. The

[2]It should be noted that the library is currently under development. Its purpose is to answer the specific questions pursued by the authors and therefore does not cover all aspects of pumped hydro storage.

DOI 10.3384/ecp218353

filling and emptying of the reservoir result in a head variation, which forces the rest of the components into off-design operation. Faster dynamics (e.g., the inertia of rotating masses in machines) can be neglected compared to the slow dynamics of the reservoir. The other component models are therefore considered quasi-stationary; they undergo a sequence of stationary operating points set by the dynamic reservoir head variation. The system is simulated with a so-called "hot start" at nominal speed, i.e. the start-up sequence from standstill is not modelled. The passive sign convention is used in this work. Quantities flowing into a component (or the system) are positive, whereas quantities flowing out are negative. At the grid connection point, charging power is positive and discharging power is negative.

3.2 Component models

Our previous publication (Beyers, Krebeck, et al. 2025) describes the component models in detail. This section briefly reiterates key aspects.

Upper reservoir, conduits and point losses. The hydraulic waterway elements and the hydraulic machines are all characterised by head h and volumetric flow q, as is common in the field of fluid systems. The head is the total mechanical energy of a flow, expressed in a height in meters:

$$h = \frac{p(t)}{\rho g} + \frac{v(t)^2}{2g} + h_{\text{geo}} \tag{1}$$

It is composed of the pressure head $p(t)/\rho g$, the velocity head $v(t)^2/2g$ and geodetic head h_{geo}. Water is considered an incompressible medium with a constant density of $\rho = 1000\,\text{kg/m}^3$.

The reservoir is governed by a dynamic mass balance, expressed here through the reservoir volume V_{ur}:

$$\frac{\mathrm{d}V_{\text{ur}}}{\mathrm{d}t} = q(t) \tag{2}$$

The reservoir is characterised by a State-of-Charge (SOC) that relates the volume contained in the reservoir to the nominal water volume:

$$\text{SOC}(t) = \frac{V_{\text{ur}}(t)}{V_{\text{ur}}^{\text{nom}}} \tag{3}$$

The water volume in the upper reservoir and the water level $x(t)$ are linked through the geometric shape of the reservoir.

$$V_{\text{ur}}(t) = \frac{x(t)\cdot\pi}{3}\cdot\left(R(t)^2 + R(t)\cdot r + r^2\right) \tag{4}$$

with the radius at the bottom r and the top of the water volume $R(t) = r + x(t)\cdot\tan(30^\circ)$.

The components through which the water is transported are either conduits (distributed "pipe" elements such as the headrace, the penstock and the tailrace) or point losses, which include all other elements, (e.g., intake, rake, bends) that incur *local* losses. They are modelled as lumped, static components with conservation of mass and energy[3]. Conservation of energy is expressed through the extended Bernoulli equation (Pritchard and Leylegian 2011), which includes losses in the form of a friction head h_{fric}:

$$h_{\text{in}}(t) = h_{\text{out}}(t) + h_{\text{fric}}(t) \tag{5}$$

In this formulation, the change in internal thermal energy of the water through the friction losses is neglected. The friction head for the conduits with the length l, diameter d and friction factor $\lambda(t)$ is:

$$h_{\text{fric}}(t) = \lambda(t)\cdot\frac{l}{d}\cdot\frac{v(t)\cdot|v(t)|}{2g} \tag{6}$$

For the point losses the term $\lambda(t)\cdot\frac{l}{d}$ is replaced by an empirical friction factor ζ specific to each component.

Hydraulic machines. The hydraulic machines are modelled with mechanistic models, which combine Euler equations that reflect ideal machine behaviour with empirical/semi-empirical loss models. The pump model is based on Tuzson (2000), and the turbine model is from Nielsen (2015) and Reines and Svingen (2020).

To demonstrate the models, the parameterised pump and turbine performance maps are shown in Figure 2 a) (pump) and Figure 2 b) (turbine). Figure 2 a) contains the pump curves, plotting head over volumetric flow rate, combined with isolines for speed (dash-dotted) and relative efficiency (solid). The pump always operates at best guide vane opening, eliminating this variable from the pump map. The turbine map in b) is visualised differently since the adjustable guide vane opening offers an additional degree of freedom in operation. The map plots the volumetric flow rate over the turbine speed, with isolines of the same guide vane opening (dashed) and relative efficiency (solid). The turbine performance map is shown for the nominal head.

Electrical machines. Both the synchronous machine and the transformer are modelled with complex phasor equivalent circuit models (ECM), with constant parameters. This is a standard modelling procedure to describe stationary and quasi-stationary behaviour. It is described in reference works, such as Müller and Ponick (2009) and Oeding and Oswald (2016). The synchronous machine has an internal-voltage-based equivalent circuit model. It is complemented by separate loss functions, e.g., for iron losses (Engevik, Valavi, and Nysveen 2016), windage losses and additional losses (Pyrhönen, Jokinen, and Hrabovcová 2013).

[3]For steady, incompressible flow along a streamline, conservation of momentum leads to the same expression as conservation of energy (Pritchard and Leylegian 2011)

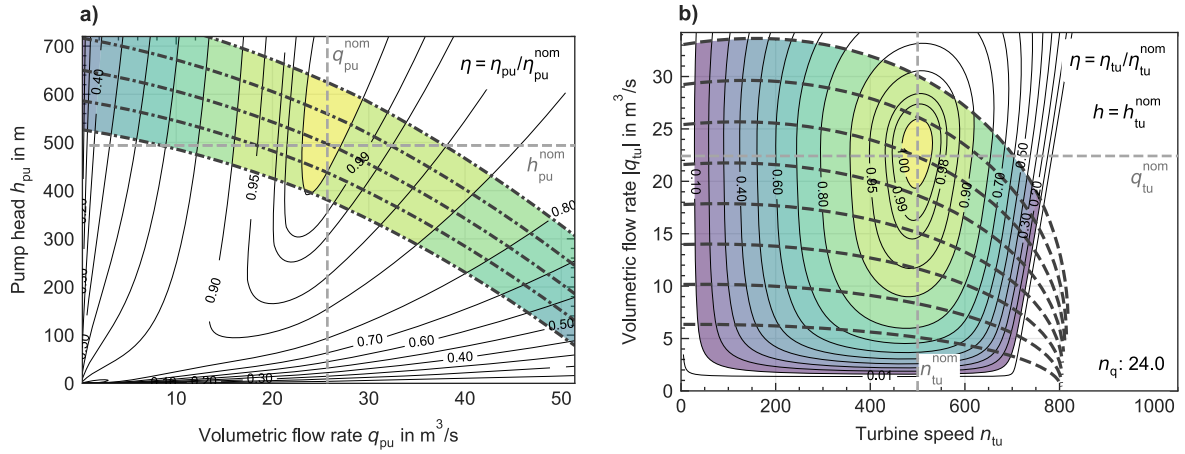


Figure 2. Parameterised pump model in a) and turbine model in b), shwon as performance maps. In the pump map, the dash-dotted curves are isolines of the same speed. The solid lines are isolines for the relative efficiency $\eta_{pu}/\eta_{pu}^{nom}$. In the turbine map, the dashed curves are isolines of the same guide vane opening. The solid lines are isolines for the relative efficiency $\eta_{tu}/\eta_{tu}^{nom}$.

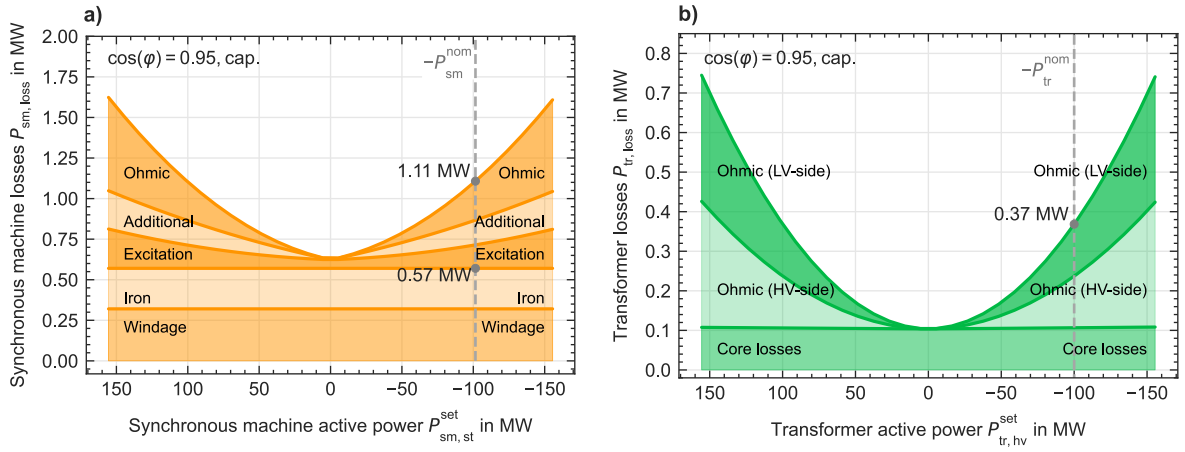


Figure 3. Loss breakdown of the parameterised synchronous machine model in a) and transformer model in b). Synchronous machine losses are plotted as a function of the active power on the stator side, $P_{sm,st}$, while the transformer losses are plotted as a function of the active power on the high-voltage side, $P_{tr,hv}$.

To provide insight into the model, Figure 3 a) shows the losses of the parameterised synchronous machine. The losses are plotted over the stator-side active power $P_{sm,st}$ at a capacitive power factor of $\cos(\varphi) = 0.95$ and nominal speed. The left side of the figure shows the motor mode (positive values, power drawn by the SM) and on the right side is the generator mode (negative values, power delivered by the SM). The losses are non-linear. At nominal discharging power, approximately 51 % of losses are constant losses (windage, iron) and 49 % are load-dependent.

The transformer is modelled with a T-shaped equivalent circuit model. The T-ECM includes a shunt element, which represents the core losses and causes a significant efficiency drop in low part-load. To illustrate the model's behaviour, the losses of the parameterised transformer are shown in Figure 3 b). The losses are plotted over the high-voltage-side active power $P_{tr,hv}$ at a capacitive power factor of $\cos(\varphi) = 0.95$. The losses are dominated by the Ohmic losses at nominal discharging power.

3.3 Implementation in a custom OpenModelica library

For this analysis, a custom-built OpenModelica library is employed, making use of the features of the Modelica language, i.e. multi-physics, equation-based and object-oriented modelling. General-purpose libraries lack required components for this analysis (e.g. a salient-pole synchronous machine) and are not necessarily parameterised for the scale of pumped hydro components. Therefore, the component models have been developed from the ground up to meet the specific modelling needs and model depth required. However, the custom models use the Modelica standard library for, e.g., units, icons, control elements, and connectors. The parameterisation of the component models is performed in separate Python scripts, where the resulting parameter sets are then copied to OpenModelica. The models used for this analysis are found in a subpackage called "ModelicaConference2025".

DOI 10.3384/ecp218353

Key aspects of the library are discussed in the following paragraphs.

The hydraulic elements, such as conduits, point losses and reservoirs, are modelled in an *acausal* manner. This has the advantage that a single model representation can be used for all modes of operation since the flow direction is not prescribed. This is important, as flow direction can vary between regular charging, regular discharging and hydraulic short circuit operation.

The library uses Modelica's object-oriented nature. There are four separate model layers: component models, subsystem models, full system models, and the control layer. The smallest unit is a component, where a "flat" model formulation is used. The hydraulic components are instantiated to build a hydraulic subsystem, referred to as the "waterway." The lower reservoir is a sink, and only the head is specified as a boundary condition. The waterway is then extended by the conversion machines (pump, turbine, synchronous machine, and transformer) to form a full system. Separating the waterway from the conversion machines has the advantage that different machine types and sizes can be tested in the same hydraulic system. Finally, the control layer is added by extending the full system, so that experimentation with different control strategies is possible. Where possible, ideal controllers in the form of implicit equations are implemented. Where explicit control loops are necessary, such as setting the turbine power to achieve a specified power at the grid connection point during HSC, they are implemented as PID (proportional-integral-derivative) control.

The full system model is shown in Figure 4. The shaft power outputs and inputs from the hydraulic machines are connected to the synchronous machine with causal input/output connectors. The electrical machines are only connected via causal input/output connectors, each with a set of connectors for power consumption and delivery. They are connected via sum-to-zero coupling of active and reactive power. The full system model includes auxiliary losses, which are modelled as an active power loss between synchronous machine and transformer. During discharging, the active power is set at the grid connection point and the power requirements are propagated back through the transformer and synchronous machine to the turbine component. During charging, the active power required by the pump is propagated in the other direction towards the grid connection point. The reactive power is determined from the active power at the grid connection point via a constant power factor in operation of 0.95 and communicated back to the synchronous machine component. In hydraulic short circuit operation, the turbine power is controlled via a PID controller to maintain the constant charging power setpoint at the grid connection.

4 Results and Discussion

The results are divided into four subsections. Section 4.1 analyses the impact of HSC operation at a single point in time, focusing on the flow conditions, the power split between machines, and the resulting efficiency. In Section 4.2, an entire HSC charging process is simulated over time. In Section 4.3, multiple HSC and regular charging/discharging processes are aggregated to quantify the system's energy capacity-power relationship, efficiency-power relationship, and operating range. Finally, Section 4.4 contains a brief consideration of the thermal behaviour.

4.1 Hydraulic short circuit operation at different power levels

To elaborate on the effects of HSC operation, the flow conditions and further variables are shown for two snapshots in time for 50 MW and 60 MW part-load charging, respectively, at a SOC of 0.5. Figure 5 is for a charging power of 50 MW at the grid connection point. At a SOC of 0.5, 50 % of the nominal water mass is stored in the reservoir and the total reservoir head is 502.77 m (comprising 480 m elevation at the reservoir bottom and a water level of 22.77 m). Recall that the head in the upper reservoir varies between 480 m (empty) and 520 m (full), yielding a relative water level of $22.77\,\mathrm{m}/40\,\mathrm{m} = 56.9\%$. Due to the tapered geometry, the relative water level and SOC differ.

The head values are shown in red at every connection point between two components. This is the total head from the extended Bernoulli equation, composed of pressure head, velocity head and geodetic head. These three contribute differently depending on the location in the waterway, but the proportions are not detailed here. Furthermore, the constant water level of the lower reservoir is set to $h = 0\,\mathrm{m}$, thus the total head is given relative to this reference level. The head difference between different points of the waterway displayed in Figure 5 is thus solely due to the friction losses in the conduits and fittings.

The pump has a power consumption of 134.55 MW at nominal speed and a pump head of $h_{\mathrm{pu}} = 504.91\,\mathrm{m}$. The resultant flow in this operating point is $24.8\,\mathrm{m}^3/\mathrm{s}$. The flow values are printed in blue above or beside the relevant conduits. The turbine needs to deliver -86.16 MW to the shaft to achieve 50 MW active power consumption by the transformer at the grid connection point. For this, $19.5\,\mathrm{m}^3/\mathrm{s}$ of the flow is redirected to the turbine. Only the remaining $5.3\,\mathrm{m}^3/\mathrm{s}$ flows up to the reservoir. Because of the low net flow in large parts of the waterway, the head losses are also comparatively low. For reference, in regular charging/discharging operation the head loss due to friction is $> 10\,\mathrm{m}$. In this HSC operating point, the pump must overcome only $504.91\,\mathrm{m} - 502.77\,\mathrm{m} = 2.15\,\mathrm{m}$ of total friction head loss for the entire waterway.

The net shaft power supplied by the synchronous machine rotor is 48.39 MW. The losses of the synchronous machine, the auxiliary losses, and the transformer losses are added to yield a total active power consumption of $P_{\mathrm{gc}} = 50\,\mathrm{MW}$ at the grid connection point. The total system efficiency can be calculated by dividing the hydraulic power flowing into the reservoir P_{ur} by the active power

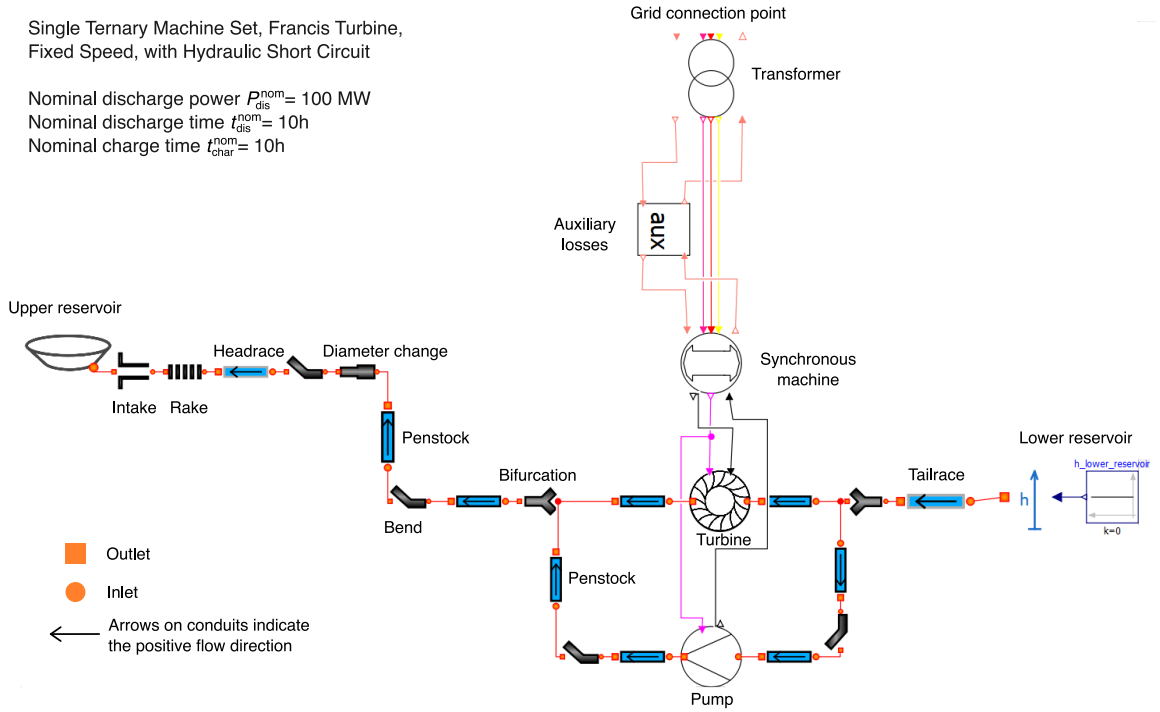


Figure 4. Full system model as implemented in OpenModelica. Note that because this is an abstract representation of the pumped hydro system, the dimensions of the waterway are not to scale.

supplied at the grid connection point P_{gc}:

$$\eta_{sys}(t) = \frac{P_{ur}(t)}{P_{gc}(t)} \tag{7}$$

with $P_{ur}(t) = h_{ur}(t) \cdot q(t) \cdot g \cdot \rho$. This operating point achieves an efficiency of 52.28 %, which is relatively low compared to efficiencies achieved in normal charging or discharging operations. It is important to note that the hydraulic short circuit efficiency cannot be estimated by multiplying all the component efficiencies because the power flow is not unidirectional. In hydraulic short circuit, there are two opposing power flows, each with its associated losses, which is the reason for the low total system efficiency. Figure 6 shows the same variables for 60 MW part-load and a SOC of 0.5, so the upper reservoir head ($h_{ur} = 502.77$ m) is identical to that in Figure 5. Conceptually, part-load charging with HSC is the same for 50 MW and 60 MW, however, three key aspects are highlighted below to better understand HSC operation. The first aspect is the complementary relationship between turbine power and charging power: a *low* HSC charging power (more part-load) requires higher turbine output (more negative power), while a high HSC charging power (less part-load) requires lower absolute turbine power. The second aspect is that the system efficiency is strongly influenced by the proportion of flow recirculated through the turbine. At 60 MW HSC charging power, less flow is recirculated through the turbine than at 50 MW ($17.3\,m^3/s$ vs. $19.5\,m^3/s$). Thus, more net flow reaches the upper reservoir, and the system efficiency is ultimately higher. The higher the flow through the turbine, the lower the system efficiency. Thirdly, the pump operating point is slightly different at 60 MW and SOC = 0.5 than at 50 MW and SOC = 0.5 because of the head friction losses. The headrace and the long diagonal penstock are the main drivers of the head friction losses. At 60 MW, the flow through these conduits is slightly higher, incurring higher losses, requiring the pump to supply a higher head (505.49 m vs 504.91 m at 50 MW). This shifts the pump operating point upwards and to the left on the pump curve (refer to Figure 2 a), which means the pump flow is slightly lower at $24.7\,m^3/s$ vs. $24.8\,m^3/s$ at 50 MW.

4.2 Hydraulic short circuit operation over time

Figure 7 demonstrates a complete 50 MW, constant-power HSC charging process over time to showcase the system dynamics. The simulation takes 4.6 s of computation time using a time step of 10 s. The upper reservoir starts at SOC = 0 and is terminated at SOC = 1 with a charging time of 44.5 hours. Figure 7 a) shows the pump head (solid line) and flow (dashed line) on the y-axis over the charging time on the x-axis. The main dynamic is the water level variation in the upper reservoir. Consequently, as the charging progresses, the pump sees an increasing head from 482.5 m to 521.8 m as the reservoir is filled and the water level rises. As the head increases, the pump flow decreases. On the pump curve (refer back to Figure 2 a), this corresponds to a leftward shift along the curve. The pump power consumption is largely governed by the pump flow.

DOI 10.3384/ecp218353

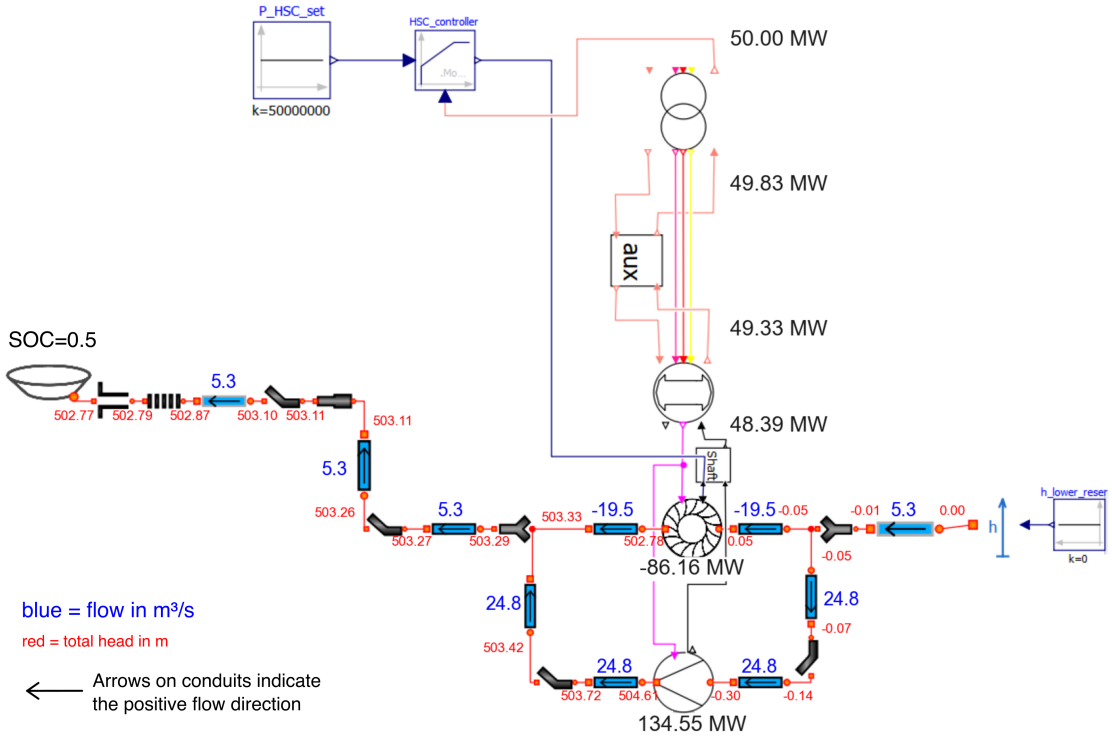


Figure 5. Snapshot in time of key variables of the pumped storage system during hydraulic short circuit operation. State-of-Charge is 0.5 and the charging power at the grid connection point is 50 MW. The flow values are printed in blue above or beside the relevant conduits. The head values are printed in red at every connection point between two components. The power values are printed in black next to the machines.

Therefore, as the flow decreases, the pump power at the shaft $P_{\mathrm{pu,sh}}$, shown in blue in Figure 7 b), decreases as well from 142.6 MW (start) to 128.1 MW (end). To achieve a constant power consumption of 50 MW at the grid connection point, the turbine power is controlled accordingly, by adjusting the guide vane opening. As shown in the red curve in Figure 7, the absolute turbine power $|P|_{\mathrm{tu,sh}}$ decreases with the pump power, enabling a constant power supply at the rotor of the synchronous machine ($P_{\mathrm{sm,ro}}$ in yellow). The total system efficiency is shown in Figure 7 c). The system efficiency starts at 50.7 %, achieves a maximum value of 52.4 % at 27.7 hours and then drops to 51.7% at the end.

4.3 Energy capacity, efficiency and operating range of the pumped hydro system.

Depending on the power supplied to or discharged from the system, the pumped storage system has a different energy capacity and system efficiency. These are referred to as the energy-power ($E-P$) and efficiency-power ($\eta-P$) relationships, which are presented and discussed in this section. To visualise these relationships, we employ the Ragone plot framework (Beyers, Bensmann, and Hanke-Rauschenbach 2023), which is briefly reiterated here. A constant-power charging process is simulated, starting from an empty system at SOC=0 and continuing until the process is terminated. Two quantities are computed from the charging process: the electrical energy capacity, calculated as $E = \int_0^{t=t_{\mathrm{end}}} P_{\mathrm{gc}} \mathrm{d}t$, and the average system efficiency, calculated as $\overline{\eta}_{\mathrm{sys}} = \int_0^{t=t_{\mathrm{end}}} \eta_{\mathrm{sys}} \mathrm{d}t / t_{\mathrm{end}}$. This process is repeated for a different P_{c} = constant charging power. All the capacity and average efficiency values are then combined and plotted in a single curve. The procedure for discharging is the same, except that the constant-power discharging starts at a SOC of 1 and continues until termination.

Figure 8 plots the resulting $\eta-P$ and $E-P$ curves versus the full range of possible charging/discharging powers on the x-axis. Regarding the visual presentation, the charging side is shown on the left, and the discharging is on the right. The x-axis ranges from high charging powers (left) to high discharging powers (right), with 0 MW at the center. Figure 8 a) shows the average efficiency $\overline{\eta}_{\mathrm{sys}}$. The first insight is that not all power levels are possible. The operating range is discontinuous, and there are three separate elements in the $\eta-P$ curve, each belonging to a mode of operation. The modes are annotated as follows: "1P" is charging with one pump, "1P1T" is the hydraulic short circuit charging with one pump and one turbine simultaneously, and "1T" is discharging with one turbine. 1T is possible between -36.1 MW and -102.9 MW. The system efficiency is generally high but drops in part-load and over-load. A charging part-load lower than 36.1 MW is not possible, because the turbine component has an built-in flow limit of $q_{\mathrm{tu}}^{\mathrm{min}} = 0.4 \cdot q_{\mathrm{tu}}^{\mathrm{nom}}$ to avoid cavitation phenomena in part-load (Pereira Jr. et al. 2019). Over-load is

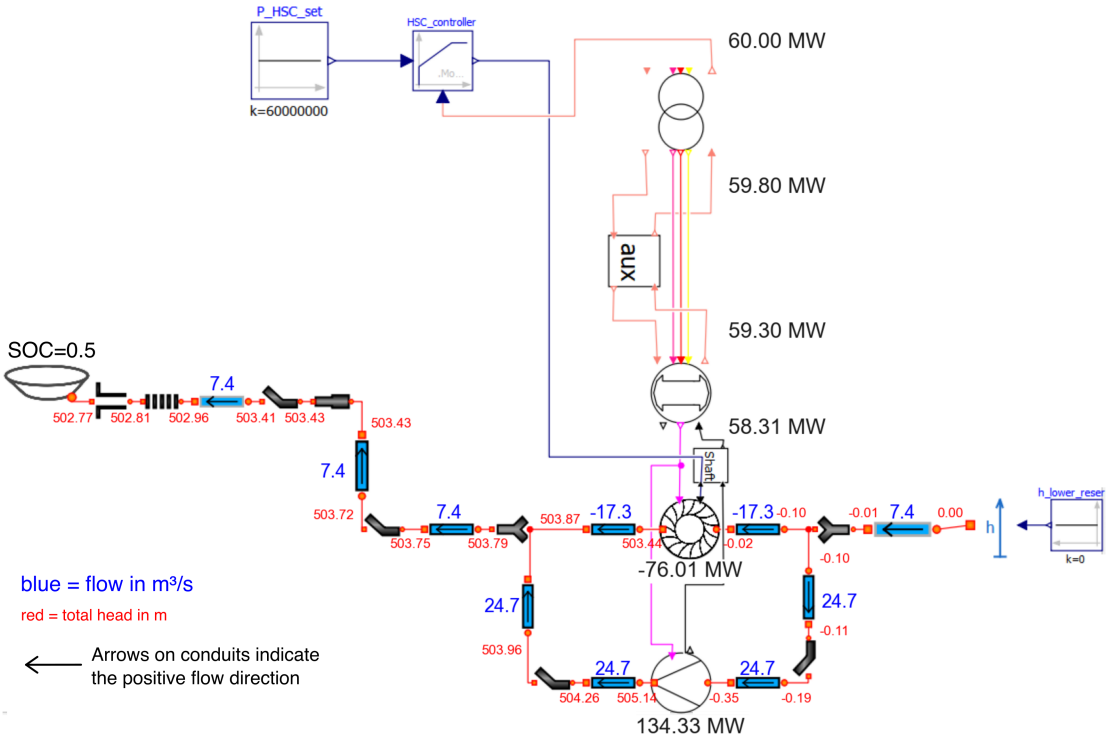


Figure 6. Snapshot in time of key variables of the pumped storage system during hydraulic short circuit operation. State-of-Charge is 0.5 and the charging power at the grid connection point is 60 MW. The flow values are printed in blue above or beside the relevant conduits. The head values are printed in red at every connection point between two components. The power values are printed in black next to the machines.

limited by the maximum turbine power of -105 MW.

Regular charging only with the pump (1P) does not comply with the constant-power framework since power variation is always present in fixed-speed configurations. To include regular charging anyway, its average power is shown as a single dot to distinguish it conceptually from the constant-power 1P1T and 1T operation. Hydraulic short circuit charging (1P1T) is possible between 39.8 MW and 107.1 MW. As discussed in Section 4.1, the efficiencies are generally lower than in regular operation and decrease further in part-load as more flow is diverted to the turbine. The turbine governs the HSC operating range. Because of the opposing relationship between turbine power and HSC charging power, the *minimum* HSC charging power of 39.8 MW is set by the *maximum* turbine power and the *maximum* HSC charging power of 107.1 MW is limited by the *minimum* turbine flow.

Figure 8 b) shows the energy capacity added to the system (charging) or removed from the system (discharging). E and P are linked by the charging/discharging duration. In Figure 8 b), dashed isochrone lines with varying slopes are plotted to mark points in the $E-P$ plane with identical charging/discharging durations. As a further reference, the energy stored in the full upper reservoir at SOC $= 1$ is plotted as a horizontal solid line. In discharging operation (1T), less energy capacity can be taken from the system than stored in the upper reservoir. In charging operation (1P), more energy capacity must be added to the system than ends up in the reservoir. In HSC operation (1P1T), the energy capacity curve has a pronounced turning point at 90.6 MW, which is not reflected in the efficiency curve. To understand this, recall that during a constant-power HSC charging process, the turbine power is continually lowered to match the lowering pump power (refer to Figure 7). At power values higher than 90.6 MW, the charging process is terminated prematurely because the minimum flow limit of the turbine is reached. The higher the HSC charging power, the earlier this limit is reached. Therefore, the capacity that can be added to the system drops until $P_{gc} = 107.1$ MW, which can only be supplied for 10 minutes[4].

In summary, HSC allows operational flexibility in part-load but at a significantly reduced efficiency. At higher HSC charging powers (closer to full load), the charging processes are cut short, significantly reducing charging energy capacity. At deeper part-load, the average system efficiency drops significantly.

4.4 Non-isothermal hydraulic short circuit operation

The original model formulation does not consider thermal effects since they are negligible during regular operation.

[4] Although E theoretically approaches zero, this is not considered relevant for continuous operation.

DOI 10.3384/ecp218353

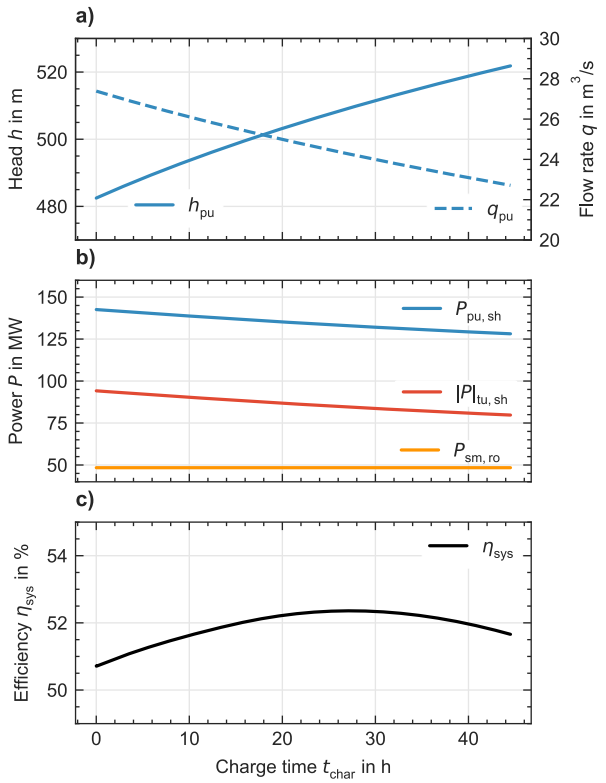


Figure 7. HSC operation at a charging power of $P_{gc} = 50\,\text{MW}$ simulated over time. Subfigure a) shows pump head and volume flow rate, b) shows the power values of turbine, pump and synchronous machine, while c) shows the system efficiency.

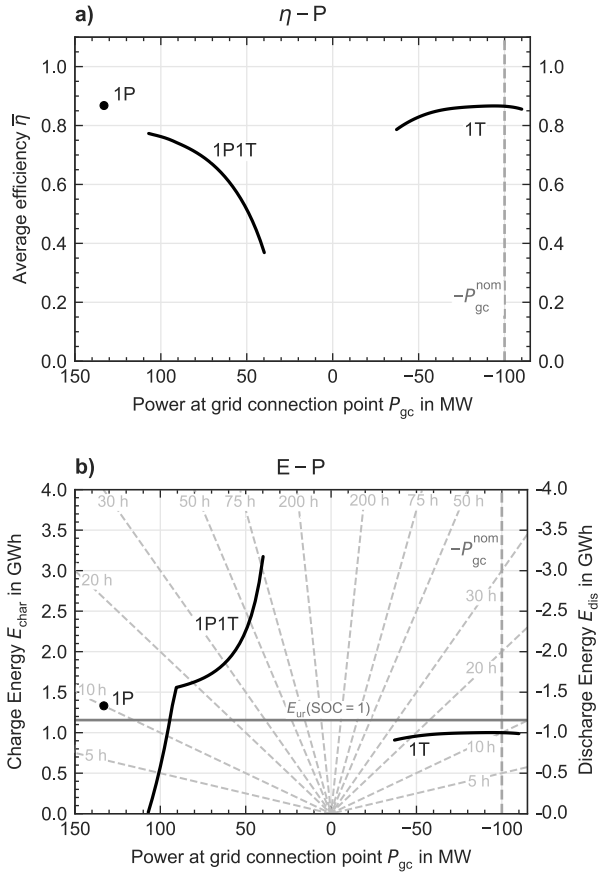


Figure 8. Efficiency-power ($\eta - P$) curves in a) and energy-power ($E - P$) curves in b) of the pumped hydro system at hand.

However, in a hydraulic short circuit, the water circulating between the machines can heat up to unacceptably high temperatures. An energy balance considering the internal thermal energy of the water is incorporated into the model to explore the thermal effects. For simplicity, we consider the entire pipe network of the hydraulic short circuit loop around the ternary machine set as a single control volume with a uniform temperature. In this control volume, the water mass is assumed constant since the incoming flow from the lower reservoir equals the outgoing flow to the upper reservoir and the variation in water density is negligible. The assumption of a uniform temperature in the HSC loop is an approximation, but the flow speeds are so large within the HSC loop that they override local effects on the timescales under analysis. The energy balance, assuming a constant water density, is:

$$m_{\text{HSC}} \cdot c_p \cdot \frac{dT_{\text{HSC}}}{\text{d}t} = \dot{m}_{\text{in}} \cdot c_p \cdot T_{\text{in}} + \dot{m}_{\text{out}} \cdot c_p \cdot T_{\text{HSC}} + P_{\text{loss,pu}} + P_{\text{loss,tu}} + P_{\text{fric}}. \quad (8)$$

The final temperature for a complete HSC charging process at $P_{gc} = 60\,\text{MW}$ is 20.72°C. For 50 MW, this becomes 21.05°C; for 39 MW, the final temperature in the HSC loop is 21.98°C. Therefore thermal effects are not an issue in the operating range determined by the machine sizing. The magnitude of the flow out of the HSC loop towards the upper reservoir is large enough to transport heat generated by losses away. A test in which the same system is equipped with a larger turbine reveals that the temperature within the HSC loop only becomes critical as flow toward the upper reservoir approaches zero. In that case, the relatively small water mass within in the HSC loop start notably accumulating the heat from machine losses since the heat is not transported away and into the upper reservoir. Critical temperatures exceeding 80°C are reached.

5 Conclusions and Outlook

This work developed a detailed physical model of a fixed-speed pumped hydro storage system with 100 MW nominal discharging power and a nominal charging and discharging duration of 10 hours. A hydraulic short circuit between the pump and the turbine was implemented to enable flexibility during charging operation. The effects of hydraulic short circuit operation were demonstrated both with instantaneous values and continuous operation over longer time periods. Instantaneous analysis of two different power levels (50 MW and 60 MW) highlights the complex component interactions during HSC operation. We recommend adopting the level of detail in this work for modelling HSC operating behaviour, specifically (i) an

accurate waterway network representation, (ii) hydraulic machine models with off-design performance relations, and (iii) consideration of the reservoir head variation. The system has a low efficiency in HSC operation. Therefore, strong economic incentives are needed for this mode of operation.

Several constant-power charging and discharging process simulations over time were combined to form $E-P$ and $\eta-P$ curves. This analysis shows that the operating range of the pumped storage system is not continuous. HSC enables part load charging between 39.8 MW and 107.1 MW. The HSC operating range is constrained by the turbine's maximum power and minimum flow limits. This should be considered when designing new pumped hydro systems. The available energy capacity drops significantly at HSC charging powers > 93 MW, a trend not apparent from the $\eta-P$ curve alone. This highlights the need to complement investigations of system efficiency with the energy capacity perspective.

The thermal analysis shows that HSC operation has only a small, negligible effect on the circulating water temperature in this specific pumped storage system. A brief investigation indicates that the temperature only becomes critical around flow regimes approaching zero flow towards the upper reservoir. Thermal considerations become relevant only when the pump and turbine powers are more evenly matched, and low flows towards the upper reservoirs are feasible operating points.

Overall, model-based investigations, such as the one conducted here, can reveal the operational flexibility potential of existing pumped hydro installations and systems in planning, thereby enabling these systems to contribute even more to modern decarbonised energy systems.

References

Alizadeh Bidgoli, Mohsen, Weijia Yang, and Ali Ahmadian (2020). "DFIM versus synchronous machine for variable speed pumped storage hydropower plants: A comparative evaluation of technical performance". In: *Renewable Energy* 159, pp. 72–86. DOI: 10.1016/j.renene.2020.05.163.

Beyers, Inga, Astrid Bensmann, and Richard Hanke-Rauschenbach (2023). "Ragone plots revisited: A review of methodology and application across energy storage technologies". In: *Journal of Energy Storage* 73, p. 109097. DOI: 10.1016/j.est.2023.109097.

Beyers, Inga, Lukas Krebeck, et al. (2025). *Comparing Operational Flexibility and Performance of Pumped Hydro Energy Storage Systems: A Quantitative Analysis of Technology Options*. Preprint available at SSRN. DOI: 10.2139/ssrn.5355183.

Engevik, Erlend L., Mostafa Valavi, and Arne Nysveen (2016). "Efficiency and loss calculations in design of converter-fed synchronous hydrogenerators". In: *2016 XXII International Conference on Electrical Machines (ICEM)*, pp. 1636–1642. DOI: 10.1109/ICELMACH.2016.7732743.

Fritzson, Peter et al. (2020). "The OpenModelica Integrated Environment for Modeling, Simulation, and Model-Based Development". en. In: *Modeling, Identification and Control: A Norwegian Research Bulletin* 41.4, pp. 241–295. ISSN: 0332-7353, 1890-1328. DOI: 10.4173/mic.2020.4.1.

Geiger, Christoph and Stefan Riedelbauch (2023). "Power Plant Transients including Hydraulic Short Circuit Operation Mode". In: *Energies* 16.11, p. 4492. DOI: 10.3390/en16114492.

Giese, Martin and Marcelo Magnoli (2019). *Recent Developments of Hydropower Machines for Pumped Storage Projects*. URL: https://cdn2.hubspot.net/hubfs/5069195/Voith-NA-Files/Voith-Developments-Hydro-PumpedStorage-2019.pdf.

Müller, Germar and Bernd Ponick (2009). *Grundlagen elektrischer Maschinen*. de. 9th ed. 1. Weinheim: John Wiley & Sons, Ltd. ISBN: 978-3-527-40524-4. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/352760782X.ch.

Nielsen, Torbjørn K (2015-09). "Simulation model for Francis and Reversible Pump Turbines". In: *International Journal of Fluid Machinery and Systems* 8.3, p. 14. DOI: 10.5293/IJFMS.202015.8.3.169.

Oeding, Dietrich and Bernd R. Oswald (2016). *Elektrische Kraftwerke und Netze*. de. 8th ed. Berlin, Heidelberg: Springer Vieweg. DOI: 10.1007/978-3-662-52703-0.

Pereira Jr., João Gomes et al. (2019-10). "Procedure for predicting part load resonance in Francis turbine hydropower units based on swirl number and local cavitation coefficient similitude". In: *Mechanical Systems and Signal Processing* 132, pp. 84–101. DOI: 10.1016/j.ymssp.2019.06.011.

Pérez-Díaz, Juan I. et al. (2015). "Trends and challenges in the operation of pumped-storage hydropower plants". In: *Renewable and Sustainable Energy Reviews* 44, pp. 767–784. DOI: 10.1016/j.rser.2015.01.029.

Pinto, Pedro Diogo et al. (2024). *Combining Technologies: Variable Speed And Hydraulic Short Circuit*. Lausanne.

Pritchard, Philip J. and John C. Leylegian (2011). *Fox and McDonald's Introduction to Fluid Mechanics*. 8th. John Wiley & Sons, Inc. ISBN: 978-0-470-54755-7.

Pyrhönen, Juha, Tapani Jokinen, and Valéria Hrabovcová (2013). "Losses and Heat Transfer". In: *Design of Rotating Electrical Machines*. Section: 9. John Wiley & Sons, Ltd, pp. 523–569. ISBN: 978-1-118-70159-1. DOI: 10.1002/9781118701591.ch9.

Reines, A F and B Svingen (2020). "Analysis and improvement of a mathematical turbine model". In: *Journal of Physics: Conference Series* 1608.1, p. 012001. DOI: 10.1088/1742-6596/1608/1/012001.

Ruppert, Leopold et al. (2017). "An analysis of different pumped storage schemes from a technological and economic perspective". In: *Energy* 141, pp. 368–379. DOI: 10.1016/j.energy.2017.09.057.

Tuzson, John (2000). *Centrifugal Pump Design*. Wiley. ISBN: 978-0-471-36100-8.

Vagoni, E. et al. (2024). *Technical White Paper, XFLEX HYDRO Project*. Tech. rep. Deliverable D10.3. EPFL. URL: https://www.xflexhydro.com/knowledge/technical-white-paper#downloads.

Valavi, Mostafa and Arne Nysveen (2018). "Variable-Speed Operation of Hydropower Plants: A Look at the Past, Present, and Future". In: *IEEE Industry Applications Magazine* 24.5, pp. 18–27. DOI: 10.1109/MIAS.2017.2740467.

Vytvytskyi, Liubomyr (2019). "Dynamics and model analysis of hydropower systems". PhD thesis. Porsgrunn: University of South-Eastern Norway. URL: http://hdl.handle.net/11250/2608105.

Introducing the NewLib Library and its application to multi-level, large-scale solar field models

Igor Belot[1], François Nepveu[2], Pierre Garcia[1], Etienne Letournel[1], Nathan Fournier[1], Teddy Chedid[1], Alexis Gonnelle[1], Guillaume Raigné[1], Pierre Delmas[1]

[1]Newheat, France, {igor.belot, pierre.garcia, etienne.letournel, nathan.fournier, teddy.chedid, alexis.gonnelle, guillaume.raigne, pierre.delmas}@newheat.fr

[2]CEA, CEA Tech Pays de la Loire, F-44340 Bouguenais, France, {francois.nepveu}@cea.fr

Abstract

Solar thermal technology is a promising solution for decarbonizing heat production in industrial applications and district heating networks. When combined with heat storage and advanced control strategies, it can cover a significant share of heat demands. However, designing and optimizing such systems is complex due to their dynamic behavior and the interplay of multiple physical phenomena. To better understand and design these systems, dynamic modeling tools are essential.

Modelica is particularly well-suited for this purpose. At Newheat, large scale solar thermal field models have been developed in the *Modelica* language using the *Dymola* environment. These models represent the thermal and hydraulic behavior of a solar thermal field at two different levels of complexity. Each is designed for different project phases, fast simulations for early-stage feasibility studies and slower but more detailed simulations for the engineering phase.

To assess the accuracy of both models, comparisons with measured data on an operational solar plant were performed. Results indicate that both models achieve high thermal accuracy, with errors of less than 4% in annual heat production. On the hydraulic side, the detailed model provides more precise results than the simplified one. The main drawback of this model being slow simulations in case of very complex solar field layouts.

Moving forward, these models will support various applications and enable scalable modeling of complex solar thermal systems, adapting to different project phases and requirements.

Keywords: solar thermal, ThermoFluidStream library, conceptual design, detailed engineering

1 Introduction

Newheat is a renewable and sustainable heat supplier, offering innovative and tailor-made solutions for major heat consumers worldwide: industrial sites and district heating networks. Its activity is to develop, design, build, finance and operate renewable heat plants, which can combine waste heat recovery, solar thermal, short and long-term thermal storage systems, optimized electrification solutions (industrial heat pumps, etc...) and, if necessary, combustion of renewable resources. The company currently has 6 sites in operation for a total of 40 MW of solar thermal fields installed.

Dynamic models of renewable heat components and plants are needed to estimate the performances and cost of the systems, and finally to assess the heat price defined in heat supply contracts over 15 to 25 years.

In the upstream phase of project development (conceptual design phase), models are used to explore all possible renewable heat production system configurations. The energy mix is not yet fixed, and an expected heat price must be assessed for several heat producer configurations, storage facilities and heat delivery points.

In the detailed engineering phase, dynamic models of complete systems are used for the detailed design of projects based on technical and economic criteria. At this stage, the hydraulic configuration of the installation and associated operating procedures (high-level control) are finalized, components are selected and specified, and location or implementation constraints (distances, soil, environment, etc.) are integrated.

NewLib models can also be used later in the construction phase to improve the Factory Acceptance Tests of the Programmable Logic Control when received, during commissioning to help tune the parameters of the

controllers, and finally in operation as digital twins to detect and diagnose faults.

In this paper, simple solar thermal field models used for conceptual design and more detailed models used for engineering needs are presented, compared, and validated against measured data from large scale solar thermal plants. After the presentation of the context and objectives in the present section, the rest of the paper is organized as follows:

- Section 2 presents the NewLib library, within the *Dymola* Environment, dedicated to the dynamic simulation of components and renewable heat production plants.
- Section 3 introduces the core of the NewLib library with special focuses on the media, pipe, heat exchanger and solar field models.
- Section 4 presents comparison results between NewLib models at 2 levels of details with measurements obtained from an operating solar plant.
- Section 5 summarizes the main messages of the present article and gives perspectives for the development roadmap regarding the NewLib library.

2 Presentation of the library

2.1 Overall presentation

Newheat has developed an in-house proprietary library called NewLib within the *Dymola* Environment, for internal purposes. It is not intended for public distribution or commercial release now. This *Modelica* Library:

- Is dedicated to the dynamic simulation of components and renewable heat production plants.
- Includes renewable heat components, control blocks and systems, tailored to the needs of its commercial projects.
- Provides multiple complexity levels of modeling for components, subsystems, materials, and fluids.
- Includes standardized interfaces enabling the interchangeability of component models of the same type in a system model.
- Enables annual simulations of renewable thermal plants within a few minutes (conceptual design phase).
- Uses the DASSL and CVODE solvers within the *Dymola* environment.

After benchmarking various thermofluidic libraries in the *Modelica* language – including the *MSL*, *IBPSA*, *Liquid Cooling*, *Thermal Power* and *ThermalSystems* – based on qualitative criteria as the library targeted industrial sector, the fluid port approach, the available media and the availability of models relevant to Newheat's need, as well as more quantitative KPIs like computation time and success rate of simulation in a parametric study, the *ThermoFluidStream* (*TFS*) library (Zimmer et al., 2022) was selected as a basis for the NewLib library, for the following reasons: Easy to use; Adequacy to Newheat's present and future needs; Robustness: based on an innovative numerical calculation scheme (Zimmer, 2020), designed to be robust even on complex thermo-fluid architectures; Fluid representation: partially compatible with the *Modelica* standard fluid library (MSL. Media).

The NewLib library is actively developed on *GitlLab*, therefore a methodology of continuous integration based on non-regression tests using the *Testing* library from Dassault (Hammond-Scott, 2022) is being followed.

2.2 Structure

The NewLib library is structured on packages as depicted in **Figure 1**. The core packages are the Components, Control and Media packages.

The Components package contains models of varying complexity representing the main components of a renewable thermal plant. These include producers such as solar fields, boilers and heat pumps; storage systems such as TTES (Tank Thermal Energy Storage), PTES (Pit Thermal Energy Storage), and BTES (Borehole Thermal Energy Storage); heat exchangers, including liquid/liquid plate heat exchangers and liquid/moist air finned-tube heat exchangers... Some of these models are largely inspired by other open-source libraries, such as *IBPSA* (Wetter, 2022). More specific models, including solar field models, will be described in more detail in this paper.

The Control package contains the models of the two levels of control commonly found in renewable heat plants. High-level control (HLC): 1. defines the conditions to enter and exit the different operating modes of the plants (and associated configuration of actuators) based on the different data acquisition made on site. For example, the operating modes of the solar field are standby, preheating, production, overheating prevention and overheating. In NewLib, the *Modelica_StateGraph2* library (Olsson, 2020) is used to develop the models of the HLC closely reflecting those implemented in operational plants; 2. defines the setpoints to be achieved by the Low-level control (LLC). LLC is responsible for applying and maintaining the setpoints defined by HLC. In practice, this typically involves regulation techniques, such as PID control. In NewLib, to ensure fast simulations, a model-based approach is used through an open loop control strategy. First-order filters are added to the command to eliminate nonlinear equations.

The Media package: Media models must be numerically robust, to avoid errors during dynamic simulations, as well as fast, accurate, and capable of computing the thermodynamic and transport fluid properties over a wide range of temperatures (and potentially pressures).

DOI 10.3384/ecp218363

Currently, only a monophasic and incompressible liquid model based on a polynomial formulation is available. This model has been extended to represent antifreeze and thermal oil heat transfer fluids from properties data as a function of the temperature to determine the corresponding polynomial coefficients.

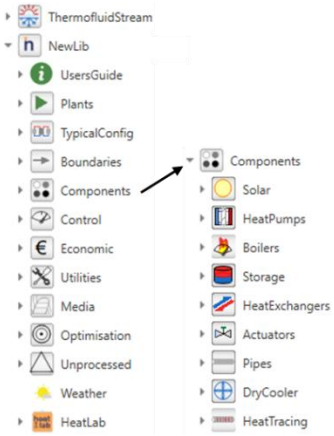


Figure 1. Packages of the NewLib library

2.3 The subsystem approach in NewLib

As part of Newheat strategic vision, NewLib library aims to provide a comprehensive framework – usable by Newheat's non-developer engineers – to all simulation requirements, from the conceptual design phase to the operational phase. Indeed, a Component-by-Component modeling approach raises several challenges, as shown in **Figure 2**:

- The model assembly may initially appear visually complex to new *Modelica* users.
- It requires configuring each component individually, often including low-priority or secondary parameters. This increases the risk of modeling errors due to incorrect settings.
- The handling of outputs becomes increasingly complex.

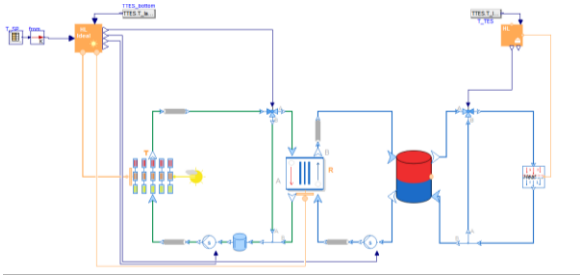


Figure 2. Component-by-Component modeling of a solar plant including a storage (TTES) loop

A first step in deploying NewLib across Newheat technical teams is to provide a user-friendly environment to it. That is the objective of the TypicalConfig package, which is an ongoing work. The package provides different templates for typical renewable heat production plants based on subsystem models (see **Figure 3**). A plant partition into 6 subsystems has been identified including Solar Field Subsystem, Storage Subsystem, Heat Consumer Subsystem, Heat Pump Subsystem, Heat Recovery Subsystem and HLC subsystem.

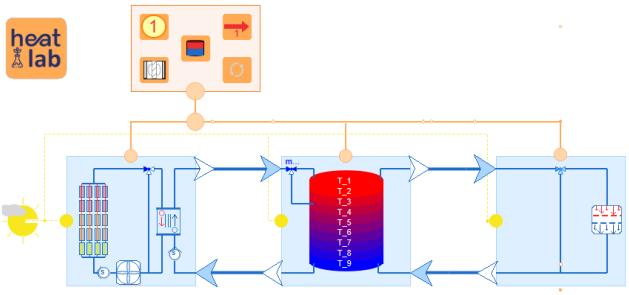


Figure 3. Subsystem modeling of Figure 2 Component-by-Component model of a solar plant

Each subsystem can model multiple hydraulic configurations that are commonly found in both operational plants and under development projects. For example, the solar field subsystem currently models two hydraulic configurations (see **Figure 4**):

- A direct configuration, where the solar field is directly connected to the thermal storage using a thermal oil as both the heat transfer fluid and the storage medium.
- An indirect configuration (used in **Figures 2** and **3**), used in all operational Newheat plants, where the thermal power from the solar field is transferred to the storage tank/user via a heat exchanger and a secondary circuit.

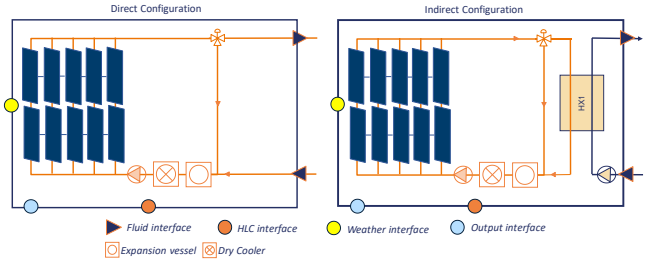


Figure 4. Hydraulic configurations of the solar field subsystem in NewLib

An error-proofing methodology is used to prevent the production of non-conforming parts in the conceptual design phase. The parameters of the subsystem components are classified into primary and secondary parameters. All primary parameters of the subsystem are defined by the user in a unique outer model called *Heatlab*. The other parameters are either pre-defined or calculated in an initial algorithm based on pre-conception rules.

The model outputs are managed using the *HideResult* annotation from *Modelica* language; KPIs (energy

balance, temperatures…) are calculated to provide a quick overview of the performances of each subsystem.

All main models of a subsystem are replaceable models, allowing for an easy switch from a conceptual design phase model, using simplified models, to a detailed engineering phase, using more complex dynamic models.

3 NewLib models description

This section provides a quick overview of the different models included in the NewLib library and usually used to represent large solar thermal plants. For each of these models both a conceptual design phase version and an engineering phase version are described.

3.1 Media models

To simulate low temperature solar thermal plants in the NewLib library, glycol water mixture (20% to 45% in volume), which is commonly used as a heat transfer fluid in those systems, is defined as a medium, with either constant properties or properties depending on the temperature of the fluid. It is considered incompressible. Polynomial equations are used to calculate the density (ρ), heat capacity at constant pressure (cp) and thermal conductivity (λ), an exponential form is used to model the dynamic viscosity (μ) of the medium:

$$\rho = a_{d,3}.T^3 + a_{d,2}.T^2 + a_{d,1}.T + a_{d,0} \quad (1)$$

$$cp = a_{c,2}.T^2 + a_{c,1}.T + a_{c,0} \quad (2)$$

$$\lambda = a_{l,2}.T^2 + a_{l,1}.T + a_{l,0} \quad (3)$$

$$\mu = A_e . exp(B_e/(T + C_e)) \quad (4)$$

With $a_{d,0}$, $a_{d,1}$, $a_{d,2}$, $a_{d,3}$, $a_{c,0}$, $a_{c,1}$, $a_{c,2}$, $a_{l,0}$, $a_{l,1}$, $a_{l,2}$, A_e, B_e and C_e being constant coefficients obtained by linear regression on fluid supplier documentation table data. If constant properties are chosen, properties of the fluid are calculated at a reference temperature defined as an input of the function. However, choosing this option can lead to significant errors in fluid pressures and/or temperatures in some cases. The main drawback of using variable properties media being larger CPU times. For specific studies during the engineering phase, hybrid media can be used. For instance, in hydraulic specification of the solar loop, constant cp, λ and ρ combined with variable μ media leads to fast results and good precision on pressure losses, while thermal accuracy is not required for this phase.

3.2 Pipes model

In the NewLib library, a single pipe model is used for both conceptual design and engineering phase. The energy balance in pipes is calculated considering a single fluid node where heat convection from inlet to outlet is calculated in a *heatTransfertElement* component from an enthalpy balance delayed by a *TransportDelayInit* component that accounts for the time delay from the inlet to the outlet of the pipe:

$$M.\frac{\partial h}{\partial t} = \dot{m}.(h_{in} - h_{out}) \quad (5)$$

$$\frac{\partial h(x,t)}{\partial t} + v(t).\frac{\partial h(x,t)}{\partial x} = 0 \quad (6)$$

Where h represents the specific enthalpy of the fluid, v represents the fluid velocity, M represents the inertial mass of the metal of the pipe and $\dot{m}$ represents the mass flow rate in the pipe. Heat losses to the external medium are calculated in a *PlugFlowHeatLoss* component, based on the plug flow model from *IBPSA* library. In this component, the outlet temperature is directly calculated from the solution of the heat equation considering only heat losses to the ambient:

$$T_{out} = T_{amb} + (T_{in} - T_{amb}).e^{-\frac{\tau}{R\rho c_p}} \quad (7)$$

With T_{amb} being the ambient temperature, τ being the residence time of the fluid in the pipe and R_{th} being the overall thermal resistance, calculated as:

$$R_{th} = \frac{2.\pi.L}{\frac{1}{r_1.h_{in}} + \frac{\ln(\frac{r_2}{r_1})}{\lambda_{pipe}} + \frac{\ln(\frac{r_3}{r_2})}{\lambda_{ins}} + \frac{1}{r_3.h_{out}}} \quad (8)$$

Where L is the pipe length, r_1, r_2 and r_3 are respectively the pipe inner radius and the insulation inner and outer radius, h_{in} and h_{out} are the inner and outer convection coefficients based on literature correlations (Yang et al., 2011, *The engineering Toolbox,* 2003) and λ_{pipe} and λ_{ins} are the thermal conductivity coefficients of the pipe and the insulation respectively defined from pipe suppliers data.

For the mass balance, constant mass flow rate is considered and the pressure losses in the pipe are defined as:

$$\Delta P = \Delta P_{reg} + \Delta P_{man} \quad (9)$$

Where ΔP_{reg} represents the regular pressure losses and is calculated considering the Darcy-Weisbach equation:

$$\Delta P_{reg} = f_D.\frac{L}{D_h}.\rho.\frac{V^2}{2} \quad (10)$$

Where f_D is the friction factor calculated from Cheng correlations (Cheng, 2008), suitable for all flow regimes.

And ΔP_{man} represents the manometric head, calculated as:

$$\Delta P_{man} = \rho.g.\Delta H \quad (11)$$

Where ΔH is the height difference between the inlet and outlet of the pipe and g the constant of gravity $g = 9.81\ m.s^{-2}$.

DOI 10.3384/ecp218363

3.3 Solar field models

The conceptual design phase solar field model in NewLib represents an equivalent solar collector consisting of a single thermal and hydraulic node. It calculates an equivalent pressure loss based on a rectangular solar field layout. The thermal part considers the heat exchange with the environment and enthalpy transport considering a delay due to the residence time of the fluid in the solar field as done for the pipe model.

The detailed engineering phase model, in contrast, explicitly represents manifolds and hydraulic singularities such as elbows and restrictions. Correlations of pressure losses in these singularities and in the regular parts are used. The thermal part is equivalent to the simplified model with a level of detail at the scale of the collector row rather than the overall solar field.

For both models, the energy balance in a single collector is calculated considering the overall thermal performance equation that is based on the quasi-dynamic equation from the ISO 9806 (ISO, 2017). This equations states that the power collected by the fluid (P_f) is the difference of the absorbed heat (P_{abs}) to the thermal losses (P_{loss}):

$$P_f = P_{abs} - P_{loss} \tag{12}$$

The power absorbed by the fluid is calculated as:

$$P_{abs} = S_{col}.\eta_{0,b}.fc.(K_b(\theta_l,\theta_t).G_b + K_d.G_d) \tag{13}$$

Where S_{col} is the surface area of the collector, $\eta_{0,b}$ is the optical efficiency of the collectors, $K_b(\theta_l,\theta_t)$ and K_d are the angle of incidence factors for direct and diffuse solar irradiations respectively, G_b and G_d are direct and diffuse solar irradiations respectively. fc is the so-called cleanliness factor that considers the reduction in the solar collector's performance caused by the accumulation of dirt, dust and other deposits (more details in §4).

The thermal losses are calculated as:

$$P_{loss} = S_{col}.\begin{pmatrix} c_1.(T_{mf} - T_{amb}) \\ +c_2.(T_{mf} - T_{amb})^2 \\ +c_3.u_{wind}.(T_{mf} - T_{amb}) \\ -c_4.(E_L - \sigma.T_{amb}^4) \\ +c_5.\frac{dT_{mf}}{dt} \\ +c_6.u_{wind}.(G_b + G_d) \end{pmatrix} \tag{14}$$

Where T_{mf} is the mean temperature of the fluid, T_{amb} is the ambient temperature surrounding the solar field, c_1, c_2, c_3, c_4, c_5, c_6 are respectively the heat loss coefficient, the coefficient of the temperature effect on the heat loss coefficient, the coefficient of the wind effect on the heat loss coefficient, coefficient of the sky temperature effect on the heat loss coefficient, the effective heat capacity and the coefficient of the wind effect on the optical efficiency, given by the collector model solar Keymark (Solar Keymark Network, n.d.), which is a certification of solar thermal collectors that provides a database of their coefficients in the ISO 9806 performance equation. Both models must be connected to the NewLib weather model to get weather input. In the NewLib library, the weather model uses weather data from Newheat onsite sensors as input to a *combiTimeTable* component of the *MSL*.

For the design phase model, the mass balance considers constant mass flow rate from inlet to the outlet of the solar field and pressure losses are calculated as:

$$\Delta P_{SF} = \Delta P_{collector} + \Delta P_{piping} \tag{15}$$

Where $\Delta P_{collector}$ and ΔP_{piping} are the pressure losses in the collectors and the pipes respectively.

In the collectors, the pressure losses are calculated as:

$$\Delta P_{collector} = \frac{\nu(T_0)}{\nu(T)}.a_0.\dot{V} + \frac{\rho(T)}{\rho(T_0)}.b_0.\dot{V}^2 \tag{16}$$

Where a_0 and b_0 are respectively the collector linear and quadratic pressure loss coefficients as given by its Solar Keymark, $\rho(T_0)$, $\nu(T_0)$ and $\rho(T)$, $\nu(T)$ are the density and dynamic viscosity at reference and actual temperature respectively and $\dot{V}$ is the volume flow rate in the collectors.

In the solar field pipes, the pressure losses are calculated as:

$$\Delta P_{piping} = \Delta P_{reg} + \Delta P_{sing} \tag{17}$$

Where ΔP_{reg} and ΔP_{sing} are the regular and singular pressure losses. The regular pressure losses are calculated as:

$$\Delta P_{reg} = \left(\frac{\dot{m}_{flow}}{\dot{m}_{flow,max}}\right)^2.\Delta P_{reg,max} \tag{18}$$

Where the maximal regular pressure loss in the solar field pipes is calculated from the Darcy-Weisbach equation using Cheng friction factor f_{D_i} at maximal fluid speed ($u_{max} = 2m/s$) in all the pipes constituting the most constraining path in the solar field:

$$\Delta P_{reg,max} = \sum f_{D_i}.L_i.\frac{\rho}{2}.\frac{u_{max}.|u_{max}|}{2.r_i} \tag{19}$$

$$f_{D_i} = \left(\frac{64}{\mathrm{Re}_i}\right)^a.\left(1.8\cdot\log\left(\frac{\mathrm{Re}_i}{6.8}\right)\right)^{2.(a-1).b} .\left(2.\log\left(3.7\cdot\frac{D_i}{rr}\right)\right)^{2.(a-1).(1-b)} \tag{20}$$

$$\mathrm{a} = \frac{1}{\left(1+\left(\frac{\mathrm{Re}_i}{2720}\right)^9\right)} \tag{21}$$

$$b = \frac{1}{\left(1 + \left(\frac{\mathrm{Re}_i . rr}{320 . r}\right)^2\right)} \quad (22)$$

Where rr is the relative roughness of the pipes and Re_i, L_i, D_i and r_i are the Reynolds number, length, nominal diameter and internal radius, respectively, of each pipe element constituting the most constraining path.

To calculate the most constraining path in the solar field, a rectangular solar field is considered, the pipes DN are determined in an initial algorithm by targeting a maximal velocity of 2m/s in each pipe.

In the engineering phase model, the hydraulic layout of the solar field is represented in much more detail. **Figure 5** shows the solar field of another large scale solar thermal plant currently under development in France, in implementation plan view and Dymola interface view of the equivalent model. The complex layout of this solar field is explained by irregular land availability. Pipes and manifolds are explicitly modeled, and regular and singular pressure losses are calculated.

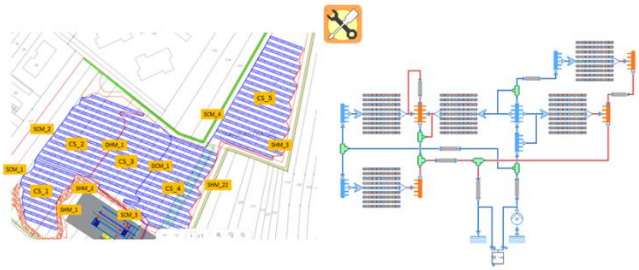


Figure 5. Implementation plan (left) and visualization of the equivalent model in Dymola using the NewLib library (right) of a complex solar field

Table 1 presents the main advantages and drawbacks of each model in the NewLib library.

Table 1. Design and engineering phase solar field models comparison in the NewLib library

	Design phase model	*Engineering phase model*
Advantages	Low CPU time Easy setup	Pressure loss precision
Drawbacks	Can not be used in engineering phase	Higher CPU time

3.4 Pump models

Design phase pump model in NewLib represents a flow generator. The mass flow rate is equal to the setpoint mass flow rate as:

$$\dot{m}_{flow} = \dot{m}_{flow,SP} \quad (23)$$

The electricity consumption of the flow generator is then estimated as follows:

$$P_{hyd} = \Delta P . \dot{V} \quad (24)$$

$$P_{elec} = \frac{P_{hyd}}{n_{tot}} \quad (25)$$

Where P_{hyd} and P_{elec} are respectively the hydraulic and electrical powers of the pump, n_{tot} is the total efficiency of the pump as defined by the user.

The engineering phase pump model in NewLib is based on the *MSL* model (*Modelica.Fluid.Machines.BaseClasses.PartialPump*). From the characteristic curve at nominal rotational speed $\dot{N}_{nom}$, a polynomial function Pc_{nom} is identified:

$$H_{m,nom} = Pc_{nom}(\dot{V}) \quad (26)$$

With $H_{m,nom}$ being the pump head at nominal rotational speed and $\dot{V}$ the volumetric flow rate. From this polynomial function, the pump head is deduced for every rotational speed $\dot{N}$ and volumetric flow rate $\dot{V}$ as:

$$H_m = \left(\frac{\dot{N}}{\dot{N}_{nom}}\right)^2 . Pc_{nom}(\dot{V} . \frac{\dot{N}_{nom}}{\dot{N}}) \quad (27)$$

Similarly, a polynomial function Pw_{nom} is calculated for the mechanical power of the pump from nominal values at different volume flow rates:

$$\dot{W}_{nom} = Pw_{nom}(\dot{V}) \quad (28)$$

The mechanical power of the pump ($\dot{W}$) is then calculated for different rotational speeds $\dot{N}$ as:

$$\dot{W} = \left(\frac{\dot{N}}{\dot{N}_{nom}}\right)^3 . Pw_{nom}(\dot{V} . \frac{\dot{N}_{nom}}{\dot{N}}) \quad (29)$$

The pressure loss (Δp) is directly deduced from the pressure head (H_m) as:

$$H_m = \frac{\Delta p}{\rho . g} \quad (30)$$

Considering constant fluid density ρ.

Then, electricity consumption is evaluated from a constant combined mechanical and electrical efficiency ε_{elec}:

$$\dot{P}_{elec} = \frac{\dot{W}}{\varepsilon_{elec}} \quad (31)$$

3.5 Heat exchanger models

For design phase simulations, an ε-NTU heat exchanger model is implemented in the NewLib library. In this model, the energy balance, constituting the heat exchange between the 2 sides of the heat exchanger is calculated from:

$$\dot{Q} = \varepsilon . \dot{Q}_{max} \quad (32)$$

DOI 10.3384/ecp218363

$$\dot{Q}_{max} = C_{min.}\Delta T_{max} \quad (33)$$

$$\Delta T_{max} = T_{h,in} - T_{c,in} \quad (34)$$

Where $\dot{Q}$ and $\dot{Q}_{max}$ are respectively the actual and maximal exchanged heat, C_{min} is the minimal capacity flow rate ($c_p \cdot \dot{m}_{flow}$) and ΔT_{max} is the maximal temperature difference in the heat exchanger. ε is the counter-flow configuration heat exchanger efficiency defined as:

$$\varepsilon = \begin{cases} \dfrac{1 - exp[-NTU \cdot (1-R)]}{1 - R \cdot exp[-NTU \cdot (1-R)]}, if\ R < 1 \\ \dfrac{NTU}{1 + NTU}, if\ R = 1 \end{cases} \quad (35)$$

With NTU (number of transfer units) defined as:

$$NTU = \frac{U.A}{C_{min}} \quad (36)$$

With U being the overall heat transfer coefficient and A the equivalent surface area of the heat exchanger. R is the heat flow rate ratio, defined as:

$$R = \frac{C_{hot}}{C_{cold}} \quad (37)$$

Where C_{cold} and C_{hot} are the capacity flow rates on the cold and hot side of the exchanger respectively. For the mass balance, constant flow rate is imposed on both sides of the heat exchanger, and pressure losses are calculated as follows:

$$p_{loss} = k_l.m_{flow} + k_q.m_{flow} \quad (38)$$

Where k_l and k_q are constant coefficients defined as inputs or calculated from nominal operating point.

The engineering phase heat exchanger model in NewLib is also based on the ε-NTU method. Additional parameters are proposed to the user to define a specific heat exchanger model with the exact number of plates and other geometrical design parameters such as corrugation angle, joint thickness or expansion factor, to calculate the number of heat transfer-effective plates.

4 Validation and Application

4.1 Use case plants description

In 2019, Newheat, the city of Narbonne in the south of France and the operator of its district heating network agreed to develop a large-scale solar thermal plant to maximize the proportion of renewable energy on the network as it develops. The objective was to replace, as much as possible, the use of gas boilers with the production of solar energy, covering most of the network's needs during the summer period, from April to October. Newheat oversaw the entire project and bears full responsibility for its various stages (design, development, construction, operation for a 25-year period). The project has received financial support from ADEME's *Fonds de Chaleur* and the Occitanie Region (France).

The solar plant project called NARBOSOL commissioned in 2021 consists of a solar field, a thermal storage tank (hot water), to offset periods of solar heat production and consumption by the heating network, and a technical room containing the programmable logic controller (PLC) and all the hydraulic equipment (pumps, valves, heat exchangers, measuring instruments, etc.). Main characteristics of NARBOSOL plant are presented in **Table 2**.

Table 2. NARBOSOL solar thermal plant main parameters.

Parameter	*Value*
Location	43.183°N, 2.967°W
Solar field aperture area (m²)	3200
Solar field peak power (kW)	2700
Solar collectors tilt (°)	35°
Storage volume (m^3)	1000
Solar heat transfer fluid	Glycol 30%
Storage fluid	Water
Annual production (MWh)	2200

4.2 Solar field Modeling

The NARBOSOL's solar field consists of 202 Savo 15 SG solar collectors (Savosolar Oyj, 2020) divided into two subsolar fields. The first subsolar field contains 12 rows, each with a range of 8 to 12 serial collectors, while the second subsolar field contains 7 rows, each with a range of 7 to 10 serial collectors (see **Figure 6**). Both subsolar fields are connected to the hydraulic skid from a cold and a hot manifold to feed all the rows with the heat transfer fluid. Balancing valves are used to equally distribute the system's total flow in each row as a function of serial collector number.



Figure 6. Aerial view of NARBOSOL solar thermal plant in Narbonne (France)

Newheat uses remotely accessible supervision technologies to monitor the plant's status and modify its parameters to maintain optimal operating conditions. All data transmitted from the sensors (temperature, pressure,

mass flow, electric consumption…) installed in NARBOSOL are recorded with a time step of 1 min in a proprietary data base. In this paper, data recorded in 2024 at NARBOSOL is used to compare the performance of both approaches in modeling the solar field and other components of the solar loop in NewLib, dedicated to the conceptual design phase and the detailed engineering phase.

Evaluation of the cleanliness factor of NARBOSOL

The cleanliness factor fc (see **Equation (13)**) of a solar field model is a parameter that affects the optical efficiency of the collectors by reducing the incident solar power.

To assess the annual mean cleanliness factor value of NARBOSOL plant, an assembly model was developed. It includes the conceptual design phase model of the solar field and other physical components such as pumps and pipes from the NewLib library. One-minute frequency measurements of weather data (Global Tilted Irradiation - GTI, wind speed, and ambient temperature), volume flow rate, and inlet temperature are used as inputs of the model. The model calculates the 8760 hourly thermal power delivered to the fluid by the solar field in 2024. **Figure 7** shows the graphical representation of this assembly model in the *Dymola* environment.

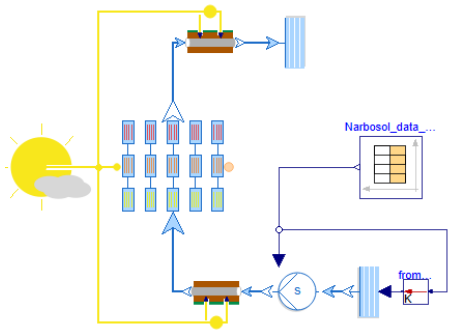


Figure 7. Graphic view of the design phase model of NARBOSOL's solar field in NewLib

A parametric study is then carried out using this assembly model, in which the root means square error (RMSE) is computed by comparing the model results with the measured thermal power as a function of the cleanliness factor.

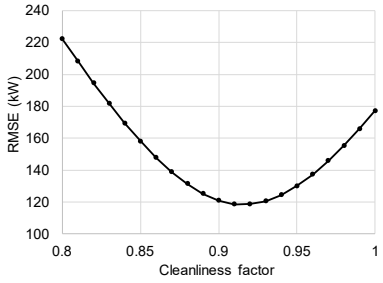


Figure 8. Parametric study for the evaluation of the cleanliness factor

Figure 8 presents the results of this parametric study. The graph clearly shows the value of the cleanliness factor of 92 % that minimizes the RMSE.

Solar field models evaluation

In this section, the performance of both NewLib solar field models (§3.3) is evaluated. To do so, 2 assembly models of the NARBOSOL plant, similarly to **Figure 7**, are developed using NewLib library's conceptual design phase models and engineering phase models. Simulations are carried out using these 2 assembly models on 1-minute step data from September 2024 at NARBOSOL.

The results are compared with the measurements from September 2024 in terms of monthly cumulative energy (**Figure 9**), thermal power (**Figure 10**), and hydraulic characteristic curve of the solar field (**Figure 11**). For both models, the value of the cleanliness factor of 92 % determined in the previous section is used. The 30% glycol water medium with variable properties (see §3.1) is used in both simulations.

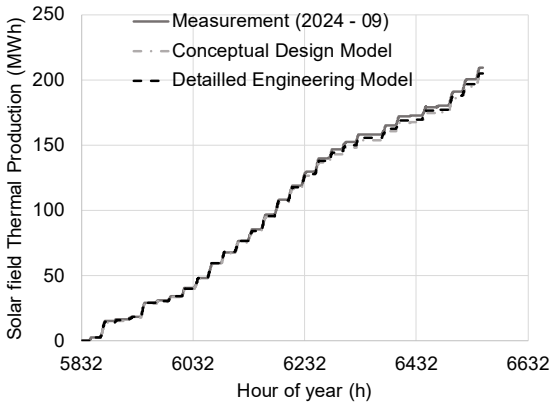


Figure 9. Cumulative Energy provided by the solar field in September 2024

Figure 9 shows a very good agreement between both models and measurement. In September 2024, the energy delivered to the fluid by the NARBOSOL solar field was measured at 209.4 MWh, while the calculated values were 202.9 MWh (3% error) and 205.0 MWh (2% error) for the conceptual design model and the detailed engineering model, respectively. These results suggest that the level of detail at the overall solar field scale used in the conceptual design model is sufficient to accurately represent the thermal performances of large-scale solar fields. Better agreement between measured and simulated data in the first half of the month can be explained by sunnier days during this period. The model being more accurate when less irradiation variations.

Figure 10 further confirms this result in terms of thermal power on two chosen cloudy and sunny days. Finaly, this graph strengthens the observation that the conceptual design model provides a very good estimation of the solar field's thermal performance despite its simplified level of detail. The correlation coefficient over the whole month

DOI 10.3384/ecp218363

on solar field thermal power is 0.9904 for the conceptual phase model compared to 0.9908 for the more detailed one.

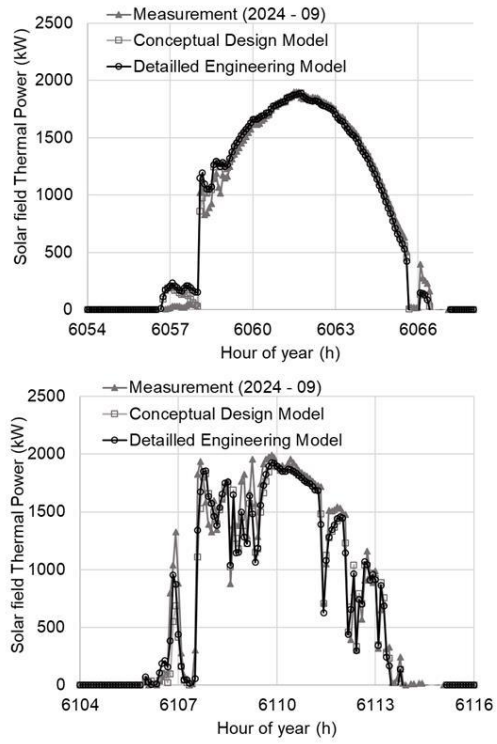


Figure 10. Solar thermal power result comparison between both models and measured data for a sunny day (top) and a cloudy day (bottom)

However, results on the solar field pressure drop (see **Figure 11**) show that the conceptual design model is unable to predict the hydraulic behavior of the solar field. In the case of NARBOSOL's model, this result is due to the estimation of the most constraining path in the solar field, based on the assumption of a rectangular solar field, which is far from the real configuration in this case. In the detailed engineering model, the hydraulic layout of the solar field is represented in much more detail, including pipes, manifolds and a refined representation of each sub-solar field with the actual number of collectors in series for each row. As shown in **Figure 11**, this level of detail enables highly accurate results on pressure loss prediction.

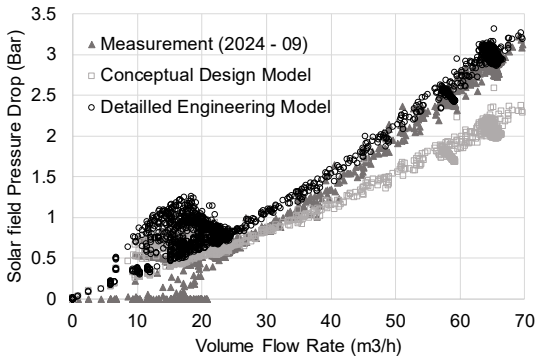


Figure 11. Hydraulic characteristic curve of the solar field for both model representations and measured data

Solar field complete loop modeling

In this final section, NewLib's detailed engineering models are integrated within a larger Component-by-Component assembly model of the solar field loop, which includes:

- A centrifugal pump model (see §3.4), a heat exchanger model (see §3.5), a singular pressure drop model of the hydraulic skid, and other standard models of hydraulic components such as a 3-ways valve and an expansion vessel.
- The HLC model which calculates the operating modes of the solar field loop (§2.2).
- LLC Models of the solar field, valve, pump and heat exchanger which determines the commands to be sent to reach the HLC model setpoints.

The centrifugal pump model is based on the technical specifications of the ETL 065-065-160 pump from KSB. It defines the pump head and the shaft power as a function of rotation speed and volume flow rate. Then a constant combined mechanical and electrical efficiency of 0.95 x 0.95 x 0.95 is assumed, which reflects losses in the pump/electric motor coupling, frequency inverter, and electric motor. The heat exchanger model is based on a nominal operating point defined by inlet/outlet temperatures of 91.2 °C / 67.1 °C on the primary side and 63 °C / 88.5 °C on the secondary side, with an exchanged thermal power of 1.9 MW. **Figure 12** presents a visualization of the described assembly model in the *Dymola* environment.

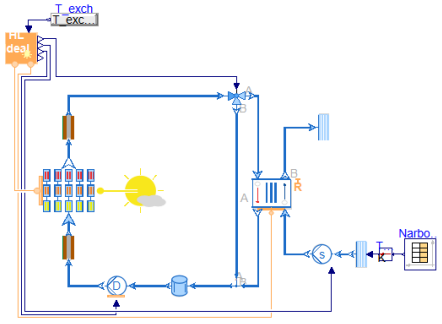


Figure 12. Model of NARBOSOL's solar field loop in NewLib as seen in Dymola

A simulation is launched on 1-minute frequency weather and inlet temperature of the secondary loop data in September 2024 at NARBOSOL, as inputs of the model. Results of heat power comparison are presented in **Figure 13** for a cloudy day. A very good agreement between simulation and measurement data is observed. The correlation coefficient is 0.9769 over the whole month.

Table 3 shows measured and simulated September 2024 energy balances in the solar loop. Regarding thermal energy, the deviation is limited, with a 2% difference between calculated and measured values.

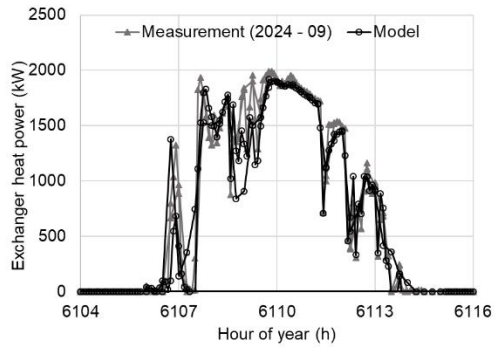


Figure 13. Heat supplied to the tank result comparison between both models and measured data

The centrifugal pump electrical consumption is also well estimated, with a deviation of 11 % compared to the measurements. This higher deviation in the pump electrical consumption can be partly attributed to differences between the actual equipment and its technical specifications.

Table 3: Energy balance comparison for September 2024 between both models and measured data

	Measurements	Model
Thermal solar Energy	209.4 MWh	211.8 MWh
Energy provided to storage	207.4 MWh	212.5 MWh
Electric pump consumption	1.8 MWhe	2.0 MWhe

5 Conclusion

In this paper, the 2 different modeling approaches available in the NewLib library are presented. Comparisons of these 2 modeling approaches with measured data on an operating solar plant have been carried out.

On the one hand, the conceptual design phase model of the solar field in NewLib provides robust thermal results but cannot be used for hydraulic conception purpose during the engineering phase of projects due to its lack of accuracy in pressure losses prediction. On the other hand, the detailed solar field model offers significantly improved accuracy in pressure loss calculations and combined with medium representation can lead to acceptable simulation time. The NARBOSOL case serves as a prime example of these advantages.

Future perspectives of this work for each phase of Newheat projects are:

- Conceptual design phase: Finalization of the template-based approach to streamline model development.
- Engineering phase: Development of detailed models for key components, including heat exchangers and valves, to enhance system accuracy.
- Operational phase: Refinement of detailed models to facilitate the implementation of a digital twin approach, ultimately optimizing maintenance operations, such as fault detection like leak identification, and improving overall system performance.

These advancements pave the way for more accurate simulations and the potential for real-time optimization in future applications.

Acknowledgements

This work has been supported by ADEME and Région Nouvelle Aquitaine (France). The authors would also like to thank *TFS* developers for their work.

References

Cheng, Nian-Sheng, 2008. Formulas for Friction Factor in Transitional Regimes, Journal of Hydraulic Engineering. 134 (9): 1357–1362. doi:10.1061/(asce)0733-9429(2008)134:9(1357). hdl:10220/7647. ISSN 0733-9429.

Hammond-Scott, H., 2022. Testing Library: A look Inside. Claytex, URL https://www.claytex.com/tech-blog/testing-library-a-look-inside/ (visited on 2025-4-28).

International Organization for Standardization, 2017. ISO 9806:2017: Solar Energy — Solar Thermal Collectors — Test Methods.

Olsson, H. 2020, URL https://github.com/HansOlsson/Modelica_StateGraph2 (visited on 2025-4-28).

Savosolar Oyj, 2020. Savo 15 SG-M collector – Technical datasheet. https://meriauragroup.com/wp-content/uploads/archive/savosolar.com/wp-content/uploads/Savo-15-SG-M-technical-datasheet-2020-03-26-EN.pdf (visited on 2025-04-29).

Solar Keymark Network. (n.d.). Solar Keymark Certification Scheme. Retrieved from https://solarheating.eu/solar-keymark/.

The Engineering ToolBox, 2003. Understanding Convective Heat Transfer: Coefficients, Formulas & Examples. [online]: https://www.engineeringtoolbox.com/convective-heat-transfer-d_430.html (visited on 2023-08-29).

Wetter, M., 2022. URL https://github.com/ibpsa/modelica-ibpsa/wiki (visited on 2025-4-28).

Yang, J-K et al., 2011. Heat Transfer Coefficient in Flow Convection of Pipe-Cooling System in Massive Concrete. Journal of Advanced Concrete Technology 9, no 1 : 103-104.

Zimmer, D., 2020. Robust object-oriented formulation of directed thermofluid stream networks. Mathematical and Computer Modelling of Dynamical Systems 26, 204–233. https://doi.org/10.1080/13873954.2020.1757726.

Zimmer, D., Meißner, M., Weber, N., 2022. The DLR ThermoFluid Stream Library. Electronics 11, 3790. https://doi.org/10.3390/electronics11223790.

Further Application of Modelica-Based Nuclear Power System Simulation: A Review of Home-Tasks in Different Scenarios Driven by Model and Data

Ao Zhang [1] Xiang Wang [1]

[1] College of Nuclear Science and Technology, Harbin Engineering University, China, {zhangao, xiang.wang}@hrbeu.edu.cn

Abstract

System simulation is inherently an applied technology, driven by specific application demands. The Modelica language enables equation-based, system-level, multi-domain, and visual modeling, facilitating researchers conducting system simulation studies of nuclear power systems. This paper introduces the custom-built AESE library on the OpenModelica platform. Examples in AESE are provided to illustrate the development and simulation of system models for conventional pressurized water reactors (PWRs) and high-temperature gas-cooled reactors (HTGRs). The paper also describes tasks completed under different application scenarios, including hardware-in-the-loop simulation, multi-objective optimization, system identification, and rapid optimization, using model-driven and data-driven approaches with the AESE library. The further application of simulation models and data has significant practical and engineering value. This paper serves as a valuable reference for the intelligent application of energy system models in the context of advanced engineering challenges.

Keywords: Modelica, Nuclear Power System, Model-driven, Data-driven

1 Introduction

System-level simulation provides a framework for understanding and analyzing the behavior of complex systems, enabling enhanced design and optimization. Multi-domain system-level modeling has gained significant popularity in seamlessly integrating physical models from various domains using a unified modeling language. This approach helps to overcome interface issues between different domain-specific software and enables the modeling of comprehensive and complex real-world systems (Mattsson et al. 1998).

Modelica, as an example of a multi-domain system modeling language, employs mathematical equations to uniformly describe physical processes across different domains. It offers visual modeling capabilities to represent the topology of a system and enables the solution of steady-state and transient simulations by solving algebraic and differential equations. The Modelica language has been widely used in the aerospace, shipbuilding, automotive, energy, and other fields (Briese et al. 2020; Soriano et al. 2016; Andreasson et al. 2013).

Nuclear power systems (NPSs) are inherently complex and face increasing demands for shorter development cycles, higher safety standards, and smarter operation. While traditional offline Modelica-based simulations have successfully supported reactor design and thermal-hydraulic analysis (Casella and Leva 2005; Cammi et al. 2011), the full potential of system simulation extends beyond this. Driven by the need for more intelligent system management and accelerated development, system simulation is evolving to support multiple practical scenarios that leverage both models and data. This work specifically focuses on the expanded application of Modelica-based NPS simulation across diverse technical dimensions:

a) Hardware-in-the-loop simulation (HILS), which enables integrated control verification by coupling physical hardware with digital models. For complex NPSs, the pressure to shorten development cycles and meet stringent safety requirements necessitates comprehensive control system testing to avoid costly failures. These demands have driven recent advancements in HILS (Mihalič et al., 2022). Although Modelica is widely used in offline NPS simulations, its application in system-level HILS research for nuclear systems remains limited.

b) Multi-objective optimization (MOO), which facilitates system-level design exploration by resolving trade-offs among competing objectives. In large-scale systems like NPSs, single-objective optimization often fails to represent the comprehensive requirements of engineering applications (Zhang and Wang 2024). Therefore, greater attention is given to multi-objective optimization problems that incorporate complex constraints. These objectives are often interdependent and cannot be optimized simultaneously, requiring coordinated strategies to identify Pareto-optimal solutions under varying conditions.

c) System identification, which leverages operational data for rapid prediction and data-driven modeling, supporting fast evaluation and intelligent optimization. A

significant amount of data generated during nuclear power operation remains underutilized. However, emerging data-driven multi-objective (DDMO) optimization frameworks aim to bridge this gap. By coupling system identification algorithms with MOO, these frameworks enable efficient exploration of design spaces, enhance productivity, and improve predictive capabilities for future operational states.

These application scenarios underscore the diverse and critical roles that system simulation can play in modern NPS development. As depicted schematically in Figure 1, they encompass physical integration, algorithmic coordination, and data utilization, collectively forming a comprehensive modeling–application framework. This paper builds upon this framework by first introducing the general modeling approach using the AESE library (Section 2), then presenting these three representative applications in detail (Section 3), and finally concluding with a summary and outlook (Section 4).

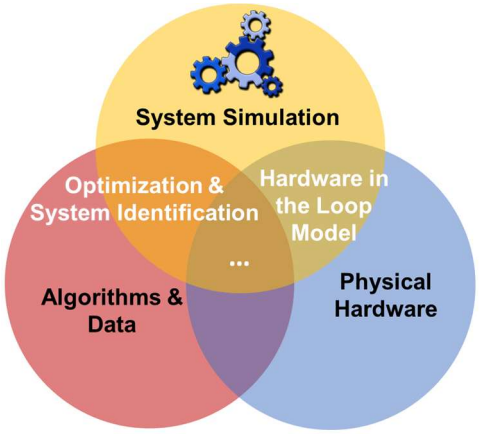


Figure 1. The further application directions of system simulation discussed in this paper

2 System Simulation

2.1 Framework

A custom-built Advanced Energy Systems Engineering (AESE) library (Zhang and Wang 2024) in OpenModelica (Fritzson et al. 2020) was developed in this study to describe various advanced energy systems. Figure 2 depicts the AESE architecture and the hierarchical structure of its He-Xe CBC system sub-library. Designed for modularity and openness, the library separates component models from working fluid models, allowing easy adaptation to future fluid models. Modelica facilitates this by supporting a high-level component library framework, defining standardized interfaces, and using inheritance or substitution for detailed component implementations. The following briefly outlines the modeling and simulation of two typical nuclear energy systems: the pressurized water reactor (PWR) and the high-temperature gas-cooled reactor (HTGR).

2.2 Pressurized Water Reactor (PWR)

The PWR-NPS consists of a nuclear reactor and two energy conversion loops (Figure 3). This study created a direct propulsion PWR-NPS model using Modelica, based on the "NS Mutsu" nuclear merchant ship (Peng 2009). The model, focused on a pressurized water reactor, describes fluid flow, heat transfer, phase changes, and power output. It was visually built (Figure 4) using adapted components from the ThermoPower (Casella and Leva 2024) and AESE libraries, including a reactor with a point kinetics model, a steam generator (SG) with drum and one-dimensional flow parts, condensers, turbines, pumps, and control units. Steady-state tests (Table 1) and dynamic analysis (Figures 5-7) verified the model's accuracy (Zhang et al. 2024). These results demonstrate that the Modelica-based model is capable of simulating and validating both steady-state and dynamic operations of conventional PWRs in the nuclear field.

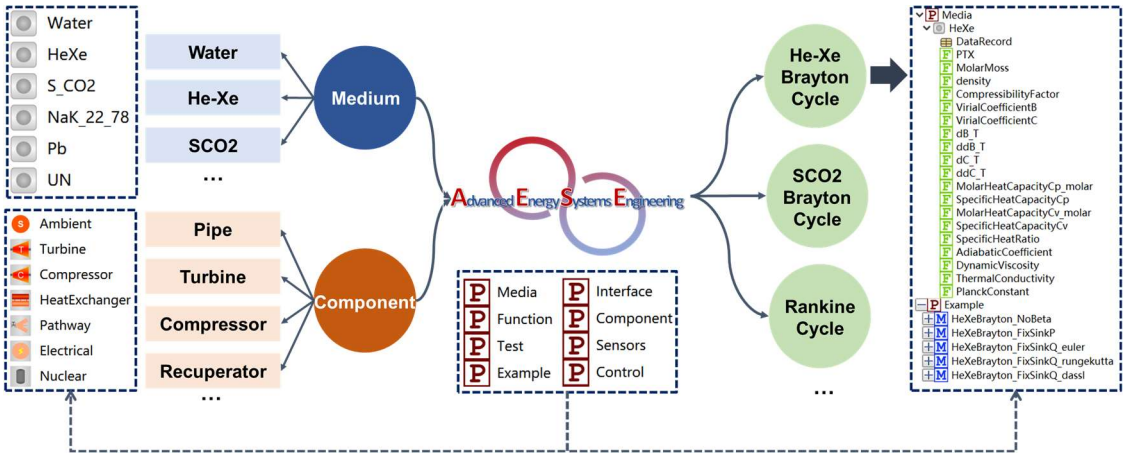


Figure 2. Schematic of the advanced energy systems engineering (AESE) library (Zhang and Wang 2024).

DOI 10.3384/ecp218373

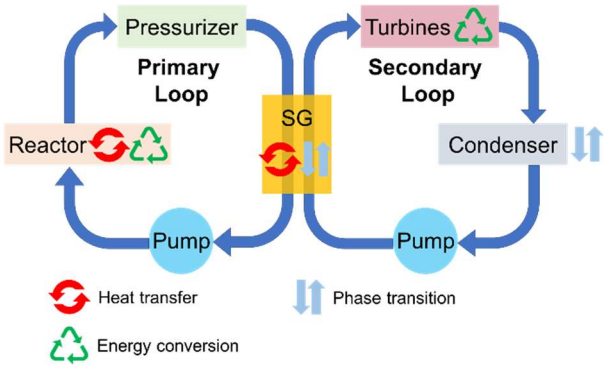


Figure 3. Structure of PWR-NPS (Zhang et al. 2024).

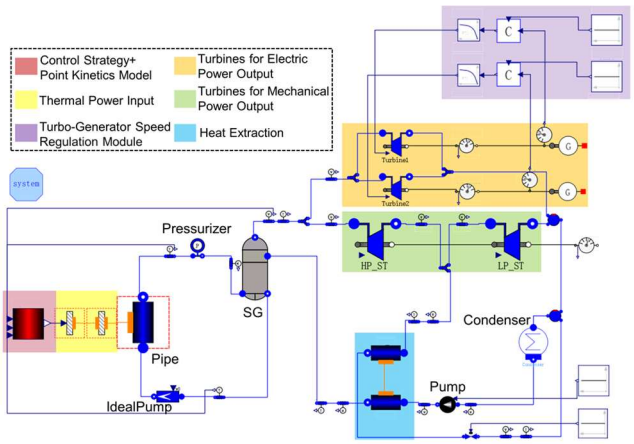


Figure 4. Simulation model of the PWR-NPS (Zhang et al. 2024).

Table 1. Steady-state simulation results.

Parameter	Reference	Result	Error
Reactor thermal power (MW)	36	35.81	0.52%
Temperature of SG primary side inlet (K)	558.15	555.18	0.53%
Temperature of SG primary side outlet (K)	544.15	544.82	0.12%
Pressure of SG secondary side (MPa)	3.9	3.72	4.61%
Feedwater temperature of SG secondary side (K)	433.15	431.28	0.43%
Steam temperature of SG secondary side (K)	524.15	519.19	0.95%
Steam output of SG (kg/s)	17	17.31	1.82%
Mechanical power output of steam turbine (MW)	7.35	7.38	0.41%

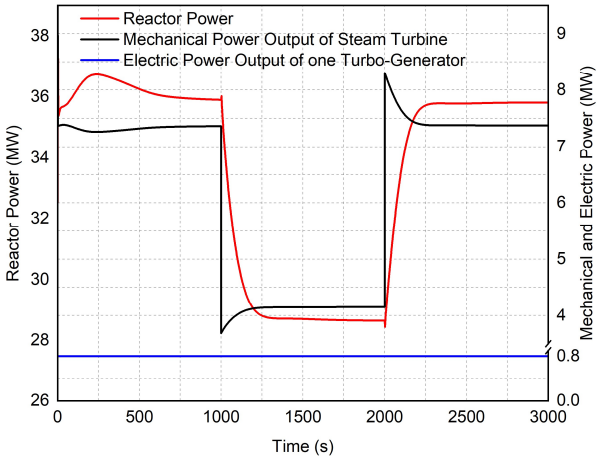


Figure 5. Transient simulation results of the nuclear power system (Zhang et al. 2024).

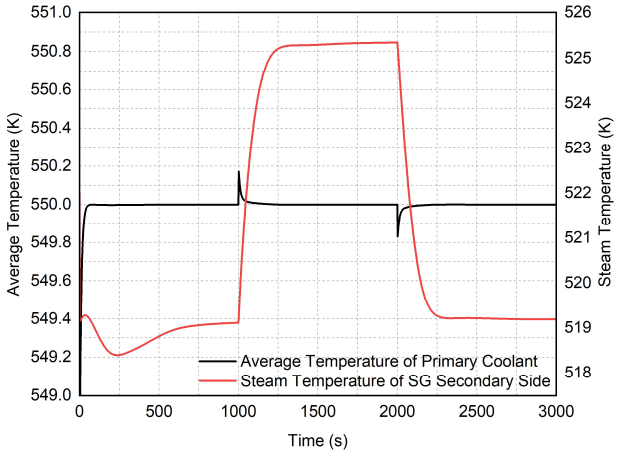


Figure 6. Transient heat transfer process on the secondary side of the SG (Zhang et al. 2024).

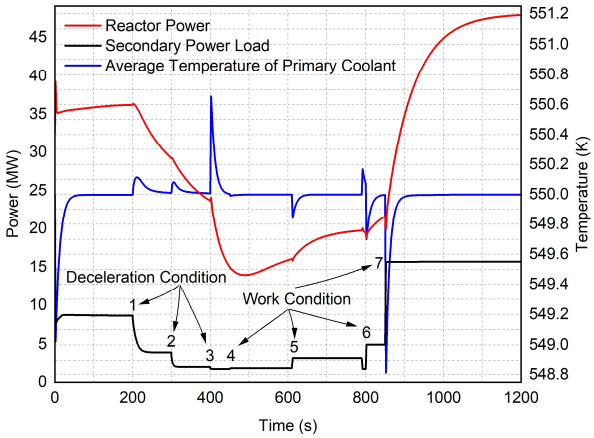
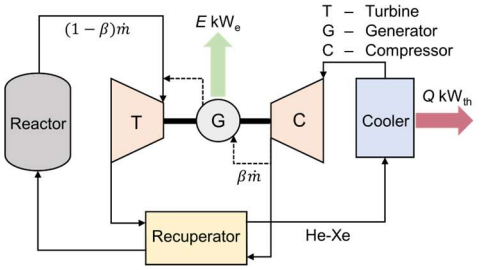


Figure 7. Transient simulation results of the NPS (Zhang et al. 2024).

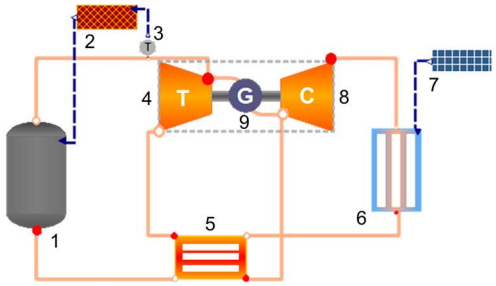
2.3 High-Temperature Gas-Cooled Reactor (HTGR)

The HTGR enables efficient coupling with the Brayton cycle via its high outlet temperature, while Brayton cycle research focuses on system configuration and gas property effects. The AESE library also includes a He-Xe property

library and supports closed-Brayton-cycle (CBC) systems (Figure 8a), offering new tools for simulating He-Xe CBC systems (Figure 8b). The presence of Xe in He causes deviations from ideal gas behavior, addressed by Tournier et al. (2006) using a binary noble gas calculation theory. Figure 9 displays the core physical properties (left) and their Modelica-based implementation in the AESE library (right). We validated He-Xe properties against literature (El-Genk et al. 2007), performed steady-state (Figures 10-11) and transient analyses for the He-Xe CBC-NPS, and assessed system performance across design parameters (Figure 12), guiding future optimization and design (Zhang and Wang 2024). These results indicate that the Modelica-based model supports pioneering studies of advanced reactor concepts in the nuclear field.



(a) Schematic diagram of He-Xe CBC system



(b) Visual model of the He-Xe CBC system in AESE

Figure 8. Modelica-based model of the He-Xe CBC system (Zhang and Wang 2024)
1- Reactor; 2-Temperature control module;
3-Temperature sensor;
4-Turbine; 5-Recuperator; 6-Cooler;
7-Cooling control module; 8-Compressor; 9-Generator.

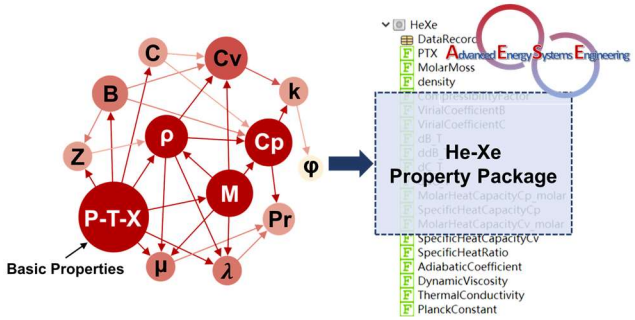


Figure 9. He-Xe property calculation process and property package in AESE (Zhang and Wang 2024).

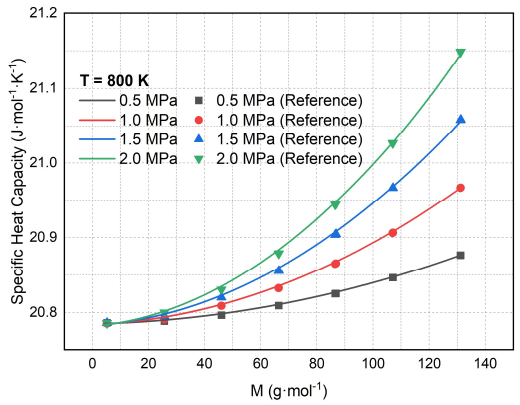


Figure 10. Verification of the specific heat capacity (Zhang and Wang 2024).

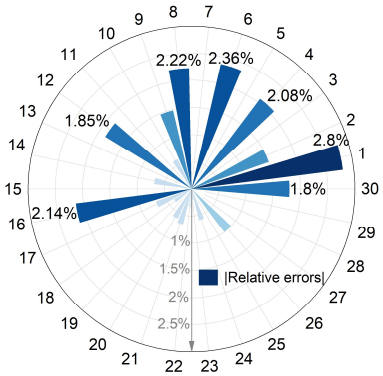


Figure 11. Relative error of key system parameter calculations compared to El-Genk et al. (2007).

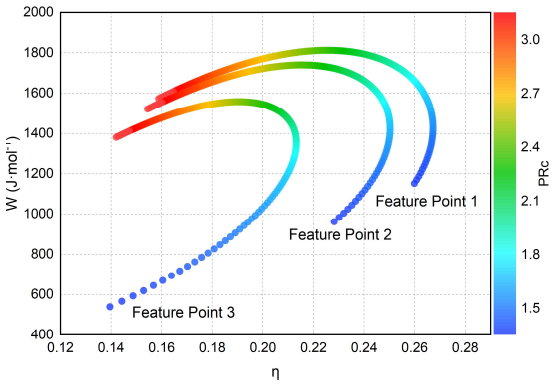


Figure 12. Efficiency and specific work of the system (Zhang and Wang 2024).

3 Further Applications in Different Scenarios

After validation, NPS models hold significant potential for valuable applications in equipment design and operation. For example, system models can drive real hardware to enable rapid control loop validation in HILS; support optimization algorithms for MOO design in large-scale systems; and utilize simulation data to drive intelligent algorithms for system identification and rapid optimization. Models can be further expanded and applied across various scenarios and dimensions.

DOI
10.3384/ecp218373

3.1 Hardware in the Loop Simulation (HILS)

HILS links a physical object with a computer-based simulation model for testing, accurately capturing the controller's dynamic, static, and nonlinear characteristics, making it a highly realistic simulation technology.

To study the variable operating condition responses of nuclear-powered engineering ships (NPES), we initially developed a desktop-level HILS model framework (Zhang et al. 2024). The HILS platform framework (Figure 13) integrates software, hardware, and a communication chain (Zhang et al. 2024), with the system model executed in OpenModelica and real-time signal transmission handled via an Arduino module. This configuration enables real-time coupling between the virtual NPES system and physical controllers for realistic validation and testing scenarios.

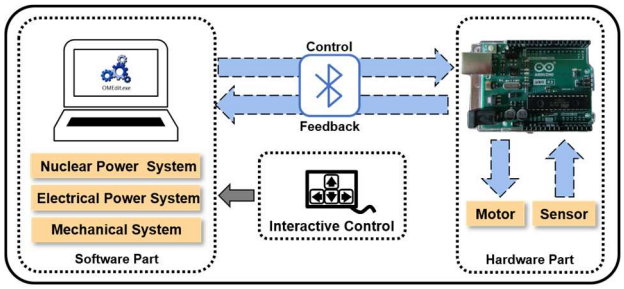


Figure 13. Framework of the HILS platform for an NPES (Zhang et al. 2024).

Both the offline model and the HILS software-level model are shown in Figure 14. The offline model uses the "Transient Signal Module," while the HILS model employs the "HILS Signal Module," enabling hardware communication via an Arduino module (COM3 port, signal pins) for control signals and sensor data. Real-time user interaction is supported through keyboard inputs, processed by signal control logic. The offline model is primarily used for preliminary system validation, whereas the HILS model allows real-time interaction with physical hardware for controller testing and signal feedback.

Figure 15 shows the HILS signal filtering process for the NPES, averaging the signal every second. The PID control was critical during the NPES's three-stage deceleration, as demonstrated by the velocity curve in Figure 16.

These results confirm the feasibility and effectiveness of the Modelica-based HILS platform for simulating and analyzing the dynamic responses of NPES under variable operating conditions. The key advantage of Modelica lies in its ability to represent multi-domain system models and integrate them with real hardware components, enabling the construction of more comprehensive and organically coupled HILS frameworks.

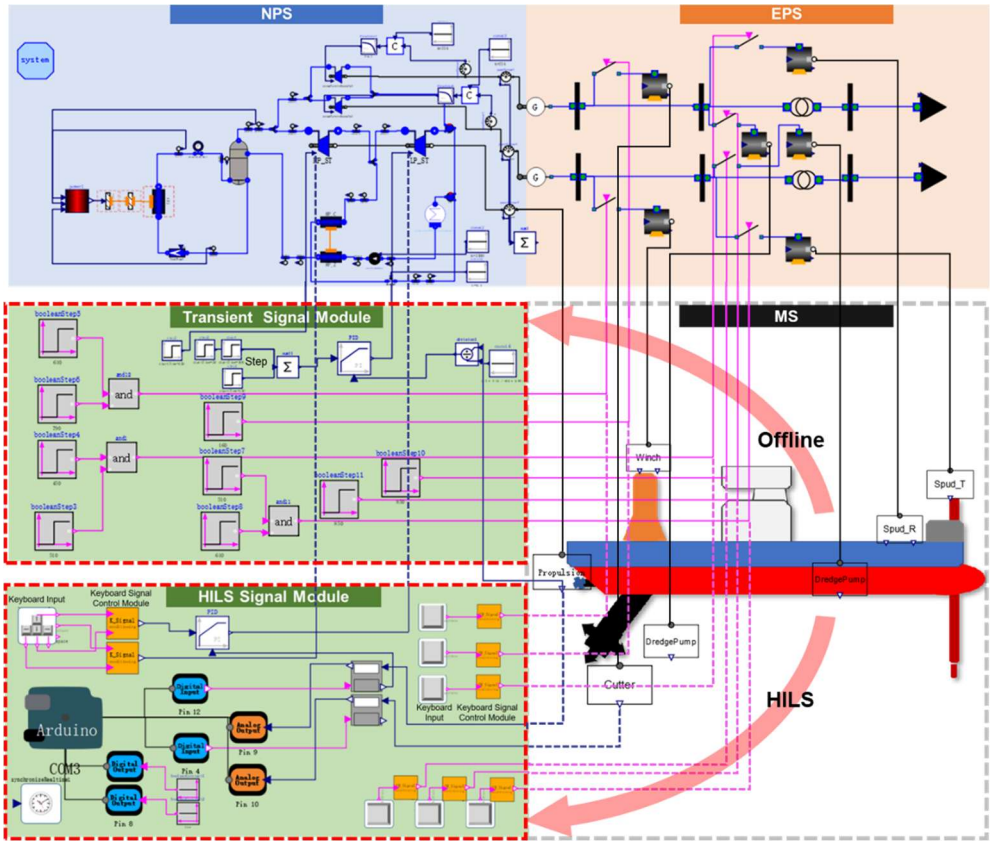


Figure 14. Software-level Models of the NPES in Offline and HILS Modes (Zhang et al. 2024).

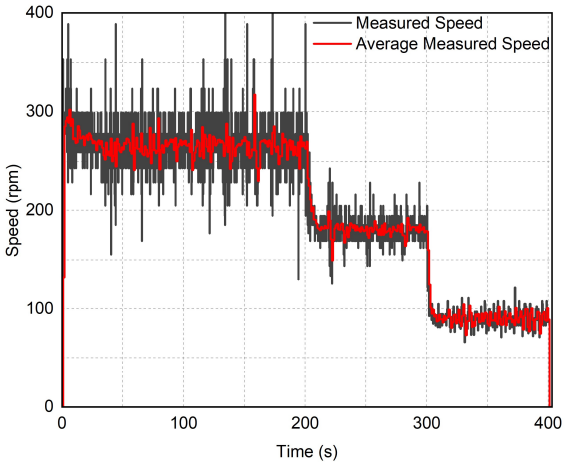


Figure 15. Simple filter process of the collected hardware response signal (Zhang et al. 2024).

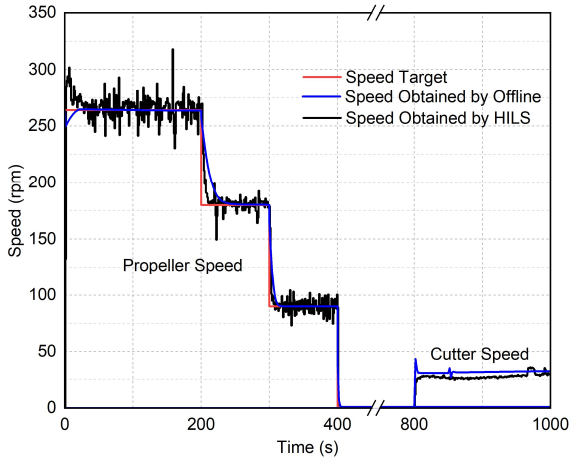


Figure 16. Data recording of hardware response and comparison of propeller speed control in HILS full condition experiment (Zhang et al. 2024).

3.2 Multi-Objective Optimization

In the He-Xe CBC system, multiple performance targets cannot be simultaneously maximized under the same operating condition, which necessitates a multi-objective optimization (MOO) approach. The NSGA-II algorithm (Deb et al. 2002) is applied within the feasible domain to achieve balanced optimization across conflicting objectives.

We implemented an NSGA-II optimization framework in Python (Jazzbin et al. 2020), coupled with the He-Xe CBC system model in OpenModelica via OMPython (Ganeson 2012) for MOO calculations (Zhang and Wang 2024). Figure 17 illustrates the MOO methodology architecture. OMPython links the NSGA-II algorithm to OpenModelica, relaying parameter settings for simulation and returning results to the optimization process. The MOO objectives include system efficiency, specific work, and the Brayton cycle system's equivalent volume with turbomachinery aerodynamic load distribution.

Figure 18 illustrates the Pareto optimal solution set obtained by optimizing two selected objectives, and Figure 19 shows the Pareto optimal solution set for three-objective optimization with different population sizes.

These results demonstrate the Modelica-based model's capability to integrate additional algorithms to support complex MOO studies in energy systems.

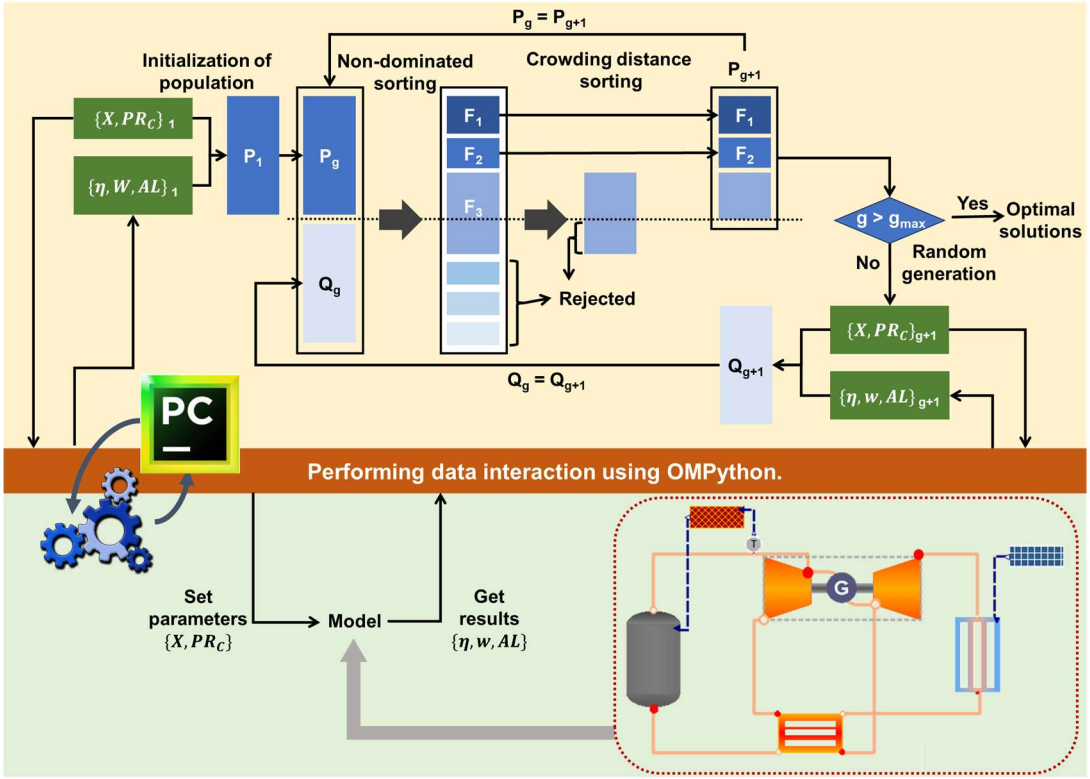
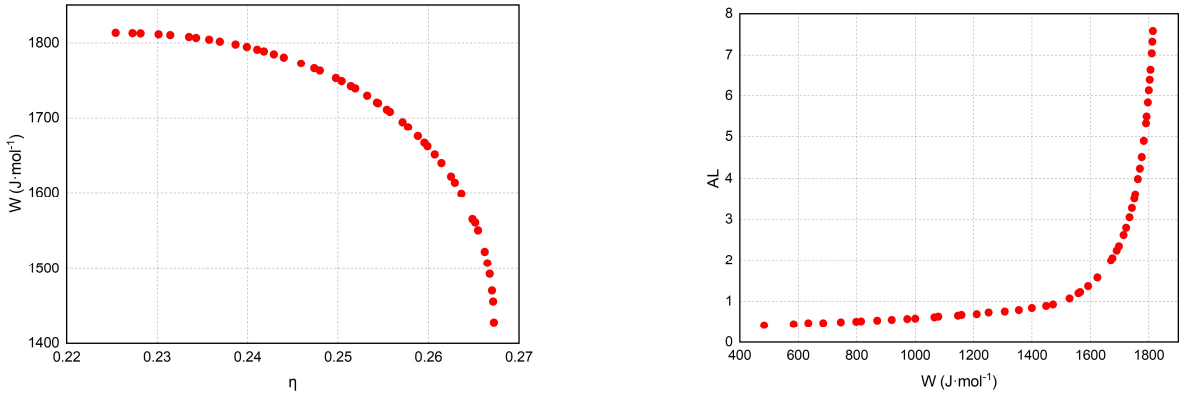


Figure 17. Workflow of NSGA-II MOO using OpenModelica and OMPython (Zhang and Wang 2024).

DOI 10.3384/ecp218373



(a) Efficiency–specific work

(b) Specific work–aerodynamic load

Figure 18. Pareto optimal solution set for two-objective (Zhang and Wang 2024).

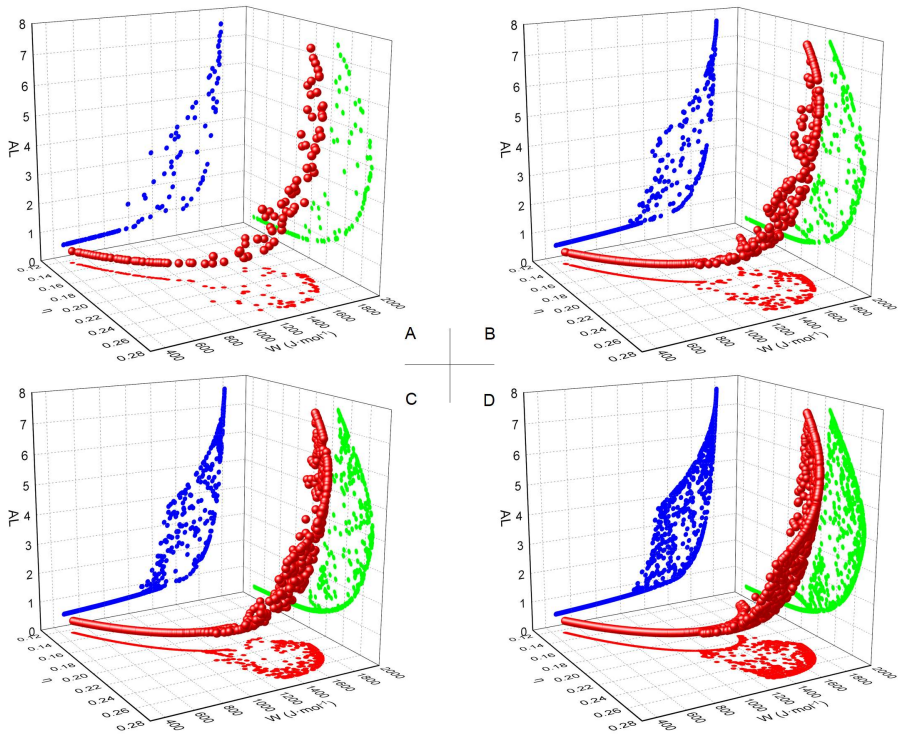


Figure 19. Pareto optimal solution set for three-objective optimization under different population sizes (Zhang and Wang 2024).

3.3 System Identification

System identification is essentially system approximation, aiming to find a model with acceptable accuracy from the data. The resulting model is an approximate representation of the actual system (Ljung 2010). The surrogate model obtained after system identification can accelerate the computational process, helping to reduce the time spent on repetitive calculations or iterations of system models. This is particularly beneficial in scenarios such as multi-objective optimization design and multi-dimensional model coupling calculations.

In the identification of steady-state systems, we constructed a result dataset based on simulations with a variety of different design parameters. The resulting dataset was used to train a neural network model, establishing a proxy for the system-wide input-output mapping. The BPNN is structured with an input layer, hidden layers, and an output layer (Figure 20). Nodes in the input layer represent the system's design parameters. The output layer includes the system's operational or performance parameters, such as temperatures and pressures at various device nodes, system efficiency, specific work, and aerodynamic loading. Each output parameter is mapped through a multi-node hidden layer, collectively forming a comprehensive model that represents the input-output relationships across the entire system.

Figure 21 compares the simulation results and network model calculations for 10 sets of randomly generated design parameters. The maximum relative error was approximately 1%, which indicates that the precision of

the constructed network model is acceptable. Furthermore, the network model, which can surrogate the original model, can be efficiently reused to meet a wider range of computational needs or scenarios.

We are also working on the identification of dynamic system models in AESE, using dynamic mode decomposition (DMD) methods to predict the system's evolution over the long term or during a specific phase (Figure 22), as well as using controlled dynamic mode decomposition (DMDc) methods for rapid prediction of multi-condition transient scenarios (Figure 23). DMD extracts coherent spatiotemporal structures from system data, while DMDc incorporates control inputs to capture input–output dynamics (Zhang et al. 2025). The rapid prediction of a system's transient response aids in efficiently and intelligently handling dynamic data changes, and is of great significance for scenarios such as fast optimization of control systems, lifespan prediction, and fault diagnosis of NPSs.

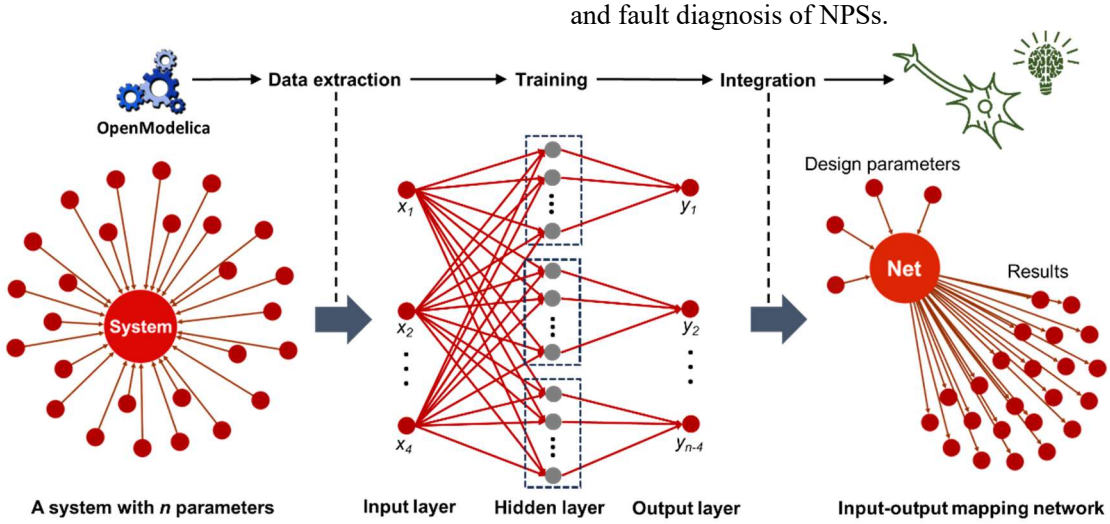


Figure 20. Neural network structure for mapping system design parameters to key performance outputs.

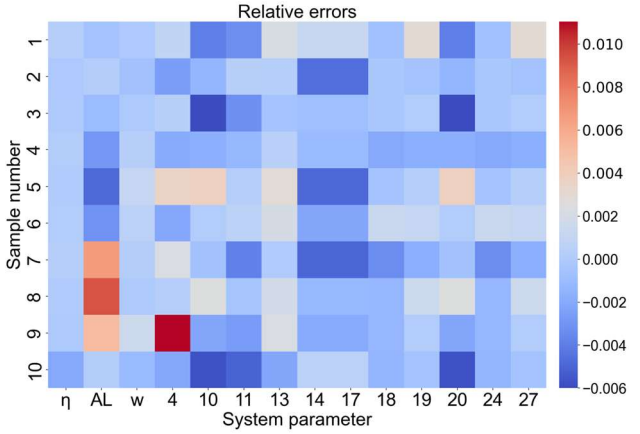


Figure 21. Verification results of the network model for key system parameter.

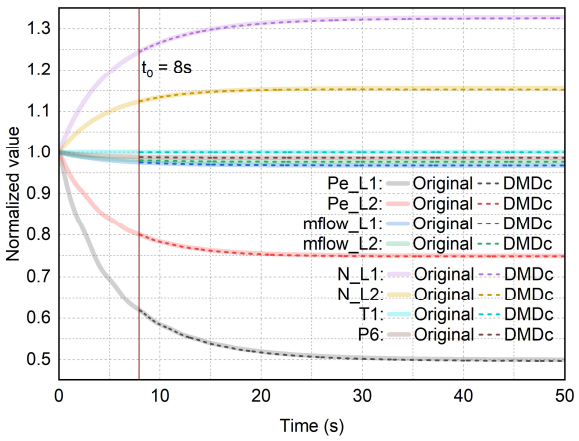


Figure 22. Dynamic prediction of key system parameters based on DMD (Zhang et al. 2025).

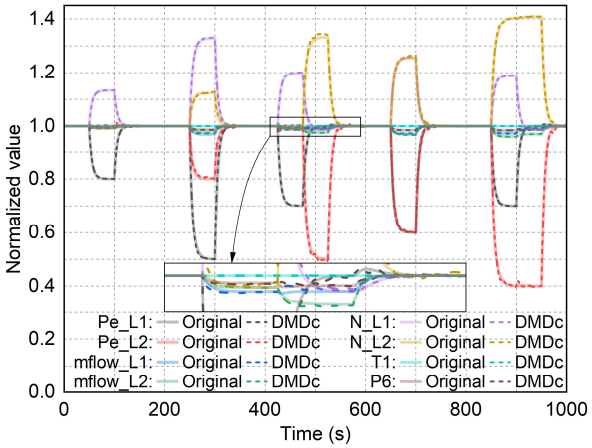


Figure 23. Dynamic prediction of key system parameters based on DMDc (Zhang et al. 2025).

3.4 Rapid Multi-Objective Optimization

As mentioned in the previous section, MOO can be more efficient due to the rapid computation enabled by the surrogate model formed after system identification.

During the optimization process, even with parallel computing, the process remains complex and time-consuming. This paper integrates the BPNN algorithm into machine learning to establish a data-driven multi-objective (DDMO) optimization framework (Figure 24). Using this framework, the system model simulation is not directly embedded in the MOO algorithm. Instead, a large

DOI
10.3384/ecp218373

dataset is first generated through simulations under various design parameters to collect data on different design levels of the system model. Using the BPNN algorithm, a network model is trained based on this dataset to accurately map the original system simulation model. Then, this network model is integrated into the MOO algorithm framework for the optimization design of the system.

Figure 25 presents the results of MOO for the He-Xe CBC system for different numbers of individuals (I) and generations (G) for both model-driven multi-objective (MDMO) and DDMO approaches. The results show consistency between the two modes of optimization. The DDMO optimization had significantly shorter computation times than the traditional MODO approach (Table 3). The DDMO shows faster optimization while maintaining accuracy. Even considering the impact of computer performance fluctuations, the data clearly shows that the DDMO framework accelerated the optimization process by over 10000 times.

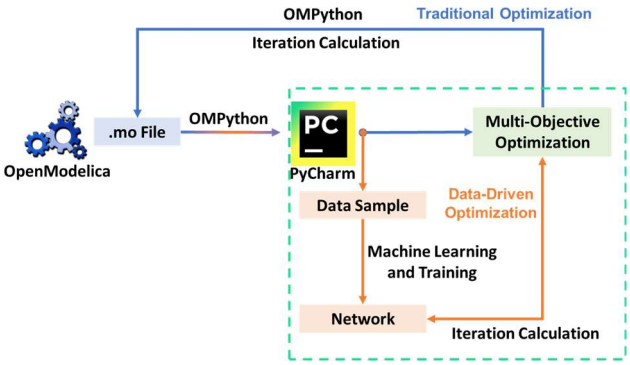


Figure 24. DDMO optimization framework.

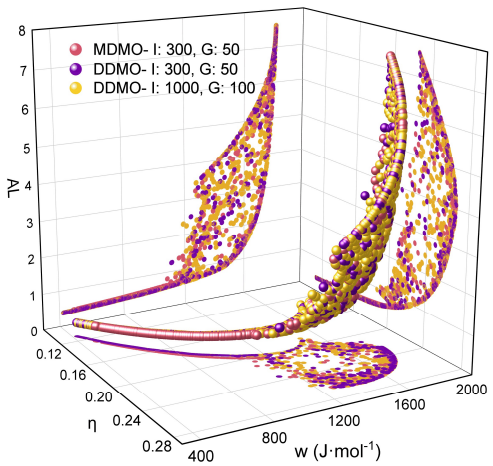


Figure 25. Results of the DDMO.

Table 2. Comparison of calculation times for two optimization methods.

Project	*MDMO*	*DDMO*	*DDMO*
Dual objectives	I: 50 G: 100	I: 50 G: 100	I: 1000 G: 100
η-w	75189.5 s	2.2 s	260.7 s
η-AL	58801.6 s	1.9 s	241.8 s
w-AL	56538.8 s	2.1 s	255.3 s
Triple objectives	I: 300 G: 50	I: 300 G: 50	I: 1000 G: 100
η-w-AL	164502.9 s	55.8 s	185.7 s

4 Conclusions

This paper provides a brief introduction to the AESE library developed for energy systems, particularly NPSs, based on the Modelica language on the OpenModelica platform. It summarizes the modeling and simulation validation of two classic types of models and specifically describes four scenarios for the further application of system simulation models.

Based on Modelica for multi-domain visualization development targeting NPSs, and benefiting from the openness of the OpenModelica platform, system models can be more conveniently integrated with hardware or algorithms to complete tasks such as hardware-in-the-loop simulation, multi-objective optimization, and system identification, addressing task requirements in design verification, operational control, fault diagnosis, and other scenarios in practical engineering with greater accuracy, safety, and speed. Model development is not limited to merely establishing models; further application of models also holds practical significance. Making good use of models and their data can lead to higher productivity.

In current applications, data-driven surrogate models often exhibit strong specificity, which is closely related to the datasets of their particular application targets. As a result, the advantages of surrogate models have not received sufficient attention. To enable more generalized rapid design and optimization of nuclear energy systems, our next step will focus on abstracting the essential nature of energy system models and integrating identification algorithms. This will allow intelligent participation and decision-making throughout the entire design and operation process within a moderately generalized NPS. In addition, to compensate for the limited accuracy of system simulation in modeling key components, the integration of Modelica-based system simulation with other three-dimensional computational tools also deserves further attention.

References

Andreasson, J., Andersson, D., Batteh, J., Gohl, J., Griffin, J., Krueger, I. (2013) “Integrated simulation of a e4WD vehicle using Modelica”, *IFAC Proceedings Volumes*, 46(21), pp. 440–445. DOI:10.3182/20130904-4-JP-2042.00097.

Briese, L.E., Acquatella B, P., Schnepper, K. (2020) “Multidisciplinary modeling and simulation framework for launch vehicle system dynamics and control”, *Acta Astronautica*, 170, pp. 652–664. DOI:10.1016/j.actaastro.2019.08.022.

Cammi, A., Casella, F., Ricotti, M.E., Schiavo, F. (2011) "An object-oriented approach to simulation of IRIS dynamic response", *Progress in Nuclear Energy*, 53(1), pp. 48–58. DOI:10.1016/j.pnucene.2010.09.004.

Casella, F. and Leva, A. (2005) "Object-Oriented Modelling & Simulation of Power Plants with Modelica", in Proceedings of the 44th IEEE Conference on Decision and Control. 44th IEEE Conference on Decision and Control, Seville, Spain: IEEE, pp. 7597–7602. DOI:10.1109/CDC.2005.1583388.

Casella, F. and Leva, A. (2024) "Modelling and Simulation of Thermal Power Generation Systems with Modelica: 20 Years of Experience with the ThermoPower Library", *IFAC-PapersOnLine*, 58(13), pp. 227–234. Available at: https://doi.org/10.1016/j.ifacol.2024.07.487.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T. (2002) "A fast and elitist multiobjective genetic algorithm: NSGA-II", *IEEE Transactions on Evolutionary Computation*, 6(2), pp. 182–197. DOI:10.1109/4235.996017.

El-Genk, M.S. and Tournier, J.-M. (2007) "Noble-Gas Binary Mixtures for Closed-Brayton-Cycle Space Reactor Power Systems", *Journal of Propulsion and Power*, 23(4), pp. 863–873. DOI:10.2514/1.27664.

Fritzson P. (2012) *Introduction to Modeling and Simulation of Technical and Physical Systems with Modelica: Introduction to Modeling and Simulation of Technical and Physical Systems with Modelica*. ISBN:9781118094259.

Fritzson, P. *et al.* (2020) "The OpenModelica Integrated Environment for Modeling, Simulation, and Model-Based Development", in *Modeling, Identification and Control: A Norwegian Research Bulletin*, pp. 241–295. Available at: https://doi.org/10.4173/mic.2020.4.1.

Ganeson, A. (2012) *Design and Implementation of a User Friendly OpenModelica - Python interface.* http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-78992.

Jazzbin, et al. (2020) *geatpy: The genetic and evolutionary algorithm toolbox with high performance in python.* Available from: http://www.geatpy.com/.

Ljung, L. (2010) "Perspectives on system identification", Annual Reviews in Control, 34(1), pp. 1–12. DOI:10.1016/j.arcontrol.2009.12.001.

Mattsson, S.E., Elmqvist, H. and Otter, M. (1998) "Physical system modeling with Modelica", *Control Engineering Practice*, 6(4), pp. 501–510. DOI:10.1016/S0967-0661(98)00047-1.

Mihalič, F., Truntič, M. and Hren, A. (2022) "Hardware-in-the-Loop Simulations: A Historical Overview of Engineering Challenges", *Electronics*, 11(15), p. 2462. DOI:10.3390/electronics11152462.

Peng, M.J. (2009) *Marine Nuclear Power Plant, 1st ed.*, Beijing: Atomic Energy Press. https://www.las.ac.cn/front/book/detail?id=2307eba88938e84d77e7a76931a15cc1.

Soriano, T., Hien, N.V., Tuan, K.M., Anh, T.V. (2016) "An object-unified approach to develop controllers for autonomous underwater vehicles", *Mechatronics*, 35, pp. 54–70. DOI:10.1016/j.mechatronics.2015.12.011.

Tournier, J.-M., El-Genk, M. and Gallo, B. (2006) "Best Estimates of Binary Gas Mixtures Properties for Closed Brayton Cycle Space Applications", in *4th International Energy Conversion Engineering Conference and Exhibit (IECEC). 4th International Energy Conversion Engineering Conference and Exhibit (IECEC)*, San Diego, California: American Institute of Aeronautics and Astronautics. DOI:10.2514/6.2006-4154.

Zhang, A., He, X., Cammi, A., Wang, X. (2024) "Modelica and Arduino-based hardware-in-the-loop simulation of a nuclear-powered engineering ship", *Nuclear Engineering and Design*, 429, p. 113650. DOI:10.1016/j.nucengdes.2024.113650.

Zhang, A. and Wang, X. (2024) "Development of Modelica-based one-dimensional thermodynamic cycle library and its application in simulation and multi-objective optimization of a He–Xe closed-Brayton-cycle system", *Progress in Nuclear Energy*, 172, p. 105205. DOI: 10.1016/j.pnucene.2024.105205.

Zhang, A., Cammi, A. and Wang, X. (2025) "System-level data-driven identification of complex closed Brayton cycles via dynamic mode decomposition", *Applied Thermal Engineering*, p. 127438. DOI: https://doi.org/10.1016/j.applthermaleng.2025.127438.

An Open-Source Industrial-Grade Collection of Renewable Energy Source Generic Models in Modelica Language

Joy El Feghali[1] Louis Garbay[1] Adrien Guironnet[1] Philibert Parquier[1] Marco Chiaramello[1]
Martin Franke[2] Luka Plavec[2]

[1]R&D department, RTE Réseau de transport d'électricité, France {joy.elfeghali, louis.garbay, adrien.guironnet, philibert.parquier, marco.chiaramello}@rte-france.com
[2]Fraunhofer Institute for Energy Economics and Energy System Technology IEE, Power System Stability and Converter Technology Division, Germany {martin.franke,luka.plavec}@iee.fraunhofer.de

Abstract

In a context of massive penetration of Renewable Energy Sources (RES) in the grid, it is of prime importance for Transmission System Operator (TSO) to correctly represent these components in their day-to-day dynamic simulations used to assess system stability, both in operational and planning situations. Well-parametrized generic Power Electronics Interfaced Resources (PEIR) models have emerged as a viable option for TSOs for large-scale stability situations, and are largely available in commercial power system software. However, despite previous efforts in the open-source community, especially in the Modelica one, there is a lack of a complete, up-to-date, and industrial-grade library. This paper presents the current status of the Dynaωo Modelica library for standard PEIR models that tries to fill this gap. The library's architecture and content, the implementation choices, and already existing industrial usages of the models are presented in detail, alongside validation cases and results.

Keywords: Western Electricity Coordinating Council (WECC), International Electrotechnical Commission (IEC), Power Electronics Interfaced Resources, Modelica, Dynawo

1 Introduction

Transmission systems are undergoing significant transformation due to the massive penetration of PEIR with the increased integration of RES and High-Voltage Direct Current (HVDC) connections.

Adding these new components to the system significantly impacts its dynamic behavior, which is crucial for accurately assessing system stability. The primary role of the electrical TSO is to ensure the security and stability of the system at all times through both preventive and corrective actions, supported by continuous monitoring and simulations. To achieve this, it is essential to accurately model and integrate these new components into time-domain simulations.

However, creating dedicated Root-Mean-Square (RMS) dynamic models for each new installation has proven to be a non-sustainable approach, due to the large number of new connections to the grid each year and the difficulty in obtaining open and transparent descriptions of the controls for intellectual property reasons. Two strategies are currently used by the different TSO across the world to tackle this issue: use black-box models directly provided by the Original Equipment Manufacturer (OEM) or/and require well-parametrized generic models allowing to correctly represent the installation behavior (EirGrid 2022; NGESO 2024; TenneT 2024).

Even if generic models can cover a large spectrum of phenomena, only adapting their parameters to the project-specific settings allows one to properly fit the response of many different installations. These standard models are mainly proposed by two entities: the IEC and the WECC. IEC 61400-27-1 international standard describes Wind Turbine (WT) and Wind Power Plant (WPP) models (61400-27-1:2020 2020). The WECC standard contains models for WPP, Photovoltaics (PV) and Battery Energy Storage Systems (BESS) installations (Pourbeik 2014).

A large part of these generic models is available in power systems commercial software such as DIgSILENT Power Factory, Siemens PTI PSS®E, or General Electric PSLF. However, in these environments, even if the general block diagram structures are visible and readable, the individual blocks and their implementation choices are hidden: this could lead to possible discrepancies in the results that are difficult to explain (Villena-Ruiz et al. 2019; Franke, Guironnet, and Cardozo 2024). On the other hand, Modelica is a high-level and open-source modeling language (Fritzson and Engelson 1998), (Mattsson, Elmqvist, and Otter 1998), which is particularly well-suited to offer transparent and unambiguous modeling of power system components in general (L. Vanfretti et al. 2013) and RES in particular.

Open-source implementations of WECC generic models have been proposed and released both in Dynaωo (Nuschke et al. 2021) and OpenIPSL (Fachini, Luigi Vanfretti, et al. 2021) Modelica libraries in recent years. Nevertheless, both efforts only cover part of the currently available WECC models due to the fast evolution of the standard. On the IEC side, the standard course of evolution is slower but the main previous work (Carbonell et al.

2023) focused on the implementation of the Edition 2020 Type IV model, which is not the mostly used by OEMs as of today from RTE's, the French TSO, experience. Recent contributions around Modelica and generic models were rather focused on improvements to the standard models, either to make them more adapted to weak grid conditions (Carbonell et al. 2023), to compare them and provide recommendations for further improvements (Franke, Guironnet, and Cardozo 2024) or to better match power hardware in the loop results (Fachini, Chang, et al. 2025), than on the completeness of the libraries. As a result, there is a lack of a complete, up-to-date and industrial-grade collection of generic RES models in Modelica. This paper tries to fill this gap by presenting the current status of the Dynaωo Modelica library which contains all the necessary standard models to cover all kinds of new RES installations that would like to connect to the grid. This open-source library will facilitate the collaboration between the different power system's stakeholders, offering a basis for discussions on the unavoidable implementation choices and easing the diffusion and usage of generic models in the academic and industrial communities.

The main contributions of this paper are thus the following:

- A presentation of a complete, up-to-date and industrial-grade Modelica collection of generic models for any kind of RES.
- A description of the modeling approach and the implementation choices done in the library.
- An introduction on the current and foreseen industrial usages of the RES models in RTE.

The paper is organized as follows. Section 2 presents the complete set of PEIR generic models available in the library as well as its global architecture. Section 3 and section 4 are dedicated to the IEC and WECC parts of the library, respectively. They provide an overview of the models, a description of the Modelica implementation, a focus on the newly implemented parts of the controls as well as on the final available models, and conclude with a models validation paragraph. Section 5 introduces two envisioned usages of the library from RTE's perspective. Finally, section 6 draws conclusions and presents potential next steps on this topic.

2 Library Content and Architecture

The Dynaωo Modelica library[1] contains at least one generic model for any kind of RES installation.

The models can be found under a PEIR package, as shown in Figure 1, containing both converter and plant models. Plant models are of prime importance for the TSOs as they are the ones that will be used in day-to-day operational and planning large-scale studies; converter models can also be useful if one wants to focus on the fast transient driven by the converter controls. For both packages, application-specific and non-specific models have been implemented.

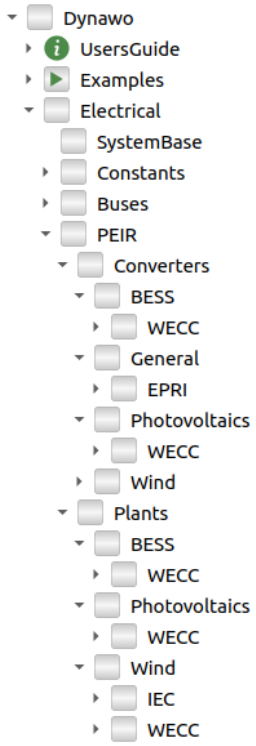


Figure 1. PEIR package in Dynaωo Modelica library.

- Application-specific models for BESS, PV and WPP described in IEC and WECC standards. The list in Table 1 will be presented in detail in the rest of this paper.
- Models with non-specific application "*General*" (i.e. could be used for PV, WPP or BESS components) from IEC and WECC standards or from projects related to grid following or grid forming converters. This part of the library is out of the scope of this paper.

These models have been developed by combining different individual modules available in the control part of the library under a dedicated PEIR package and physical electrical parts such as lines, transformers, and injectors.

PEIR	***Subtype***	***Available models***
Wind	Type III WT	IEC (61400-27-1:2020)
	Type III WPP	IEC (61400-27-1:2020), WECC
	Type IV WT	IEC (61400-27-1:2015/2020), WECC
	Type IV WPP	IEC (61400-27-1:2015/2020), WECC
PV	–	WECC
BESS	–	WECC

Table 1. Available PEIR models in Dynaωo Modelica library.

[1] https://github.com/dynawo/dynawo

DOI 10.3384/ecp218383

3 IEC Models

3.1 Overview and Modelica implementation

The IEC 61400-27-1 standard specifies generic and modular dynamic models suitable for power system stability studies. These models may be used to represent the dynamic behavior of WPP at their Point of Connection (PoC) to the transmission network and of WT at their output terminal. The overall structure of the WPP model proposed in the standard can be found in (61400-27-1:2020 2020).

- The **WPP control and communication** model, also known as Power Park Control (PPC), is in charge of controlling the active and reactive power (P,Q) of the whole plant at the PoC, by computing and sending appropriate orders to the WT. The ancillary services required by TSO, like primary voltage regulation or frequency sensitive mode are handled in the PPC. The PPC is mainly involved in slow dynamics (several seconds).

- The **WT** model is intended to represent an aggregation of a set of identical Wind Turbines. The WT is receiving an active power order p_{WTref}[2] and a voltage or reactive power order (depending on the chosen control mode) x_{WTref} from the PPC. These orders are used to compute reference active and reactive currents, which are then limited to respect the physical limits of the unit. These currents $(i_{\text{pcmd}}, i_{\text{qcmd}})$ are then injected based on the synchronization angle θ_{Pll} given by the Phase-Locked Loop (PLL). The control does not incorporate any inner current loop, so the converter is modeled solely as a current source. The WT control is involved in faster dynamics than the PPC (from ms to a few seconds) and is, for example, essential to represent the Fault Ride Through (FRT) behavior.

- **AUX** model is intended to represent auxiliary equipments which may be, for instance, STATCOMs or static VAR compensators. They have not been implemented at the moment but will be considered in future works.

- **Power Collection System (PCS)** represents an aggregation of the internal network between the WT and the PoC. In large WPPs, it is a crucial part of the model as there could be dozens of kilometers of cables in the real installation. The IEC standard does not specify a clear and unambiguous model for the PCS, even if it proposes a method to compute aggregated parameters based on the real installation complete internal network characteristics. In the Modelica implementation proposed in this paper, it has been decided to represent the PCS with a classical Π line model : $(R+jX)$ line impedance and $(G+jB)$ shunt admittance.

The overall Modelica implementation of the WPP models has been made to be as close as possible to what is specified in the standard. The only deviations at this level are the following:

- f_{sys} has been replaced by *omegaRef* which is the notation used in Dynaωo for the system frequency (which means the rotation speed of the reference frame in which the phasors are computed).

- A measurement block has been inserted to obtain the WPP terminal complex voltage and current.

- An additional input *tanPhi* has been considered for the WT model (in the standard *tanPhi* is considered as a fix parameter).

3.2 Models description

3.2.1 Type IV Model

Type IV WT, also called full-converter WT, are interfaced to the grid with a AC/DC and then a DC/AC converter. This creates a decoupling between the mechanical system of the WT and the electrical system. Moreover, this type of WT has a very flexible control that can be used to offer support functions to the grid. The implementation of the Type IV model based on the Ed. 2020 of the IEC 61400-27-1 standard is presented in details in (Carbonell et al. 2023). A first version of the standard was published in 2015 which is still used by many vendors. Considering this, an implementation of the IEC Type IV Ed. 2015 has been added to the Dynaωo library. The main differences between the two editions of the standard are described below and summarized in Table 2.

- **Dedicated measurement module**: The 2020 version includes a dedicated grid measurement module that is used once for protection and once for control at the WT level and once for the PPC at the WPP level. In the 2015 version, there is no such centralized measurement module, and the filtering process is made independently in each control or protection sub-module. The centralized version used in the 2020 version avoids duplicating the declaration of the time constants and decreases the risk of inconsistencies.

- **WT electrical system**: The 2020 version includes an electrical system module to represent electrical components between the generator system (which is the output of the converter) and the point where the WT is connected. It would typically be used to represent the MV/LV transformer of the WT. In the 2015 version, this electrical system module is just a circuit breaker that will disconnect the WT based on the flag

[2] All variables and parameters shown for the models are in per unit (pu). In Dynaωo library, a *Pu* is added to the variable and parameter names, it is removed in this article for better readability.

F_{OCB} (Open Circuit Breaker). The 2015 version of this module can be extracted from the 2020 version just by setting $r_{es} = x_{es} = g_{es} = b_{es} = 0$. Considering this, the electrical system module is implemented for both versions.

- **Reactive current injection**: The additional reactive current to be injected during fault differs between the two versions. In the 2015 version, it is proportional to the voltage measured at WTT terminal u_{WTT}, with a dead-band $[u_{db1}, u_{db2}]$: $i_{qv} = K_{qv}(u_{db1} - u_{WTT})$ when $u_{WTT} \leq u_{db1}$; while in the 2020 version, it is proportional to the derivative of u_{WTT} : $i_{qv} = K_{qv}(\Delta u_{db1} - \Delta u_{WTT})$ when $\Delta u_{WTT} \leq \Delta u_{db1}$. This derivative is computed with a time constant t_{Uss}, if t_{Uss} is very large this becomes equivalent to the 2015 version, while if t_{Uss} is small the converter will inject reactive current only with fast variations of voltage.

- **Over-Voltage Ride Through (OVRT)**: In addition to the Under-Voltage Ride-Through (UVRT) operation, the 2020 version also includes an OVRT operation. This aims to adapt the reactive current order if voltages exceed a given parameter U_{qRise}. The control path for additional reactive current in the case of OVRT is the same as in the case of UVRT but with the opposite reaction, considering that this path reacts to voltage variations (see previous point). This part of the control is not present in the 2015 version, which means that there is no particular operation in case of over voltages.

- **WPP description**: The 2020 version includes a full description of WPP including PPC with its two main control blocks (active and reactive power controls) along with the communication and measurement modules. Two variants are proposed for the communication module : a linear one consisting of lead-lag transfer functions applied to every communicated variable, and a pure delay one. At this time, only the linear variant has been implemented but the pure delay variant could be developed in the future to better observe some dynamics. In the 2015 edition, a draft version of PPC active and reactive control blocks was proposed in the standard, but without any general structure to build the full WPP model. The choice made in the Dynaωo implementation is to use the structure proposed in the 2020 version and put into it the 2015 control blocks to build 2015 WPP models.

Additionally, it is important to keep in mind that there are two variants of Type IV WT : Type IV A and IV B, the only difference being that Type IV B includes an additional module giving a simplified representation of the mechanical part.

	2015 version	***2020 version***
Measurement	Electrical quantities independently filtered.	Centralized measurement module.
Electrical system	Not in the norm but added in the Modelica implementation.	Classical PI-system between converter and WT terminal.
UVRT reactive current injection	Included and proportional to the voltage u_{WTT}.	Included and proportional to the derivative of the voltage u_{WTT}.
OVRT reactive current injection	Not included.	Included.
WPP	No main structure in the norm but modeled in Dynaωo based on the 2020 structure with the 2015 control blocks.	Full description of WPP including PPC with P and Q/U controls, communication and measurement modules.

Table 2. Summary of the main differences between 2015 and 2020 versions of the IEC 61400-27-1 standard and additional Modelica implementation

3.2.2 Type III Model

The Type III model represents a wind power plant with Doubly-Fed Induction Generators (DFIG). It shares many of its sub-models with the Type IV model. The differing models are presented in more detail below:

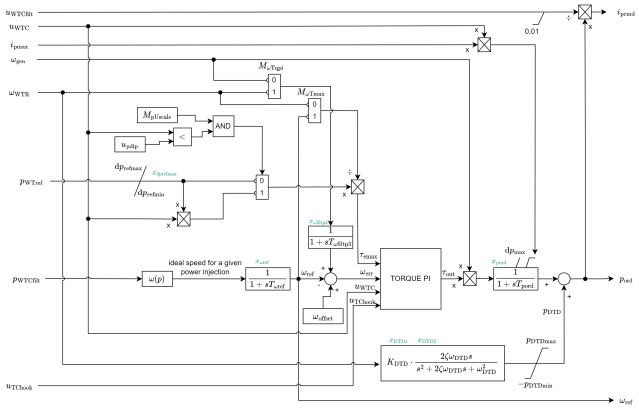
- **Active power control** of the wind turbine control, including its sub-model, the **Torque PI controller** (see Figure 2). Its general behavior is influenced by an $\omega(p)$ lookup-table that provides the angular velocity at which the turbine should rotate when it is injecting a certain active power. The specific implementation choices in these controllers, especially integrators, limiters and their interactions, are delicate and can have a large influence on model behavior.

- **Two-dimensional aerodynamic** model (see Figure 3). It models the energy conversion from available wind power p_{avail} to what the wind turbine passes on to the mechanical rotor. This is influenced by the rotor blades' pitch angles θ and the turbine's angular velocity ω_{WTR}.

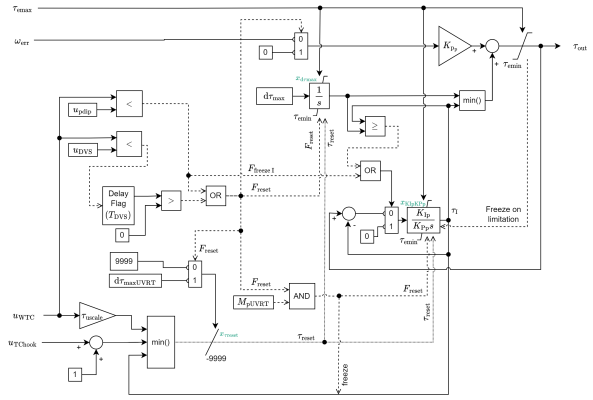
- **Pitch angle control** (see Figure 4). It uses two PI controllers to control the pitch angle, one associated with rotor speed $\omega_{WTR} - \omega_{ref} + K_{PX}(p_{ord} - p_{WTref})$ and one with generator power $p_{ord} - p_{WTref}$.

- **Generator system models** (see Figure 5). Two generator system models have been implemented: **III-A**

DOI 10.3384/ecp218383



(a) Active power control model



(b) Torque PI control model

Figure 2. IEC Type III Wind Turbine control models.

with an explicit model of the current control loop and **III-B** with a simple model of a crowbar-circuit.

All the figures are extracted from (*COLib, a collaborative open-source dynamic simulation library for power systems* 2025).

3.3 Models validation

The 2015 version of the Type IV model has been validated by comparing the results with the detailed 2020 version. For the Type III model, in-depth comparisons against the results obtained with the DIgSILENT PowerFactory (DIgSILENT 2024) models have been done. Indeed, a structured qualitative validation procedure has been conducted for the components implemented in OpenModelica (Fritzson, Pop, et al. 2020) as follows:

- Set up the model with inputs and outputs in both simulation environments.
- Define test cases with varying input signals.
- Run the simulations and visually compare output signals using OMPython[3] and powfacpy[4].

[3] https://github.com/OpenModelica/OMPython

[4] github.com/FraunhIEE-UniKassel-PowSysStability/powfacpy

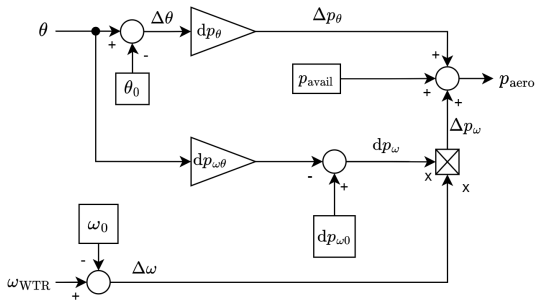


Figure 3. IEC Type III Two-Dimensional Aerodynamic Model.

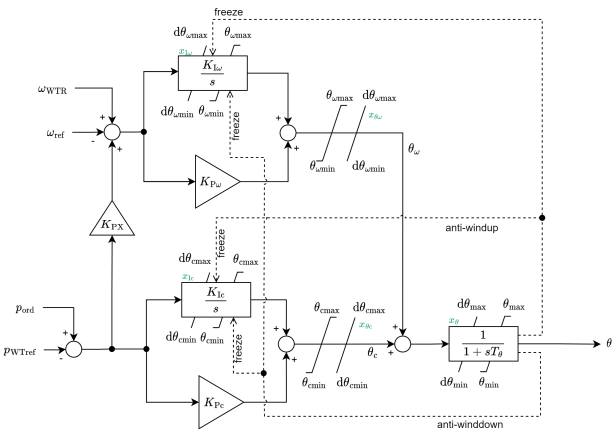


Figure 4. IEC Type III Pitch Angle Control Model.

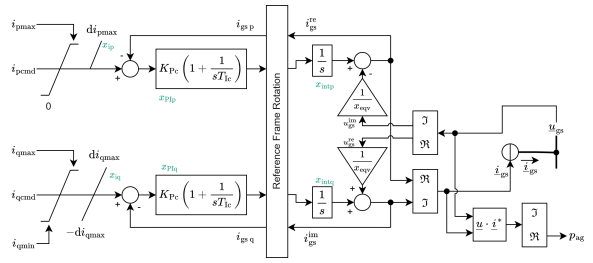
Exemplary results are presented in Figure 6 showing a test case of the validation of the P control model. Several step variations of the input voltage at the converter u_{WTC} are performed, and it is seen that the outputs of the P control model, active power order of the WT p_{ord} and the active current command i_{pcmd}, in OpenModelica and DIgSILENT PowerFactory are identical. In Figure 7, several step variations are done on the input active current command i_{pcmd} of the Type III-A generator system model. The real active current injection i_{gsre} is observed at the output. A perfect match is observed between both implementations.

Figure 8 displays the response of the complete wind turbine model in a Single Machine Infinite Bus (SMIB) configuration to a change of the active power reference setpoint P_{ref} ($= p_{\mathrm{WTref}}$). The responses of both models are very similar, even if not completely identical. All these results allow to validate the WT Type III Modelica implementation.

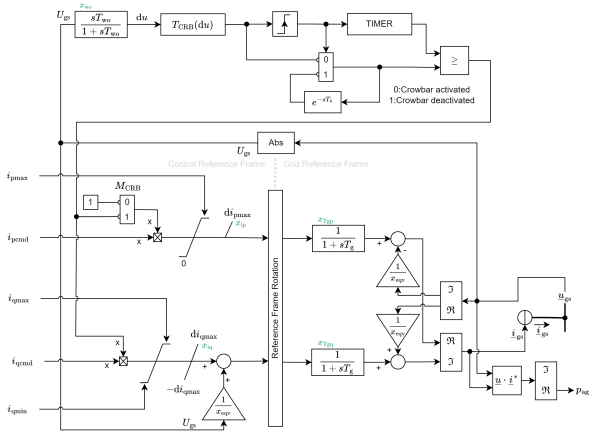
4 WECC Models

4.1 Overview

WECC generic models represent second generation generic renewable energy systems that are interfaced with the grid through power electronics converters, like PV, Wind Turbine Generator (WTG), and BESS. These models are used in transient stability studies where the electric



(a) Generator System A



(b) Generator System B

Figure 5. IEC Type III Wind Turbine Generator Systems.

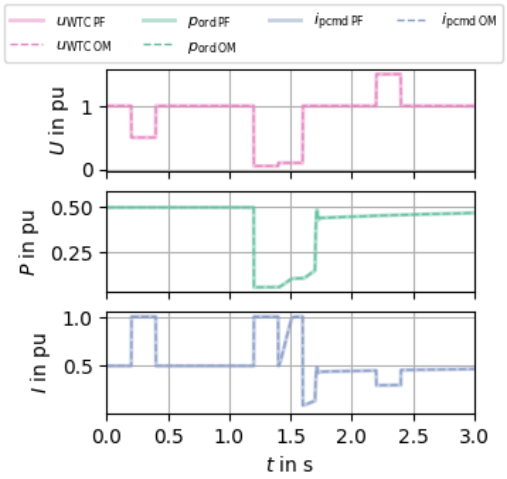


Figure 6. Voltage at converter terminal u_{WTC} variation (pink) for IEC Type III P Control Model. Active power order p_{ord} (green) and active current command i_{pcmd} (bleu) are observed. The dashed lines correspond to DIgSILENT PowerFactory (PF) simulations and the straight lines correspond to OpenModelica simulations.

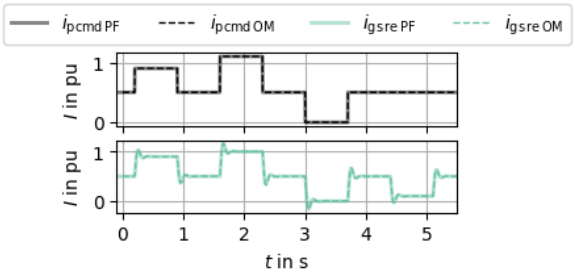


Figure 7. Active current command i_{pcmd} variation (black) for IEC Type III-A generator system model. Real current i_{gsre} (green) is observed. The dashed lines correspond to DIgSILENT PowerFactory (PF) simulations and the straight lines correspond to OpenModelica simulations.

controls and converters are modeled along with the plant controls.

The WECC RES models are implemented by combining the different individual blocks (REPC_*, REEC_* and REGC_*) that represent the controls of an installation as seen in the Figure 9a for PV plants for example. These three control systems and the mechanical system models are presented below :

- The plant control - **Renewable Energy Plant Controller** (REPC_*) models - sets the main controls of the plant : voltage or reactive power control and frequency-dependent active power. The REPC_A model is available in Dynaωo and could be used for PV, WTG, and BESS.

- The electrical control - **Renewable Energy Electrical Controls** (REEC_*) models - includes the local inverter functionalities such as FRT characteristic with fast reactive current injection, local voltage and reactive power control, and current limitation with respect to the priority given to active or reactive current. This control system calculates the active and reactive current set-points *idCmd* and *iqCmd*. Three models are available in Dynaωo : 1) the REEC_A model that is recommended for WTG and now for PV, 2) the simple REEC_B model that was recommended for PV, 3) and the REEC_C model that is recommended for BESS where the State Of Charge (SOC) is modeled allowing negative current (i.e. consumption).

- The generator/converter control - **Renewable Energy Generator/Converter** (REGC_*) models - is the last part of the control and interfaces with the network. It enables the conversion of the current set-points calculated by the REEC part into the final currents (or voltages) delivered to the network. Three models are available in Dynaωo : 1) the REGC_A for current source models, 2) the REGC-B, and 3) REGC-C for voltage source models. All these three blocks could be used for PV, WTG, and BESS models.

- The **WTG mechanical side models** for Type III and Type IV WTGs. All mechanical models introduced in the WECC are available in Dynaωo : the drive-train shaft dynamics models WTGT_A used for Type

DOI 10.3384/ecp218383

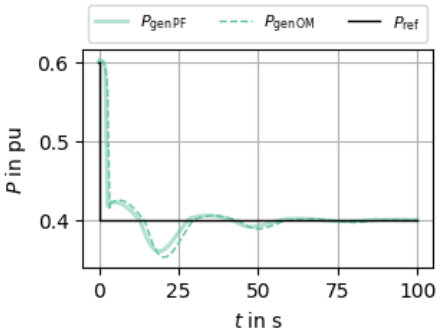


Figure 8. SMIB test comparison of the full IEC Type III-A WT model: Reference power step P_{ref} (black) from 0.6 to 0.4 pu. Generated active power P_{gen} (green) is observed. The dashed line corresponds to DIgSILENT PowerFactory (PF) simulation and the straight line corresponds to OpenModelica simulation.

PEIR	***Model combination***			
	Plant Control	***Electrical Control***	***Generator Control***	***Mechanical Controls***
Wind (Type IV)	REPC_A	REEC_A	REGC_A (_B or _C)	optional WTGT_B (or _A)
Wind (Type III)	REPC_A	REEC_A	REGC_A (_B or _C)	WTGA_A, WTGQ_A, WTGP_A (or _B), WTGT_A
PV	REPC_A	REEC_A (or _B)	REGC_A (_B or _C)	-
BESS	REPC_A	REEC_C	REGC_A (_B or _C)	-

Table 3. Possible combinations for PEIR WECC models in Dynaωo Modelica library.

III WTGs and WTGT_B used for Type IV WTGs. These models represent the rotor speed changes and resulting torsional oscillations after faults or sudden wind speed changes. The pitch controller (WTGP_A and WTGT_B), the torque controller (WTGQ_A), and the aero-dynamics (WTGA_A) models are used to properly model a Type III WTG.

Table 3 presents the current list of possible model combination in Dynaωo Modelica library for PV, WTG Type III, WTG Type IV and BESS. Noting that the new models introduced by the WECC standard, such as REEC_D, REEC_E, REPC_C and REPC_D, are currently being implemented, and will be completing the current list.

4.2 Modelica implementation

As shown in Figure 9b, the implementation in Dynaωo Modelica library follows the same structure as presented in (Ellis, Behnke, and Elliott 2013) for the **IBR level control** where the three control blocks are interconnected and receiving measurements from the network. Inside each of these blocks, the controls are described through a *block-diagram* approach following the latest version of the *Model User Guide for Generic Renewable Energy System Models 2023* (Ramasubramanian 2023).

In the Dynaωo Modelica implementation, each IBR is connected to the grid through a **terminal**. The following blocks, shown in Figure 9b represent the connection between the IBR and the network :

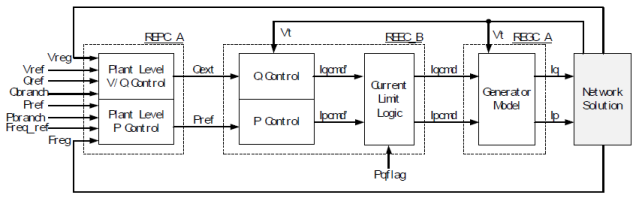
- The current **injector**, where the interface to the grid is a current source, calculates the real and imaginary currents that IBR model will inject. The dq transformation is done inside this block and is described through *equation-based* approach. For voltage source models, the dq transformation is done inside the REGC model.
- The **PLL** is added in all the models to calculate the voltage phase θ needed for the dq transformation. This block is described in a *block-diagram* approach. In the WECC standard, the PLL is neglected for the REGC_A and REGC_B models and only represented for the REGC_C model. In Dynaωo, the PLL was initially added in all models to overcome some numerical issues by adding some dynamics to the voltage phase instead of calculating it instantaneously.
- A classical Π **line** is modeled to represent an aggregation of the Inverter-Based Ressource (IBR) unit transformer and the step-up transformers and is connected to the Point of Common Coupling (PCC).
- Finally, the **measurement** block calculates variables needed for the plant control : the complex voltage and current, and the active and reactive power at the PCC. Note that the other variables needed for the PLL, the electrical control, and generator controls are measured at the injector level.

4.3 Models description

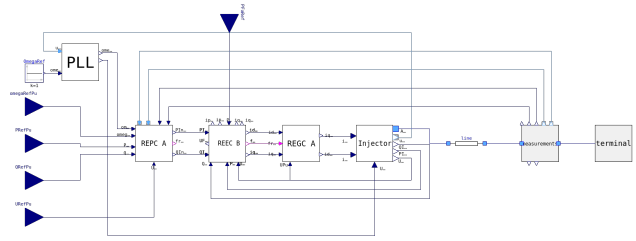
The first WECC generic models implementation in Dynaωo has been presented in (Nuschke et al. 2021), where REPC_A, REEC_A, REEC_B, REGC_A, and WTGT_A are described for PV and WTG modeling. This paper will focus on the new models added such as the REEC_C model enabling the representation of BESS, and the voltage-source models REGC_B and REGC_C for voltage-source interface with the grid.

4.3.1 Electrical control REEC_C model

The REEC_C model is quite similar to the REEC_A model; thus, in Dynaωo Modelica library, these models share the same base-model. Here, some of the differences are highlighted. Table 4 summarizes the key point differences.



(a) WECC PV current source model *(Ellis, Behnke, and Elliott 2013)*



(b) WECC PV current source model implementation in Dynaωo Modelica library

Figure 9. WECC PV current source model.

- The **reactive current injection** during fault is different. The REEC_C injects reactive current actively when the voltage at the injector is different than the reference voltage, unless the gain is set to zero or the deadband is extremely wide. However, the REEC_A has a switching logic between different states in normal operation state, during and after voltage dips, for the reactive current injection mainly related to Type III and Type IV WTG.

- The **d-axis current limits** are different. The minimal d-axis current *idMin* in REEC_A is 0, while the REEC_C model takes the negative value of *idMax*, enabling the charging and discharging of the battery. The limits of the d-axis and q-axis are calculated in the *CurrentLimitsCalculationC* model (equation-based block in REEC_C) depending on the state of charge of the battery, the P/Q priority flag and the Voltage Dependent Limitations (VDL) look-up tables.

- An additional path exists in REEC_C for the **state of charge** and includes an integrator block with a time constant t_{battery}, that represents process of charging and discharging. *SOC*0 is the initial state of charge of the battery and is a parameter in per unit given by the user, having a value between 0 and 1. *SOCMax* and *SOCMin* are, respectively, the maximum and minimum allowable state of charge.

  Values of the active current maximum and minimum limits are conditioned by the SOC. When the maximum state of charge is reached, the minimal active current is forced to zero, the battery cannot store anymore energy. When the minimum state of charge is reached, the battery cannot inject energy anymore. When $SOCMin < SOC < SOCMax$, the active current limits values will depend on the P/Q priority and the VDL look-up tables as explained in (Ramasubramanian 2023).

	REEC_A	***REEC_C***
Reactive current injection	Switching logic for current injection during and after fault.	Active injection of reactive current.
d-axis current limits	$idMin = 0$	$idMin = -idMax$ when not fully charged
SOC	Not included.	Included.

Table 4. Summary of the main differences between REEC_A and REEC_C

4.3.2 Generator control models

All REGC_* models have common parts, such as the inputs and the calculation of the reference q-axis current $iqRef$; this is implemented in a base common model for REGC. Here are some differences between the three models. Table 5 summarizes the key point differences.

- The **inner-current control-loops** : The REGC_B was proposed to remediate the numerical instability issues of the REGC_A in weaker networks by having a voltage source interface. However, the inner-current control-loops are still neglected, as in REGC_A. The REGC_C includes a generic representation of the inner-current control-loops with PI controllers on the d and q axes.

- The **interface with the grid** : In REGC_A, a current-source interface, the outputs of the block are the reference d-axis $idRef$ and q-axis currents $iqRef$ that will be connected to the injector block. In both REGC_B and REGC_C models, a voltage source behind an impedance (*RSource*, *XSource*) is added, with values in per unit, to emulate the source impedance of the Voltage Source Converter (VSC).

  In the Dynaωo Modelica implementation, a *VSourceRef* block is implemented, taking as an input the d-axis and q-axis current references along with the measured voltage dq components $udInj$, $uqInj$ and calculating the d-axis and q-axis voltage references following Listing 1. Then a dq to RI transformation is done, resulting in the real *urSource* and imaginary *uiSource* voltage components that will be interfaced with the grid (Listing 2).

Listing 1. d-axis q-axis voltage references

```
udSourceRef = udInj + idRef * RSource -
    iqRef * XSource
```


DOI 10.3384/ecp218383

	REGC_A	***REGC_B***	***REGC_C***
Current loop control	Not included.	Not included.	Included.
Interface with the grid	Current source interface.	Voltage source interface.	Voltage source interface.

Table 5. Summary of the main differences between REGC_A, REGC_B and REGC_C

```
uqSourceRef = uqInj + iqRef * RSource +
    idRef * XSource
```

Listing 2. Real and imaginary voltage components

```
urSource = udSourceRefTe * cos(theta) -
    uqSourceRefTe * sin(theta)
uiSource = udSourceRefTe * sin(theta)+
   uqSourceRefTe * cos(theta)
```

4.4 Model Validation

4.4.1 BESS model

The REEC_C model in Dynaωo is integrated in the BESS model. This model has been validated against results found in (Pourbeik and Petter 2017) for a SMIB test case, in OpenModelica (Figure 10) and in Dynaωo engines.

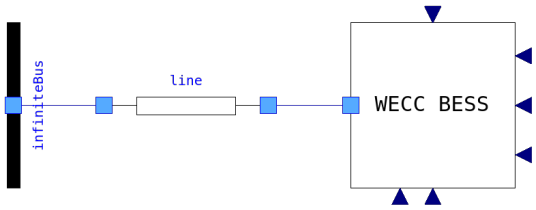


Figure 10. BESS SMIB test case in OpenModelica.

The BESS model comprises the REPC_A, REEC_C and REGC_A control models. It is connected to an infinite bus through an impedant line ($X_{\text{line}} = 2.0661 \cdot 10^{-6}$ pu, base $Sn_{\text{ref}} = 100$ MVA). The initial operating point for the battery is : $P = 0.5$ pu , $Q = 0$ pu (base $SNom = 6$ MVA) - as shown by the plot in Figure 11 - and $U = 1$ pu (base $UNom = 225$kV).

At t = 1 s, a voltage dip ($U = 0.55pu$) is applied at the infinite bus to mimic a three-phase short-circuit in the external network and is then removed at t = 1.5 s (fault clearing). At the fault insertion, the fast reactive current injection starts acting and the reactive current rapidly reaches its maximum value as seen in Figure 11 for *iqCmd*. This value is maintained until the fault clearance. On the active side, it is the normal control path that reacts to the fault by increasing the active current reference to try to maintain the active power despite the voltage dip. However, this increase is limited by the maximum current limitation and the reactive current increase (Q priority). P and Q in Figure 11 confirm this behavior and allows validating the model implementation thanks to a perfect match with the reference results (Pourbeik and Petter 2017).

The active power P plot in Figure 11 also demonstrates that the BESS is able to discharge (producing active power), which is the expected behavior as the initial state of charge $SOC0 = 0.5$ is between the accepted range [$SOCMin = 0.2$, $SOCMax = 0.8$]. The discharge time is much longer than the simulation time considered here.

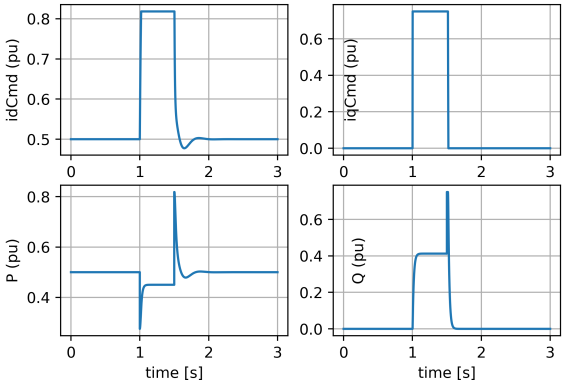


Figure 11. Validation results for a voltage dip - WECC BESS. On the top left, the active current *idCmd* is plotted, on the top right, the reactive current *iqCmd*. On the bottom left, the active power P is plotted and on the bottom right, the reactive power Q. The values are in per unit base $SNom$, $UNom$. Reference results are found in (Pourbeik and Petter 2017).

5 Applications

5.1 Models Validation and Operational Stability Assessment

The Modelica IEC and WECC models presented in the previous sections are used in the model validation process introduced into the French Connection Network Code (CNC) in February 2024 (RTE 2025). Indeed, for each new PEIR installation connected to the grid, the CNC requires the tuning of a generic dynamic model. One of the main criteria for grid code compliance is that the dynamic responses of the tuned generic model closely match those of the proprietary model used by the producer during simulation performance tests and those of the installations during real field tests. To assess the accuracy of this tuning, RTE has defined a set of standardized tests (e.g., three-phase faults, active/reactive power setpoint changes, voltage dips or surges at the point of connection) and associated compliance criteria (Cardozo et al. 2023) and developed an open-source tool (Marin et al. 2024) named DyCoV[5]. Its purpose is to automate the comparison process between:

[5]https://github.com/dynawo/dyn-grid-compliance-verification

1. Signals generated using the tuned generic IEC or WECC model, implemented within the Dynaωo Modelica library,

2. Reference signals obtained from the proprietary model used by the producer, typically simulated in a commercial tool, and from real field tests.

These comparisons are performed over the full set of standardized tests. The overall function of DyCoV is illustrated in Figure 12. The input format are Dynaωo parameters (*par*) and dynamic data (*dyd*) files to describe the tuned generic model and *csv* or comtrade files for the reference signals. In short, DyCoV applies consistent signal processing techniques to both sets of signals, enabling meaningful comparisons. The tool then computes a set of Key Performance Indicators (KPIs), such as the Maximum Error (MXE), Mean Error (ME) and Mean Absolute Error (MAE) between two signals during a disturbance. Finally, DyCoV generates a detailed *pdf* report that includes the test conditions, signal comparisons, computed KPIs, and an evaluation of compliance against RTE's predefined thresholds.

An example of such a report, focusing on an active power setpoint test, is shown in Figure 13 and Figure 14. On Figure 13, the reference signal from the proprietary model is plotted in orange and the tuned generic model, in blue. The event simulated is a step order in active power at t = 20 s. It can be seen that both models are settling around the same point in time at t = 20.93 s. However the reaction time is very different as the reference model reacts with a pure delay. This validation methodology aligns with the principles of the IEC standard (61400-27-1:2020 2020).

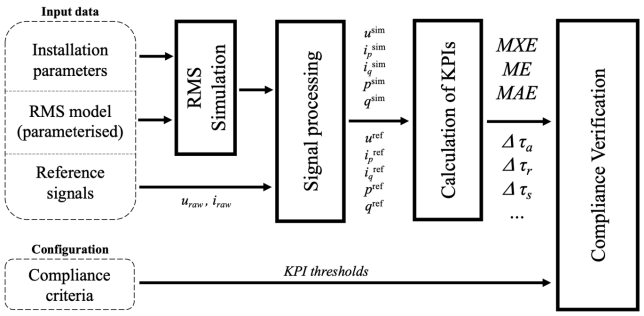


Figure 12. Functional diagram of DyCoV, RTE's dynamic grid compliance verification tool

Crucially, from a Modelica perspective, DyCoV relies entirely on the open-source implementation of IEC and WECC generic models developed within the Dynaωo library. All simulations and compliance assessments are therefore performed using transparent, reproducible, and openly available models.

The validated tuning of the generic models are then stored to complement RTE's historical dynamic data base for time-domain simulations and stability assessment in order to be used in the near future in the operational evaluation of transient, voltage, or small-signal stability.

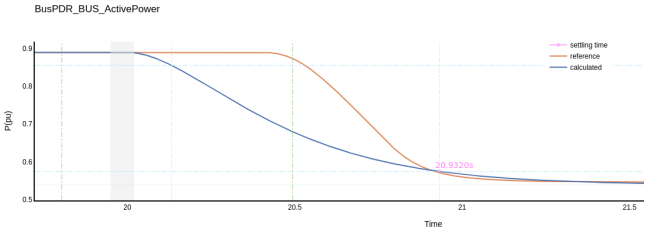


Figure 13. Extract of a DyCoV report. Generic WECC tuned model in blue and proprietary model in orange.

Compliance thresholds on the curves:

	Pre-event			Event		
	MXE	ME	MAE	MXE	ME	MAE
P	0.05	0.02	0.03	0.05	0.02	0.03

Compliance checks on the curves:

	Pre-event			Event			
	MXE	ME	MAE	MXE	ME	MAE	Compl.
P	0.000829	0.000829	0.000829	0.191	0.0841	0.102	False

Figure 14. Extract of a DyCoV report. KPIs calculated based on Figure 13.

5.2 Prospective studies

Another interesting usage for standard IEC and WECC models are prospective studies. Indeed, TSOs can construct a set of parameters for these models that fulfills the minimal requirements from their CNC and then use this set of parameter as default value for the future units.

Such a strategy allows the TSOs to conduct stability studies in prospective situations and to anticipate future issues as well as to easily test the impact of modifications in the CNC, such as a change in the dynamic performance requirement for primary voltage regulation for example.

6 Conclusion and Perspectives

This paper presents the current status of the Dynaωo Modelica library for PEIR components. It notably introduces the library's architecture and content then details the IEC and WECC models available with a special focus on the recently added parts. Models introduced can be simulated in OpenModelica and Dynaωo engines. Amongst the different features of the library, it is important to notice:

- *its completeness*: at least one model is available for any kind of RES installation, and a special effort is done to try to follow the standard evolution.

- *its transparency*: it is an open-source library, thus including the implementation choices for each and every individual block.

DOI 10.3384/ecp218383

- *its industrial quality*: it is already used in the network connection procedure by RTE and so in the associated discussions with stakeholders.

The next efforts will concentrate on two different parallel axes. On the one hand, there are still evolutions on the standard and especially on the WECC side that introduce new and improved control modeling options. It is necessary to keep working on the library to support the newest versions of the models: one key point will be the implementation of newly proposed *Grid Forming* standard models. On the other hand, the industrial usages of the models will increase on RTE side with expected use in operational stability studies and the continuation of exchanges with stakeholders in the connection process. This will help improve the overall quality of the library and make it even more robust.

References

61400-27-1:2020, IEC Standard (2020). *Wind energy generation systems - Part 27-1: Electrical simulation models – Generic models*.

Carbonell, Borja et al. (2023). "An open source Modelica implementation of the IEC 61400-27-1 type 4 wind turbine model for power system stability assessment". In: *Electric Power Systems Research* 214, p. 108313.

Cardozo, Carmen et al. (2023). "Compliance criteria to assess RMS model accuracy for wind power plants based on IEC 61400-27-2". In: *22nd Wind and Solar Integration Workshop (WIW 2023)*. Vol. 2023. IET, pp. 461–468.

COLib, a collaborative open-source dynamic simulation library for power systems (2025). https://colib.net/. Accessed: 2025-04-09.

DIgSILENT (2024). *PowerFactory*. Version 2024 SP4. URL: https://www.digsilent.de/.

EirGrid (2022-02). *Requirements for WECC Model Submission*. https://cms.eirgrid.ie/sites/default/files/publications/Requirements-for-WECC-Model-Submission-v1.0.pdf.

Ellis, Abraham, Michael Robert Behnke, and Ryan Thomas Elliott (2013). *Generic solar photovoltaic system dynamic simulation model specification*. Tech. rep. Sandia National Lab.(SNL-NM), Albuquerque, NM (United States).

Fachini, Fernando, Hao Chang, et al. (2025). "Customized open source renewable energy models validated through PHIL lab experiments". In: *Renewable Energy* 244, p. 122627. ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2025.122627. URL: https://www.sciencedirect.com/science/article/pii/S0960148125002897.

Fachini, Fernando, Luigi Vanfretti, et al. (2021). "Modeling and Validation of Renewable Energy Sources in the OpenIPSL Modelica Library". In: *IECON 2021 – 47th Annual Conference of the IEEE Industrial Electronics Society*, pp. 1–6. DOI: 10.1109/IECON48115.2021.9589148.

Franke, Martin, Adrien Guironnet, and Carmen Cardozo (2024). "Comparing IEC and WECC generic dynamic models for type 4 wind turbines". In: *Electric Power Systems Research* 235, p. 110806.

Fritzson, Peter and Vadim Engelson (1998). "Modelica—A unified object-oriented language for system modeling and simulation". In: *European Conference on Object-Oriented Programming*. Springer, pp. 67–90.

Fritzson, Peter, Adrian Pop, et al. (2020). "The OpenModelica Integrated Environment for Modeling, Simulation, and Model-Based Development". In: *Modeling, Identification and Control* 41.4, pp. 241–295. DOI: 10.4173/mic.2020.4.1.

Marin, J.L. et al. (2024). "An Open-Source Tool for the Validation of Power Park Modules Generic Models". In: *CIGRE Session Materials*.

Mattsson, Sven Erik, Hilding Elmqvist, and Martin Otter (1998). "Physical system modeling with Modelica". In: *Control engineering practice* 6.4, pp. 501–510.

NGESO, National Grid Electricity System Operator - (2024-03). *Guidance Notes On Modelling Requirements - GC0141 Grid Code Modification*. https://www.neso.energy/document/316631/download.

Nuschke, Maria et al. (2021). "Implementation and Validation of the Generic WECC Photovoltaics and Wind Turbine Generator Models in Modelica". In: *Modelica Conferences*, pp. 633–642.

Pourbeik, Pouyan (2014). "WECC second generation wind turbine models". In: *Electric Power Research Institute, Knoxville, TN*.

Pourbeik, Pouyan and Jeffrey K Petter (2017). "Modeling and validation of battery energy storage systems using simple generic models for power system stability studies". In: *Cigre Sci. Eng* 9, pp. 63–72.

Ramasubramanian, D (2023). "Model user guide for generic renewable energy system models". In: *Technical Update 3002027129, EPRI*.

RTE (2025-02). *Documentation Technique de Reference*. https://www.services-rte.com/files/live//sites/services-rte/files/documentsLibrary/20250228$_{D}TR_{9}954_{e}n$.

TenneT (2024-04). *RfG compliance verification Power generating modules types B, C and D, version 2.0*. https://www.netbeheernederland.nl/sites/default/files/2024-03/rfg$_{c}ompliance_{v}erification_{d}ocument_{v}2.0.pdf$.

Vanfretti, L. et al. (2013). "Unambiguous power system dynamic modeling and simulation using modelica tools". In: *2013 IEEE Power Energy Society General Meeting*, pp. 1–5. DOI: 10.1109/PESMG.2013.6672476.

Villena-Ruiz, Raquel et al. (2019). "Implementation of IEC 61400-27-1 Type 3 Model: Performance Analysis under Different Modeling Approaches". In: *Energies* 12.14. ISSN: 1996-1073. DOI: 10.3390/en12142690. URL: https://www.mdpi.com/1996-1073/12/14/2690.

Glossary

BESS Battery Energy Storage Systems. 1, 2, 5–7, 9

CNC Connection Network Code. 9, 10

DFIG Doubly-Fed Induction Generators. 4

FRT Fault Ride Through. 3, 6

HVDC High-Voltage Direct Current. 1

IBR Inverter-Based Ressource. 7

IEC International Electrotechnical Commission. 1–3, 6, 9, 10

KPIs Key Performance Indicators. 10

MAE Mean Absolute Error. 10

ME Mean Error. 10

MXE Maximum Error. 10

OEM Original Equipment Manufacturer. 1, 2

OVRT Over-Voltage Ride Through. 4

PCC Point of Common Coupling. 7

PCS Power Collection System. 3

PEIR Power Electronics Interfaced Resources. 1, 2, 9, 10

PLL Phase-Locked Loop. 3, 7

PoC Point of Connection. 3

PPC Power Park Control. 3, 4

PV Photovoltaics. 1, 2, 5–7

RES Renewable Energy Sources. 1, 2, 6, 10

RMS Root-Mean-Square. 1

SMIB Single Machine Infinite Bus. 5, 9

SOC State Of Charge. 6, 8

TSO Transmission System Operator. 1–3, 10

UVRT Under-Voltage Ride-Through. 4

VDL Voltage Dependent Limitations. 8

VSC Voltage Source Converter. 8

WECC Western Electricity Coordinating Council. 1, 2, 5–7, 9–11

WPP Wind Power Plant. 1–4

WT Wind Turbine. 1, 3–5

WTG Wind Turbine Generator. 5–8

DOI 10.3384/ecp218383

Dynamic modelling of an Ammonia to Power application at high efficiency using a solid oxide fuel cell system

Michele Bolognese[1] Emanuele Martinelli[1] Luca Pratticò[1] Matteo Testi[1]

[1]Fondazione Bruno Kessler, Italy, mbolognese@fbk.eu

Abstract

Ammonia is a promising zero carbon and sustainable hydrogen carrier that can be used as a fuel in solid oxide fuel cells (SOFC) by offering advantages related to the ease of storage and the possibility of being used directly without an external reformer. In this study, a Modelica-based dynamic model of an 'Ammonia to Power' (A2P) system was developed by integrating ammonia decomposition kinetics, electrochemical reactions, all the system-level components and the main control loops. A novel Balance of Plant (BoP) configuration is proposed, featuring a five-way heat exchanger that recovers waste heat primarily using the fuel stream as the thermal energy vector instead of air. The model evaluates transient responses to operational perturbations, the behavior of the different control loops, and recirculation percentage rates to optimize system performance. Efficiency is calculated as the ratio of the power output from the SOFC to the power derived from the fresh ammonia line.

Keywords: SOFC, Ammonia, BoP, High temperature fuel cell

1 Introduction

The sustainable energy transition needs carbon-neutral solutions to meet the worldwide increasing energy demand (16 TW in 2010 to 23 TW in 2030 and 30 TW in 2050) while minimizing the environmental impacts (Dresselhaus and Thomas 2001). In this contest, green hydrogen produced from renewable energy sources (RES) plays a crucial role and is considered a potential alternative to fossil fuels. However, the establishment of a hydrogen economy faces substantial hurdles that must be addressed. Beyond the necessity of large-scale production of renewable-based green hydrogen, there are also long-standing challenges related to transportation, distribution, storage, and end-use of hydrogen that need a solution. Therefore, it is essential to develop innovative and effective strategies that can facilitate sustainable energy transition. In this perspective, the potential of utilizing ammonia as a hydrogen carrier for on-site power generation via ammonia decomposition is systematically discussed (Zhai et al. 2023). Hydrogen possesses a high gravimetric energy density of approximately 120 MJ/kg, surpassing conventional fuels like gasoline (44 MJ/kg) but the volumetric energy density of hydrogen is lower, so it requires larger vessels for storage and transportation. Different physical and material-based technologies have been explored for the dense storage of hydrogen but physical processes such as liquefaction and compression, which are used for storage and transportation, are energy intensive. For instance, hydrogen liquefaction consumes about 13.8 kWh/kg of liquid hydrogen, which corresponds to approximately 40% of hydrogen's energy content (Zhang et al. 2023). Similarly, hydrogen compression to high pressures, such as 700 bar, is essential for applications like fuel cell vehicles to achieve acceptable storage compactness. This process is energy-intensive; for instance, a study reported that using multiple pressure stages in refueling stations can reduce compressor power consumption by 17%, highlighting the significant energy demands of hydrogen compression. On the other hand, material-based storage, such as zeolites and metal hydrides, can provide a hydrogen storage capacity of up to 13 wt%., but these technologies often face challenges related to technological readiness levels (TRL). Higher energy densities are obtained with hydrogen storage in the form of chemicals such as methanol (12.5 wt %) or ammonia (17.65 wt%). Ammonia (NH_3) has emerged as a promising hydrogen carrier and carbon-free fuel, offering a unique combination of high hydrogen content, established global infrastructure, and compatibility with various energy conversion technologies. Historically, the ammonia synthesis via the Haber-Bosch (HB) process has been fundamental in worldwide agriculture and industry, by enabling the large-scale ammonia production via nitrogen and grey hydrogen produced via steam methane reforming, under high temperature and pressure operating conditions. The potential of ammonia as an energy carrier spans multiple applications, including internal combustion engines (ICE (Tornatore et al. 2022)), gas turbines (Pashchenko 2024a) , and fuel cells (Pashchenko 2024a; Qi et al. 2023; Pashchenko 2024b). Among these, solid oxide fuel cells (SOFCs) stand out for their ability to utilize ammonia directly as a fuel, bypassing the need for external hydrogen production or cracking processes. This direct utilization enables SOFC systems to achieve high power densities while producing only nitrogen and water as byproducts, aligning with the goals of a sustainable energy future (Dhawale et al. 2023). Several authors studied the configurations of direct ammonia SOFC system and their performance (Cinti et al. 2016; Nemati et al. 2025; Ayerbe et al. 2024). In their work, Cinti et al. have demon-

strated the improved efficiency of a direct ammonia SOFC system if compared to a hydrogen-based one. Nemati et al. have shown the importance of the anode off-gas recirculation and pre-cracking to improve the efficiency and reduce the impact of nitriding. The dynamic behavior of the A2P BoP is notably complex due to the interaction among its various functional zones, subcomponents, and control loops. This study addresses this challenge by developing a comprehensive dynamic model in Modelica language, serving as a prototyping tool that enables the analysis of overall system performance, including the direct ammonia solid oxide electrolyzer, the external cracker, and the waste heat recovery system under various operating scenarios. Additionally, the incorporation of advanced control loops and recirculation strategies offers new insights into the system-wide dynamics and operational stability, contributing novel understanding to the field of P2A plant integration and control

2 Methodology

Dynamic modeling is crucial to understanding and optimizing the behavior of an energy system balance of a plant under different operational conditions by simulating complex interactions between the system components. Dynamic modeling activities can also be a useful guide for the development of a control strategy to enhance overall system efficiency, flexibility, and reliability. Modern process modeling has evolved beyond simple material and energy balance calculations to become a strategic tool for managing and better understanding the potential of prototypes by gaining a competitive advantage (Ayerbe et al. 2024). Robust models enable rapid adaptation to market changes and serve as repositories of corporate knowledge. Models also capture valuable experimental data that aid in process optimization, redesign, and patent development. This creates a continuous improvement cycle in which each use enhances the value of the model, increasing the return on investment, and driving innovation. The Modelica language (Mattsson and Elmqvist 1997; Qiu et al. 2024) enables the simulation of complex systems such as the balance of plant (BoP) of hydrogen systems by using non-causal modeling, reusable components, and equation-based formulation. Non-causality allows the same equations to be used for forward and inverse kinematics, enhancing adaptability across engineering applications. In addition, the use of proper media for the fuel and air stream ensures accurate thermodynamic property calculations that are fundamental for calculating a heat and material balance that gives crucial information regarding the design of the overall system. In this study, one-dimensional equidistant discretizations is applied to each component of the BoP model. Furthermore, diffusion phenomena were neglected by simplifying the simulation.

2.1 BoP Model Description

This work presents the development of an 'Ammonia to Power' (A2P) plant model based on solid oxide fuel cell technology, which incorporates all other components of the system. In Figure 1, the functional Process and Instrumentation Diagram (P&ID) is presented:

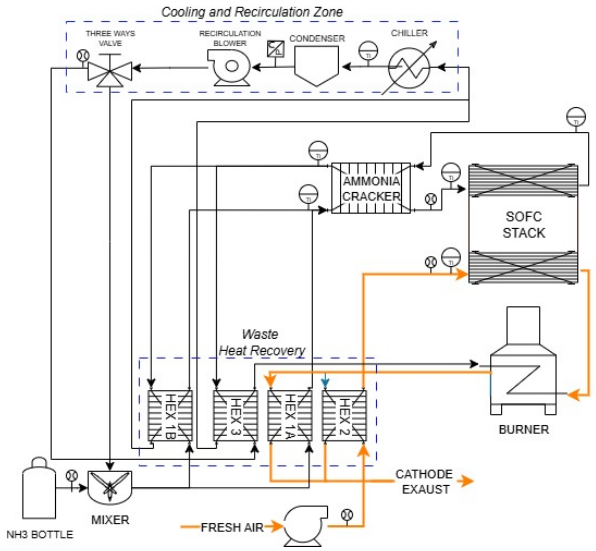


Figure 1. Functional Process and Instrumentation Diagram (P&ID) with fuel loop (black line) and air loop (orange line)

In the P&ID, the air flow loop is identified in orange, while the fuel flow loop appears in black. Each component is clearly labeled, and the regions associated with the waste heat recovery system, as well as the fuel cooling and recirculation circuits, are marked with dashed outlines for easy identification. To contribute to the development of an efficient and sustainable ammonia-based energy solution, this study aims to provide information on system design, operation, and control strategies. The key core of the system is represented by a solid oxide fuel cell stack, but particular attention was paid to the waste heat recovery and recirculation aspect of the anode off-gas within the AMON project. In Figure 2, the layout of the BoP model is presented using the Modelon Impact (Modelon 2025) interface.

From the layout in Figure 2, it is possible to distinguish the hot BOP that works at high temperature from the cold BOP starts from the condenser (block 6) and finishes downstream of the recirculation blower (block 9). Moreover, it is possible to also notice that other components of the Modelon library (joints and splits) were used to couple and split the fuel and air streams in the Hot BoP.

The BoP components in Figure 2 are numbered and described in the following.

1) Stack

The stack model is based on the Fuel Cell library (Andersson et al. 2011) by Modelon Impact (*Fuel Cell Library* 2025; Modelon 2025). The model was modified by implementing ammonia cracking in the anode channel. The stack model was validated by considering the polarization curves obtained in FBK test benches for other works within the AMON project. Moreover, to the stack an ac-

DOI 10.3384/ecp218395

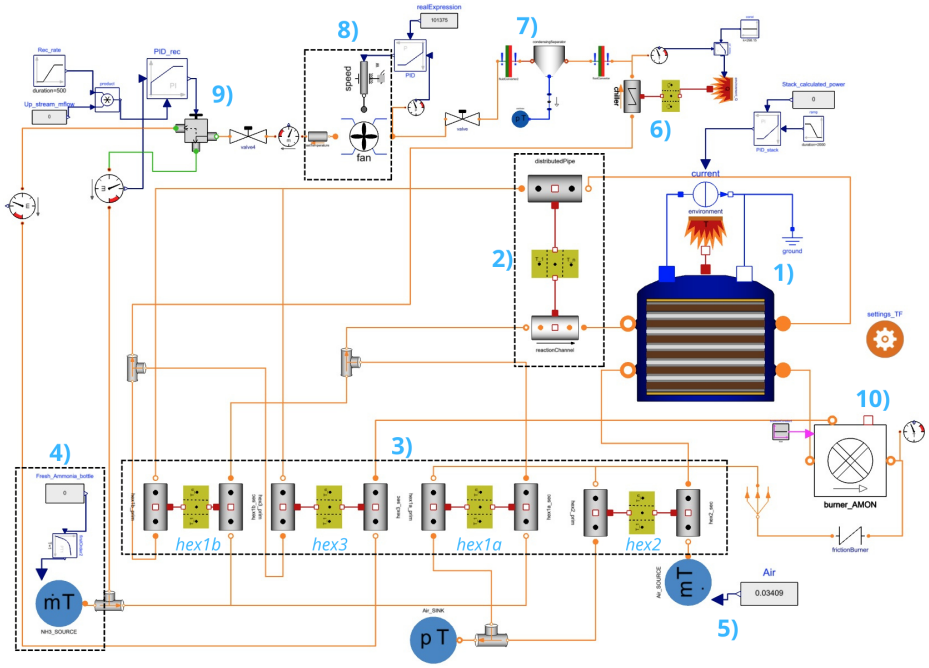


Figure 2. Layout of Ammonia to Power BoP model in Modelon Impact.

tive control loop based on a PID that controls the current delivered by the stack is applied in order to control the desired power for the different operating modes.

2) External Cracker

The External Cracker model is based on the use of the Modelon Standard Library by using the reaction channel and distributed pipe model that are connected to the metal wall model in counterflow mode.

3) Heat Exchanger Network

The model of the heat exchanger network is based on 4 heat exchangers that are calibrated according to the technical datasheet provided within the AMON project. In addition, several split and joint components of the Modelon library were used to modify the mass flow rates of the four different HEXs. In particular, air, which typically serves as the thermal carrier for the high-temperature balance of plant (BoP) components in SOC technologies, is used only 10% to preheat the recirculated fuel mixture (NH_3 and N_2), while the remaining portion is used to preheat the air entering the stack. Each heat exchanger (HEX) consists of a distributed pipe connected to a metal wall.

4) Fresh Ammonia Sourcer

This is an ideal ammonia source model at 25 °C and the ammonia mass flow rate required by the system is calculated as a function of the current delivered from the stack and the utilization of the fuel:

$$\dot{m}_{\mathrm{NH_3}} = \frac{n_{\mathrm{cell}} I}{2 U_{\mathrm{fuel}} F} \frac{2}{3} MM_{\mathrm{NH_3}} \tag{1}$$

where n_{cell} is the number of cells in the stack, I is the current, U_{fuel} is the fuel utilization (equal to 0.7), F is the Faraday constant and $MM_{\mathrm{NH_3}} = 0.017031\,\mathrm{kg/mol}$ is the molecular weight of ammonia.

5) Air Source

This is an ideal air source model at 25 °C with the mass flow rate defined as follows:

$$\dot{m}_{\mathrm{Air}} = \frac{n_{\mathrm{cell}} I}{4 U_{\mathrm{Air}} F} Y_{\mathrm{O_2}} MM_{\mathrm{Air}} \tag{2}$$

where U_{Air} is air utilization (equal to 0.11), $Y_{\mathrm{O_2}}$ is the oxygen content of the air (0.2094%) and $MM_{\mathrm{Air}} = 0.028965\,\mathrm{kg/mol}$ is the molecular weight of the air.

6) Chiller

The chiller is an ideal and simplified model of a simple pipe connected to a thermal power source that chills the Off Gas Anode (AOG) to 25 ° C using a specific PID component to modulate the cooling power required.

7) Condenser

The Condensing Separator model is a Modelon library component based on a condensing volume that efficiently separates water from a generic gas stream. A specific condensate medium is used by implementing two medium converters that can couple the medium used for the fuel and the condensate one.

8) Recirculation Blower

This is a simplified model of an axial fan in which the characteristics of the fan with respect to the volume flow rate V_{flow} and the power absorbed by the fan P_{fan}, have been implemented as follows:

Moreover, a time constant τ affects the equations that govern the mass flow rate (m_{flow}) and power consumption (P) as can be seen in the following equations:

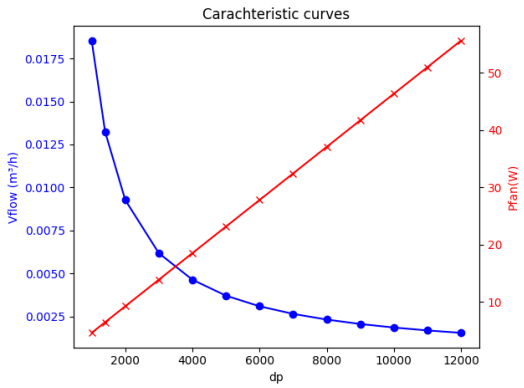


Figure 3. Characterization curves of the recirculation fan

$$\tau \frac{dm_{\text{flow}}}{dt} + m_{\text{flow}} = V_{\text{flow}_{\text{in}}} dA_{\text{inflow}} \tag{3}$$

$$\tau \frac{dP}{dt} + P = P_{\text{fan}_{\text{in}}} \tag{4}$$

where dA_{inflow} is the mass density calculated in the mass port upstream of the component.
The speed of the recirculation blower is controlled by a PID controller to maintain the upstream pressure higher than the environmental pressure during transitions between different operating modes.

9) 3-Ways recirculation valve

This three-way valve model is designed for single-phase medium applications and belongs to the ThermalPower 1.28 library of Modelon (Modelon 2025). The valve assumes turbulent, non-choked flow with static mass and energy balances and no actuator dynamics. By default, the flow characteristic is linear, but can be customized.

10) Ammonia Catalytic burner

The burner model is a fixed volume combustion chamber of the Modelon library. In this study, it was modified according to the A2P system by replacing fossil fuels with a mixture of hydrogen (H_2) and ammonia (NH_3). Energy calculations use the Lower Heating Value (LHV) of the H_2/NH_3 blend.

3 Main processes

In this section, the main processes implemented in the A2P dynamic model are described. A particular medium NASA-based of 5 elements (H_2, NH_3, H_2O, N_2, O_2) was created to implement Ammonia in the fuel composition.

3.1 Ammonia cracking Processes

For ammonia cracking, the studies by Kishimoto et al. (Kishimoto et al. 2017) and Lucentini et al. (Lucentini et al. 2021) were considered. The model of Kishimoto et al. was applied to the cracking of ammonia within the anode channel of the stack (block 1), as they developed an experimentally validated ammonia decomposition reaction (R_{kish}) for a Ni-YSZ anode. The reaction model developed by Lucentini et al. (R_{luc}) for the decomposition of ammonia over Ni-Ru supported on CeO_2 was applied to the external cracker reaction channel (block 2). The following equations present the ammonia degradation reaction rates for both the stack and the external reformer:

$$R_{\text{kish}} = S_{\text{Ni-pore}} A^{\text{Ni-pore}} \exp\left(-\frac{E}{RT}\right) p_{\text{NH}_3}^{a} \left(p_{\text{H}_2} + c\right)^{b} \tag{5}$$

$$R_{\text{luc}} = \frac{k K_1 C_{\text{NH}_3}}{\left(1 + K_1 C_{\text{NH}_3} + \sqrt{\frac{C_{\text{H}_2}}{K_4}}\right)^2} \tag{6}$$

In this study, the materials constituting the storage cell and the cracker catalyst are not yet defined, so the same materials of the reference papers were considered by using the same values of the coefficients for the two reaction rates. Subsequently, the ammonia decomposition reaction of Lucentini et al. is calibrated considering the coefficient values valid for $5Ni1Ru/CeO_2$ by implementing the geometric characteristics of the cracker presented in Table 1, which refers to the equivalent diameter of the pipe having the same cross section as the rectangular channel of the cracker plates.

Table 1. Technical specifications of the external cracker

Parameters	Value	Unit	Description
n_{plates}	34	–	Number of plates
A_{heat}	0.4	m^2	Heat exchange area
length	20.7	cm	Total length
height	13.7	cm	Total height
width	7.7	cm	Total width
d_{eq}	2.5	cm	Equivalent diameter
ε_{bed}	0.245	–	Porosity of the bed
f_v	0.196	–	Catalyst volume fraction

In this study, the external cracker was considered as a fixed-bed reactor; thus, the concentration of ammonia and hydrogen is calculated only over the void volume. The amount of catalyst needed within the reactor bed (g_{cat} [g]) depends on the temperature and the conversion percentage, as shown in Figure 4.

3.2 Electrochemical Process

In this study, the structure of the stack model is based on the Modelon Fuel Cell Library, which utilizes the coupling between the anode channel, the elementary cell, and the cathode channel. Within the elementary cell, a simplified equivalent electrical circuit was used to characterize the polarization curves of the simulation by comparing them with the experimental data obtained at the FBK test bench using a short stack of 6 cells provided by Solydera. More

DOI
10.3384/ecp218395

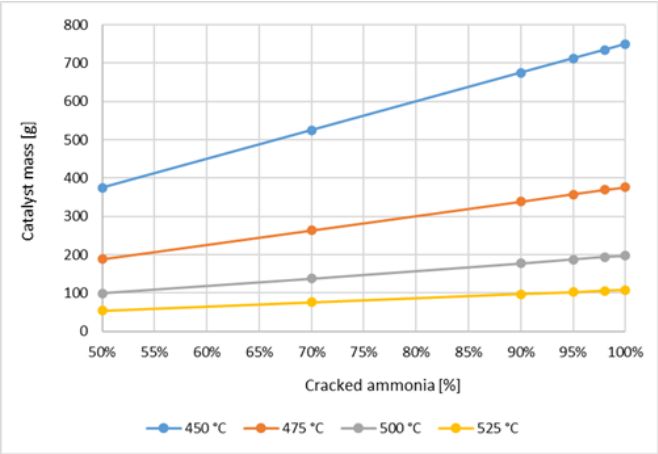


Figure 4. Ammonia cracking percentage as a function of temperature and mass of the catalyst.

details of these validation activities are presented in another FBK study at the Modelica Conference.

Taking into account the desired rangeability of the operating conditions of the A2P system within the AMON project, the cell voltage in the cell model was calculated by considering only ohmic losses.

$$V_{\text{cell}} = E_0 - \text{ASR}(T)\, j_{\text{cell}} \tag{7}$$

where E_0 is the Nernst voltage, j_{cell} is the current density, and ASR is the area specific resistance calculated as follows:

$$E_0 = \frac{1}{n_e F}\left(\Delta g_{\text{reaction}} + RT_{\text{cell}} \ln\left(\frac{p_{H_2O}}{p_{H_2} p_{O_2}^{0.5}}\right)\right) \tag{8}$$

$$\text{ASR}(T) = \text{ASR}_0 + \exp\left[\frac{E_a}{R}\left(\frac{1}{T_{\text{cell}}} - \frac{1}{T_0}\right)\right] \tag{9}$$

$$j_{\text{cell}} = \frac{I}{A_{\text{cell}}} \tag{10}$$

For this work, a value of $\text{ASR}_0 = 0.34 \times 10^{-4}\,\Omega/\text{cm}^2$ and an activation energy $E_a = 24.8 \times 10^3\,\text{kJ/mol}$ were used with a reference temperature $T_0 = 600\,^\circ\text{C}$, obtaining a relative error less than 3%, consistent with previous work on SOC technologies performed in other EU projects.

3.3 Condensation Process

The condensation process follows a first-order differential equation governed by the condensation time constant τ_{sat}. This parameter controls the rate at which water vapor condenses into liquid and is defined as follows:

$$\tau_{\text{sat}} \frac{d}{dt}\dot{m}_{\text{water}} + \dot{m}_{\text{water}} = \dot{m}_{\text{feed}}(X_{\text{feed}} - X_{\text{sat}}) + k_{\text{molar}} \max(\varepsilon - \dot{m}_{\text{water}}, y_{H_2O} - y_{\text{sat}}) \tag{11}$$

Table 2. Stack Technical Parameters

Parameters	Value
Active Area [cm^2]	320
Number of cells	80
Max Current [A]	128
Fuel Utilization [%]	70
Max Power [kW]	8

where

- X_{feed} is the water mass fraction in the inlet stream,
- X_{sat} is the saturated water mass fraction, calculated as:

$$X_{\text{sat}} = \frac{y_{\text{sat}}}{1 - y_{\text{sat}}} \cdot k_{\text{molar}} \cdot (1 - X_{\text{feed}}) \tag{12}$$

where y_{sat} is the saturated mole fraction of water vapor calculated as a function of temperature T and pressure p:

$$y_{\text{sat}} = \frac{p_{\text{sat}}(T)}{p} \tag{13}$$

Here, $p_{\text{sat}}(T)$ the saturation pressure is calculated as a function of temperature and is a medium parameter.

The factor of molar ratio k_{molar} is calculated as the ratio of the molar fraction of water to the molar fraction of gas without water:

$$k_{\text{molar}} = \frac{MM_{H_2O}}{MM_{H_2\text{dry_gas}}} \tag{14}$$

3.4 Combustion of H_2 / NH_3 mixture

In the burner model, the following oxidation reactions for H_2 and NH_3 have been integrated into the combustion reaction by implementing a specific stoichiometric matrix:

$$H_2 + \frac{1}{2}O_2 \rightarrow 2H_2O \tag{15}$$

$$NH_3 + \frac{3}{4}O_2 \rightarrow \frac{3}{2}H_2O + \frac{1}{2}N_2 \tag{16}$$

These reactions assume ideal conditions under the assumption of complete combustion, where all reactants are fully converted into products, primarily water vapor and nitrogen gas, with no unburned residues or intermediate species.

3.5 Heat Recovery System

In this study, particular attention was paid to the heat recovery system of the thermal energy coming from the high temperature SOFC and the additional burner. The heat heat recovery system is composed of 4 heat exchangers:

- **hex1a** (fuel-fuel): preheating of the recirculated fuel and fresh ammonia mixture by using the 50% of mass flow rate of anode fuel from the stack after the heating of the external cracker
- **hex1b** (fuel-fuel): preheating of the non-recirculated fuel that feeds the burner by using the 50% of mass flow rate of anode fuel from the stack after the heating of the external cracker
- **hex2** (fuel-air): preheating of the recirculated fuel and fresh ammonia mixture by using the 10% of exhaust gas mass flow rate from the burner
- **hex3** (air-air): preheating of fresh air by using the 90% of exhaust gas mass flow rate from the burner

All the HEXs mentioned above are in counterflow configuration; the specific geometries were implemented coherently with the design data of the AMON project. In the following figure is possible to see the trends of hot and cold fuels vs the longitudinal distance of each heat exchanger.

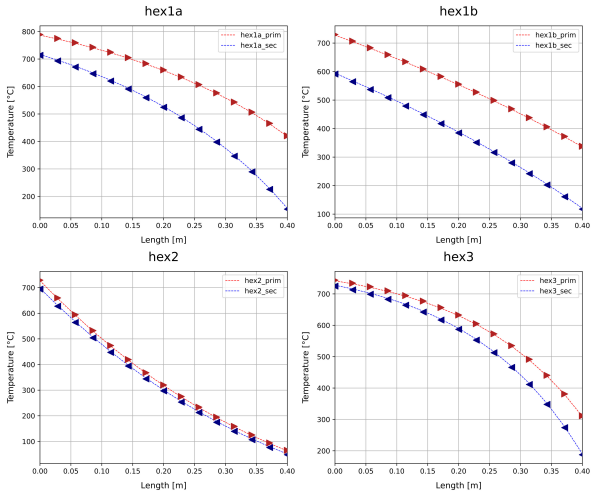


Figure 5. Trends of temperature of hot and cold sides vs distance for each HEX.

Figure 5 shows the temperature profiles along the length of four heat exchanger configurations (hex1a, hex1b, hex2, hex3), illustrating heat transfer between primary and secondary fluids. The decreasing primary and increasing secondary temperature trends suggest counterflow or parallel flow behavior, influenced by variations in the design of the heat exchanger and operating conditions.

4 Results

In this section, the main results of the BoP model are presented. An initial simulation of 200,000 seconds was performed, with an elapsed time of approximately 14 to 15 minutes, to verify the behavior of the external reformer. Setting a fuel recirculation rate at 88 % and a power at 8 kW (nominal condition). In Figure 6, it is possible to see that during the simulation time, the NH_3 conversion rate of the external reformer and the related trends of NH_3 molar fraction and inlet/outlet fuel temperatures reaches converge and the outlet NH_3 molar fraction is around 3% in a consistent way with the decision-making decisions made during the design phase of the system for the AMON project.

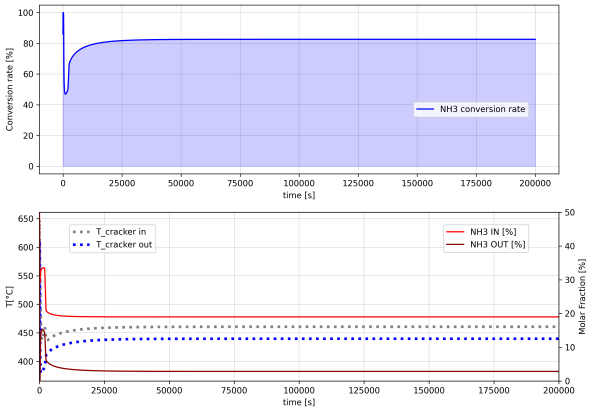


Figure 6. External cracker behaviors at nominal condition with 88 % of recirculation rate. Ammonia conversion rate [%] (upper plot), inlet and outlet cracker fuel temperature, and NH_3 molar fraction (lower plot)

Eventually, considering the same simulation, the relative pressure profiles of air (blue line) and fuel (red line) are presented in Figure 6.

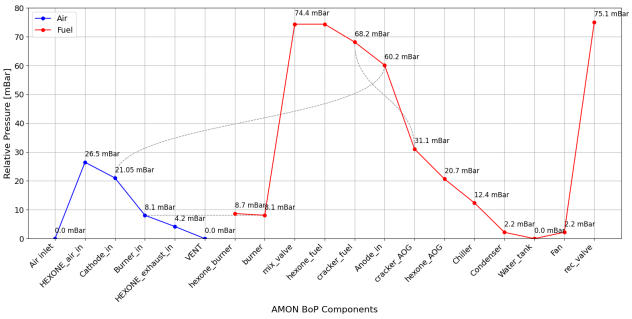


Figure 7. Pressure drops trends for all the components of the BoP model [air channel in blue and fuel channel in red]

It can be observed that the fuel pressure profile is higher than that of air, with a recorded pressure difference of approximately 39 mbar under nominal conditions (8 kW) in this simulation. This configuration ensures that the anode pressure at the stack inlet is higher than the cathode stack pressure, which is a crucial safety measure to prevent backflow or mixing risks. The fuel loop achieves a peak pressure of 74.4 mbar through fan-driven circulation, with a pre-recirculation pressure drop of approximately 73 mbar. This results in a recirculation rate of 88%, while 12% is directed to the burner. On the other hand, the air pressure profile achieves a maximum pressure of 26.5 mbar upstream of the heat recovery system. Afterward, a short sensitivity analysis of the fuel mass flow rate recircu-

DOI
10.3384/ecp218395

lation rate was performed to select the best configuration. The fuel that is recirculated after the cracking, electrolysis process, chilling, and condensation is a mixture mainly composed of N_2. In Table 4, four different cases with a recirculation rate of 80, 85, 88, and 90 % have been evaluated.

Considering the design decisions made during the design phase of the system, the 88% fuel recirculation rate was chosen as the optimal solution because it allows the recirculation of the blower operation to be respected in terms of the maximum flow rate and achieves an efficiency of 73.13%, which is above the 70% target of the AMON project.

Finally, in Figure 8, the transitions between full load (8 kW) and partial mode (2.4 kW) were simulated in order to verify the behaviors of the control loops dedicated to the modulation of current for the power control of the stack, the control of the AOG recirculation (CL_fan) by modulating the rpm [Hz] of the blower, and the control of opening/closing of the 3-way valve (CL_3ways). Moreover, in the same figure it is possible to see the trend of the overall efficiency of the system, which is calculated as follows:

$$\eta_{\text{BoP}} = \frac{P_{\text{stack}}}{LHV_{NH_3} \cdot \dot{m}_{NH_3,in}} \tag{17}$$

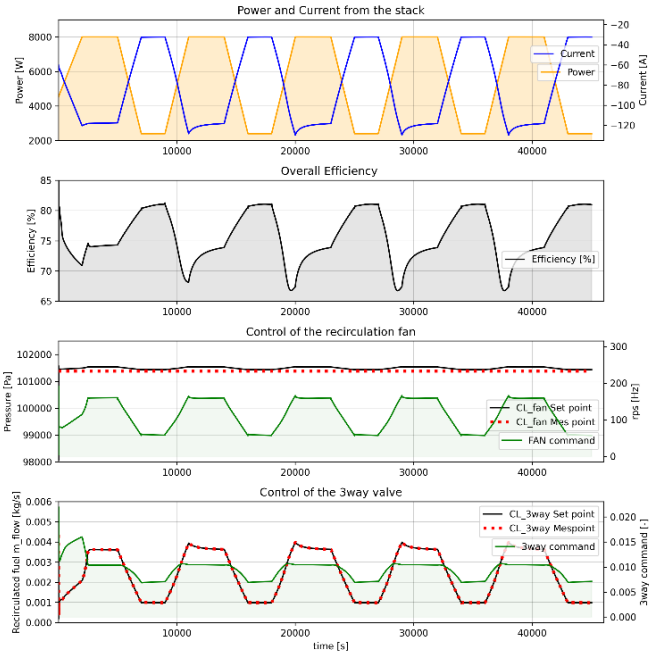


Figure 8. Dynamic simulation results of Full to Partial Mode transitions: Power and Current of the stack (upper), Overall Efficiency of the system (medium-high), Control of recirculation fan (medium-low), and Control of the 3-way valve (lower plot).

It is possible to see that the control of the stack power by modulating the current is quite stable. However, during the transition from partial to full operation, the overall efficiency of the system experiences a transient decrease. This initial drop in efficiency occurs only in the first moments after the switch, but subsequently stabilizes within the typical range associated with full load operation, which was previously calculated as approximately 72–74%.

Moreover, the CL_3ways control loop governing the 3-way valve demonstrates consistent stability, while the CL_fan control loop for the recirculation fan shows minor deviations from the set point (around 200 Pa). These deviations are sufficiently small to be considered negligible, so the control loop can be considered to function correctly by maintaining the required pressure set point. Overall, despite some small adjustments required, the control strategy implemented in the model exhibits robust stability and good dynamic response, ensuring reliable operation under varying load conditions.

5 Conclusion

This work presents a description and comprehensive analysis of a Modelica-based dynamic model of an Ammonia2Power (A2P) system utilizing the coupling of a direct ammonia solid oxide fuel cell and an external reformer. With the integration of detailed ammonia decomposition with different catalysts for the external reformer and the stack, the electrochemical processes, and an innovative BoP addressing enhanced waste heat recovery, the model enables in-depth analysis of system behavior under different operating modes. Two different simulations were performed, one addressed to calculate heat and material balance (after 200000 s of stationary behavior) and one dedicated to the transitions between full and partial modes. Furthermore, the simulation results demonstrate that the proposed system can achieve a high ammonia conversion rate and efficient heat recovery, with an AOG recirculation rate of 88% and stable operation in full (8 kW) and partial (2.4 kW) modes. The pressure profile of both fuel and air streams was calculated, ensuring safe operation by maintaining anode pressure higher than the cathode pressure, minimizing backflow risks. Finally, the dynamic Modelica-based modeling approach provides valuable insights into the transient behavior of the system, the impact of AOF recirculation and heat integration on the overall efficiency of the system, and the effectiveness of control strategies. This study confirms the technical feasibility and potential of ammonia as a vector and hydrogen carrier for direct utilization in SOFC BOP. The model acts as a robust tool to guide the design phase of the prototype and the optimization of future A2P systems, supporting the transition to sustainable energy and carbon-neutral solutions. Future work will focus on the implementation of a proper state machine for the control strategy, further refinements of control loops, and verification of emergency scenarios to evaluate system safety in order to enhance system flexibility and reliability under real-world operating conditions.

Table 3. Sensitivity Analysis for Different Recirculation Rates (80, 85, 88, 90%)

	Case 1	**Case 2**	**Case 3**	**Case 4**
Recirculation Fraction	80%	85%	88%	90%
m_{flow} AOG [kg/h]	12.492	15.12	17.856	20.628
T_{AOG} [°C]	230	239	241	244
Efficiency [%]	71.20	73.30	74.40	75.01
NH_3 fresh m_{flow} [kg/h]	2.16	2.11	2.08	2.06
Δp recirculation fan [mbar]	46.96	56.99	72.94	95.49
NH_3 cracking rate [%]	93.91	92.94	83.00	84.51
$X_{\text{NH}_3,\text{cracker out}}$ [%]	3.95	3.23	2.85	2.48
$X_{\text{NH}_3,\text{stack out}}$ [%]	1.6e−6	0.002	0.05	0.17
Air required [kg/s]	0.0310	0.0308	0.0306	0.0305
Chiller [kW]	1.007	1.219	1.407	1.603

Acknowledgements

The project is supported by the Clean Hydrogen Partnership and its members, Hydrogen Europe and Hydrogen Europe Research, under Grant Agreement No. 101101521.

References

Andersson, Daniel, &Aring, Erik Berg, Jonas Eborn, Jinliang Yuan, Sund&eacute, and Bengt N (2011-06). "Dynamic modeling of a solid oxide fuel cell system in Modelica". en. In: URL: https://ep.liu.se/en/conference-article.aspx?series=ecp&issue=63&Article_No=66 (visited on 2025-04-15).

Ayerbe, Cécile, Caroline Boulos, and Francesco Castellaneta (2024-09). "Navigating protection mechanisms and innovation models: A literature-based configurational framework of intellectual property strategies". In: *Technovation* 137, p. 103101. ISSN: 0166-4972. DOI: 10.1016/j.technovation.2024.103101. URL: https://www.sciencedirect.com/science/article/pii/S0166497224001512 (visited on 2025-04-15).

Cinti, G., G. Discepoli, E. Sisani, and U. Desideri (2016). "SOFC operating with ammonia: Stack test and system analysis". In: *International Journal of Hydrogen Energy* 41.31, pp. 13583–13590. ISSN: 0360-3199. DOI: https://doi.org/10.1016/j.ijhydene.2016.06.070. URL: https://www.sciencedirect.com/science/article/pii/S036031991630595X.

Dhawale, Dattatray S., Saheli Biswas, Gurpreet Kaur, and Sarbjit Giddey (2023). "Challenges and advancement in direct ammonia solid oxide fuel cells: a review". en. In: *Inorganic Chemistry Frontiers* 10.21, pp. 6176–6192. ISSN: 2052-1553. DOI: 10.1039/D3QI01557B. URL: https://xlink.rsc.org/DOI=D3QI01557B (visited on 2025-04-15).

Dresselhaus, M. S. and I. L. Thomas (2001-11). "Alternative energy technologies". en. In: *Nature* 414.6861, pp. 332–337. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/35104599. URL: https://www.nature.com/articles/35104599 (visited on 2025-04-15).

Fuel Cell Library (2025). en-US. Publication Title: Modelon. URL: https://modelon.com/library/fuel-cell-library/ (visited on 2025-04-15).

Kishimoto, Masashi, Naoto Furukawa, Tatsuya Kume, Hiroshi Iwai, and Hideo Yoshida (2017-01). "Formulation of ammonia decomposition rate in Ni-YSZ anode of solid oxide fuel cells". In: *International Journal of Hydrogen Energy* 42.4, pp. 2370–2380. ISSN: 0360-3199. DOI: 10.1016/j.ijhydene.2016.11.183. URL: https://www.sciencedirect.com/science/article/pii/S0360319916335169 (visited on 2025-04-15).

Lucentini, Ilaria, Germán García Colli, Carlos D. Luzi, Isabel Serrano, Osvaldo M. Martínez, and Jordi Llorca (2021-06). "Catalytic ammonia decomposition over Ni-Ru supported on CeO2 for hydrogen production: Effect of metal loading and kinetic analysis". In: *Applied Catalysis B: Environmental* 286, p. 119896. ISSN: 0926-3373. DOI: 10.1016/j.apcatb.2021.119896. URL: https://www.sciencedirect.com/science/article/pii/S0926337321000229 (visited on 2025-04-15).

Mattsson, Sven Erik and Hilding Elmqvist (1997-04). "Modelica - An International Effort to Design the Next Generation Modeling Language". en. In: *IFAC Proceedings Volumes* 30.4, pp. 151–155. ISSN: 14746670. DOI: 10.1016/S1474-6670(17)43628-7. URL: https://linkinghub.elsevier.com/retrieve/pii/S1474667017436287 (visited on 2025-04-15).

Modelon (2025). *Modelon Impact*. https://modelon.com/modelon-impact/. Accessed: 2025-07-18.

Nemati, Arash, Hossein Nami, Javid Beyrami, Rafael Nogueira Nakashima, and Henrik Lund Frandsen (2025). "Efficient and durable system design for ammonia-fueled solid oxide fuel cells using multiscale multiphysics modeling approach". In: *Fuel* 392, p. 134837. ISSN: 0016-2361. DOI: https://doi.org/10.1016/j.fuel.2025.134837. URL: https://www.sciencedirect.com/science/article/pii/S0016236125005617.

Pashchenko, Dmitry (2024a-03). "Ammonia fired gas turbines: Recent advances and future perspectives". In: *Energy* 290, p. 130275. ISSN: 0360-5442. DOI: 10.1016/j.energy.2024.130275. URL: https://www.sciencedirect.com/science/article/pii/S036054422400046X (visited on 2025-04-15).

Pashchenko, Dmitry (2024b-03). "Ammonia fired gas turbines: Recent advances and future perspectives". In: *Energy* 290, p. 130275. ISSN: 0360-5442. DOI: 10.1016/j.energy.2024.130275. URL: https://www.sciencedirect.com/science/article/pii/S036054422400046X (visited on 2025-04-15).

Qi, Yunliang, Wei Liu, Shang Liu, Wei Wang, Yue Peng, and Zhi Wang (2023-10). "A review on ammonia-hydrogen fueled internal combustion engines". In: *eTransportation* 18, p. 100288. ISSN: 2590-1168. DOI: 10.1016/j.etran.2023.100288. URL: https://www.sciencedirect.com/science/article/pii/S2590116823000632 (visited on 2025-04-15).

Qiu, Kaiying, Junlu Yang, Zhi Gao, and Fusuo Xu (2024-04). "A review of Modelica language in building and energy: Development, applications, and future prospect". In: *Energy and*

Buildings 308, p. 113998. ISSN: 0378-7788. DOI: 10.1016/j.enbuild.2024.113998. URL: https://www.sciencedirect.com/science/article/pii/S0378778824001142 (visited on 2025-04-15).

Tornatore, Cinzia, Luca Marchitto, Pino Sabia, and Mara De Joannon (2022-07). "Ammonia as Green Fuel in Internal Combustion Engines: State-of-the-Art and Future Perspectives". English. In: *Frontiers in Mechanical Engineering* 8. ISSN: 2297-3079. DOI: 10.3389/fmech.2022.944201. URL: https://www.frontiersin.orghttps://www.frontiersin.org/journals/mechanical-engineering/articles/10.3389/fmech.2022.944201/full (visited on 2025-04-15).

Zhai, Lingling, Shizhen Liu, and Zhonghua Xiang (2023). "Ammonia as a carbon-free hydrogen carrier for fuel cells: a perspective". en. In: *Industrial Chemistry & Materials* 1.3, pp. 332–342. ISSN: 2755-2608, 2755-2500. DOI: 10.1039/D3IM00036B. URL: https://xlink.rsc.org/?DOI=D3IM00036B (visited on 2025-04-15).

Zhang, Tongtong, Joao Uratani, Yixuan Huang, Lejin Xu, Steve Griffiths, and Yulong Ding (2023-04). "Hydrogen liquefaction and storage: Recent progress and perspectives". en. In: *Renewable and Sustainable Energy Reviews* 176, p. 113204. ISSN: 13640321. DOI: 10.1016/j.rser.2023.113204. URL: https://linkinghub.elsevier.com/retrieve/pii/S1364032123000606 (visited on 2025-04-15).

DOI
10.3384/ecp218395

Direct Ammonia Solid Oxide Fuel Cell Stack: Modelling and Experimental Validation

Emanuele Martinelli[1] Michele Bolognese[1] Nirmala Nirmala[2] Narges Ataollahi[3] Matteo Testi[1]

[1]Fondazione Bruno Kessler, Research Center, Italy, {emmartinelli,mbolognese,testi}@fbk.eu
[2]Modelon Deutschland GmbH, Hamburg, Germany, nirmala.nirmala@modelon.com
[3]Department of Civil, Environmental and Mechanical Engineering, University of Trento, Italy, narges.ataollahi@unitn.it

Abstract

Interest in ammonia as an energy carrier is growing due to its superior storage and transport properties compared to hydrogen. The objective of this work is to construct a useful tool for predicting the behavior of a solid oxide fuel cell (SOFC) stack fed directly with ammonia. This configuration is particularly interesting because the internal cracking of ammonia eliminates the need for an external cracker, thus reducing the overall cost of the system. The ammonia decomposition reaction was implemented in the anode channel of the stack and calibrated against literature results. The model was then validated in the ohmic region only by calculating the area specific resistance (ASR) and comparing the results with experimental data collected at the Bruno Kessler Foundation (FBK) laboratory. This SOFC model can therefore be used as a starting point for the analysis of a scale-up application.

Keywords: Solid oxide fuel cell, High temperature fuel cell, Internal ammonia cracking, Experimental validation

1 Introduction

With the European Union setting a target of zero greenhouse gas emissions by 2050 (EuropeanCommission 2022), the need for sustainable, low-impact solutions for energy generation has become critical. According to the 1.5 °C scenario outlined by the International Renewable Energy Agency (IRENA 2024), hydrogen will account for up to 14% of total final energy consumption. It will also play a central role in the decarbonisation of hard-to-decarbonise industries (e.g., steel, chemicals and refineries), in the storage of excess renewable energy to provide flexibility to the grid, and in supporting long-distance transport directly or via hydrogen carriers. Among the potential hydrogen carriers, ammonia is a promising option due to its carbon-free nature and high hydrogen storage capacity; it offers favourable gravimetric (17.88 %wt) and volumetric (10.7 kg_{H_2} per 100 l) hydrogen densities, and, in its liquid form, it achieves a volumetric energy density of 12.9 MJ/l, surpassing that of liquid hydrogen at 8.6 MJ/l. The ammonia liquefaction process requires less extreme conditions, achieved at -33.3 °C at atmospheric pressure or about 10 atm at ambient temperature (Wang et al. 2022), making ammonia a better solution for long distance energy transport. It is also the second most produced and traded chemical in the world, 70% of which is used as a base for fertilisers and the remainder as an input for industrial processes, such as a refrigerant or in the manufacture of synthetic fibres, plastics and explosives (IEA 2021).

Fuel cells are electrochemical devices that convert chemical energy directly into electrical energy, without the need for combustion or intermediate conversion into mechanical energy; this makes them more efficient than traditional combustion generators. The basic working principle of fuel cells is the electrochemical reaction between a fuel and an oxidant, typically hydrogen and oxygen, respectively. In a solid oxide fuel cell (SOFC), the oxygen supplied to the cathode is reduced to form oxygen ions (O^{2-}), which then flow through the electrolyte to the anode. There, the O^{2-} ions combines with hydrogen to form water, releasing two electrons that will flow to the cathode through an external electrical circuit (reaction 1). The hydrogen required at the anode can be supplied directly or can be produced by the cracking of more complex fuels, such as methane and ammonia. The primary advantage of SOFCs is that they operate at high temperatures; this creates an environment in which complex fuels (with ammonia being the fuel of choice in this particular case study) spontaneously decompose at the anode electrolyte, releasing hydrogen directly into the anode channel (see the configuration on the right-hand side of Figure 1). If an external fuel processor were needed to extract hydrogen from the fuel, the system's complexity would increase, and the costs would rise concomitantly.

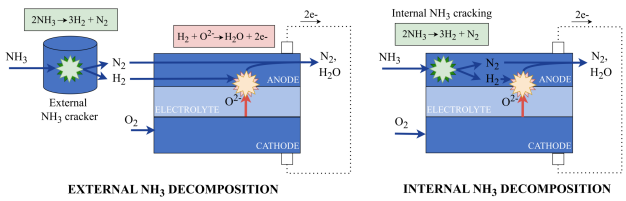


Figure 1. Ammonia decomposition in different configuration (indirect on the left, direct on the right) of a solid oxide fuel cell system.

In the present case study, a model of a short SOFC stack operating at elevated temperatures and fed directly with ammonia was developed and validated; this configuration is of particular interest because it allows the direct decomposition of ammonia in the anode channel, according to the endothermic reaction 2 ($\Delta H = 46.18\ kJ/mol$), without the need for an external cracker as required in indirect ammonia fuel cells (Lyu et al. 2023).

$$H_2 + O^{2-} \rightarrow H_2O + 2e^- \tag{1}$$

$$NH_3 \rightarrow \frac{3}{2}H_2 + \frac{1}{2}N_2 \tag{2}$$

Different equations were proposed to quantify the ammonia conversion performance over different catalysts (Wan et al. 2021; Zheng et al. 2023); Kishimoto et al. have developed and validated Equation 3, on which is presented the ammonia decomposition model in subsection 2.1 (Kishimoto et al. 2017). This reaction model was then implemented in the anode channel of the substack to obtain the final direct ammonia stack model. The stack model was then validated with experimental data collected on a test bench provided by SolydEra S.p.A. at the Fondazione Bruno Kessler laboratory; the stack consists of 6 cells connected in series with an active surface of 80 cm^2. The values of the operating parameters have been obtained from the experimental setup, while, for reasons of confidentiality, the material structure and the geometric organisation of the stack used in the model have been taken from literature works. To determine the applicability of the model and verify its robustness, the discrepancy between the experimental and simulation results was calculated for each test described in subsection 2.4.

This work focuses on understanding ammonia-based energy conversion and also provides a practical tool for analysing the operation of a SOFC stack that does not require the use of an expensive external cracker. This tool can also be used for future scale-up applications within a complex Balance of Plant (BoP).

2 Methods

The methodology adopted in this work is investigated using the Modelica language and Modelon Impact as tool. Modelica is an open-source language for modelling complex systems covering multi-domains such as mechanical, electrical, thermal and control. Modelon Impact is a cloud-native Modelica modelling and simulation platform, which offers a suite of libraries with preconfigured and validated components to help users get started in building their systems. In this work, building blocks from Modelon's Fuel Cell Library (FCL) have been employed. The FCL is primarily used for modelling, simulation, analysis, and control of fuel cell design and operation.

To build a direct ammonia solid oxide fuel cell (SOFC) stack model, it is first necessary to create a suitable ammonia medium and calibrate the ammonia decomposition reaction model. This reaction model is then implemented

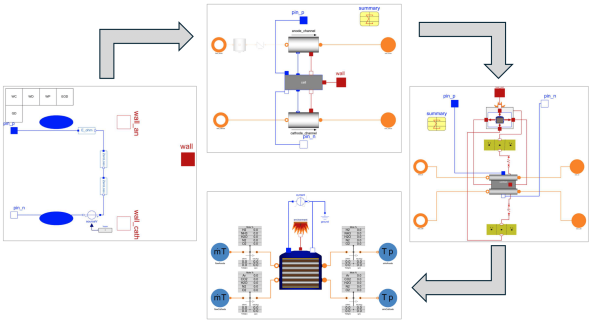


Figure 2. Cell, substack, stack and stack simulation models. Arrows indicate the dependence of nested models, from innermost to outermost models.

in the anode channel of the substack model, which also comprises the cell and cathode channel models. The cell model calculates the cell voltage by estimating the area specific resistance $ASR\ [\Omega/cm^2]$; the values of the parameters used in Equation 10 are found experimentally. The substack model is encapsulated within the stack model, where the heat loss characteristics are defined. Finally, a stack simulation model is created to define the boundary conditions from the experimental setup and to validate the stack model. The nested structure used for the stack validation is shown in Figure 2.

2.1 Ammonia Decomposition Reaction

A medium containing ammonia (H_2, NH_3, H_2O, N_2 and O_2) was created with thermodynamic properties taken from the Modelon NASA database (McBride, Zehe, and Gordon 2002) and compared to the NIST database (NIST 2023) for verification. An ammonia decomposition model was then subsequently created and validated with the reaction channel model shown in Figure 3.

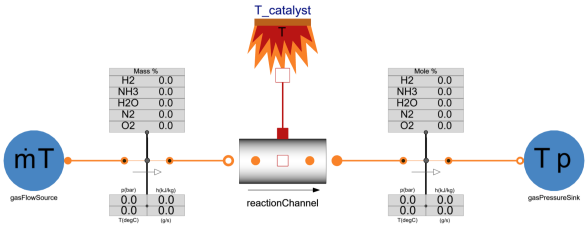


Figure 3. Discretised reaction channel model created to calibrate the ammonia decomposition reaction.

The ammonia decomposition reaction model, embedded within the reaction channel, was built on the basis of the work of Kishimoto et al., who validated Equation 3 against experimental results varying the catalyst temperature, ammonia mass flow rate or hydrogen mole fraction in the input fuel (Kishimoto et al. 2017).

$$R_{dec} = CE\ S_{Ni-pore}\ A^{Ni-pore}\ \exp\left(-\frac{E}{RT}\right) \cdot \cdot \left(p_{NH_3}\right)^a \left(p_{H_2} + c\right)^b \tag{3}$$

DOI 10.3384/ecp218405

For the reaction calibration, the unit volume reaction rate R_{dec} $[mol/m^3 s]$ was calibrated with the conversion efficiency parameter CE $[-]$. The other coefficients were chosen according to the calibration performed by (Kishimoto et al. 2017), assuming the same material properties as the Ni-YSZ catalyst used in the validation process; this material is one of the most commonly used in commercially available SOFC stacks, which makes it relevant to the case study. To obtain the decomposition reaction rate in $[mol/s]$, R_{dec} is multiplied by the catalyst volume V_{cat} $[m^3]$, calculated as $V_{cat} = V_{ch} \cdot f_{cat}$, where f_{cat} $[-]$ is the ratio of catalyst volume to channel volume. As the ammonia decomposition reaction is endothermic, (Kishimoto et al. 2017) also provides an estimate of the heat flow Q_{reac} $[W]$ withdrawn from the reaction with Equation 4.

$$Q_{reac} = \Delta H \, R_{dec} \, V_{cat} \tag{4}$$

The change in enthalpy ΔH $[J/mol]$ caused by the decomposition reaction is evaluated as $\Delta H = -40265.95 - 24.23214T + 0.00946T^2$, where T $[K]$ is the fluid temperature within the channel for each discretization.

The geometric and operational parameters considered for the reaction channel model are given in Table 1. The assumptions that were considered during the modelling design phase are reported in the following list:

Table 1. Geometrical and operating parameters adopted in the reaction calibration model.

Var.	Value	Unit	Description
N	10	-	discretization number
m_{flow}	1.266e-6	kg/s	NH_3 mass flow
p	101325	Pa	system pressure
T_{cat}	600, 650, 700	°C	catalyst temperature
D	8.3	mm	channel diameter
L	10, 20, 40	mm	catalyst length
f_{cat}	0.01	-	catalyst fraction
ε	2.5e-5	m	pipe roughness

- the reaction takes place only within the catalyst volume V_{cat}, which is considered fully active, and at a constant temperature T_{cat} $[°C]$ along the channel.
- the ammonia mass flow m_{flow} $[kg/s]$ is calculated from the volume flow V_{flow} $[ml/min]$ under normal conditions using the ideal gas law;
- D $[m]$ is the equivalent diameter, calculated from the cross-section of the rectangular channel;
- the pressure drop dp $[Pa]$ along the channel is calculated for all flow regimes with Equation 5, where the friction factor λ is found from the Reynolds number Re and the absolute surface roughness ε;
- the heat transfer coefficient α $[W/m^2K]$ is calculated over the entire Reynolds number (Re) range and for each discretisation with the Equation 6, where Nu_{ave} represents the average Nusselt number for the section and λ $[W/mK]$ is the conductivity of the fluid;

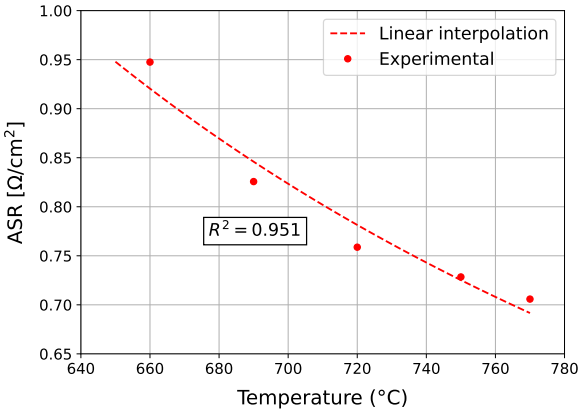


Figure 4. Experimentally found ASR values and values obtained from Equation 10 are plotted against average stacked temperature.

$$dp = \lambda(Re, \varepsilon) \, \frac{L}{D} \, \rho \frac{v^2}{2} sign(v) \tag{5}$$

$$\alpha = \frac{Nu_{ave} \, \lambda}{D_{hyd}} \tag{6}$$

2.2 Cell Model

The `FuelCell.Membranes.SOFC.Simplified` cell model includes mass flow, electrical and thermal connectors. The former are used to import data on the pressure, temperature and composition of the media flowing into the anode and cathode channels from the substack model at a higher level; these data relate only to the amount of gases required for the hydrogen oxidation reaction to occur (i.e., Reaction 1) and are then used to calculate the thermodynamic state of these gases at the electrode-electrolyte interface. The electrical connectors are used to construct the electrical circuit of the cell.

The electrical circuit is built within the cell to allow the calculation of the cell voltage V_{cell} $[V]$ with the Equation 7; the circuit is represented by two pins, a block that calculates the Nernst voltage E_0 with Equation 8 and a block that calculates the voltage drop due to internal losses in the cell ΔV_{losses}.

$$V_{cell} = E_0 - \Delta V_{losses} = E_0 - (V_{act} + V_{ohm} + V_{conc}) \tag{7}$$

$$E_0 = -\frac{1}{n_e F} \left(g_{reaction} + RT_{cell} \ln \left(\frac{p_{H_2O}}{p_{H_2} \cdot p_{O_2}^{0.5}} \right) \right) \tag{8}$$

According to Andersson et al., the internal cell losses can be estimated by Equation 9, where $ASR(T)$ $[\Omega/cm^2]$ is the area specific resistance of the cell, which depends only

on the temperature of the cell (Andersson et al. 2011). E_a $[J/mol]$ is an activation energy representing all the possible losses the current experiences as it passes through the stack's electrical circuit, while ASR_0 is the reference area specific resistance measured at the reference temperature T_0 $[K]$. This simplification is necessary in this case study because the materials used in the SOFC stack are covered by trade secrets, so using the parameters and coefficients required by the specific activation, ohmic and concentration loss models would have been misleading.

$$V_{cell} = E_0 - ASR(T) \cdot j_{cell} \tag{9}$$

$$ASR(T) = ASR_0 \cdot \exp\left[\frac{E_a}{R}\left(\frac{1}{T_{cell}} - \frac{1}{T_0}\right)\right] \tag{10}$$

$$\ln ASR = \frac{E_a}{R}\frac{1}{T} + \left(\ln ASR_0 - \frac{E_a}{RT_0}\right) \rightarrow y = mx + q \tag{11}$$

The experimental ASR_0, E_a and T_0 values, reported in Table 2, are obtained by solving Equation 11 for the experimental points given in Figure 4 (Zendrini et al. 2021), where the average furnace temperature is determined by averaging the anode and cathode gas temperatures at the inlet and outlet. This average value is considered to be representative of the thermal conditions of the stack for ASR calculations, but it is dependent on the internal geometry and materials of the stack. Therefore, it cannot be validated with relevant results in the current case study.

Table 2. Values of the coefficients used in the $ASR(T)$ equation for the calculation of cell internal losses when the stack model is run with pure ammonia.

	Unit	Experimental	Simulation
ASR_0	Ω/cm^2	0.92	0.4
E_a	kJ/mol	21.028	21.028
T_0	$^\circ C$	660	660

The simulation values in Table 2 are found following the procedure described in subsection 3.2. It should be noted that the activation energy is kept constant, but ASR_0 varies; this allows large deviations in the cell voltage trend to match experimental curves, as shown in Figure 8.

2.3 Substack Model

The `FuelCell.Stacks.SOFC.SubStack` model includes the anode, elementary cell and cathode channels. These models are then connected thermally, electrically and in terms of mass flow; gas streams are introduced into the anode, where ammonia is decomposed, and cathode channels. The mass ports connect the two channels to the cell, only transporting the necessary species for the membrane reaction. The excess gas then flows through the channel and out of the stack. The thermal ports establish the electrodes' temperature, with a third port assigned to the cell temperature and excess heat; this heat flow must be dissipated through the end plates and side wall of the stack and transmitted to the stack-level heat box model.

Table 3. Geometrical parameters adopted for the description of the substack model (Omer et al. 2023).

Var.	Value	Unit	Description
N	10	-	number of discretizations
n_{cells}	6	-	number of cells
A_{cell}	80	cm^2	active cell area
w_{cell}	7.3	cm	cell width
l_{cell}	10.96	cm	cell length
$h_{an,ch}$	0.4	mm	anode channel height
$h_{an,el}$	0.66	mm	anode electrode thickness
h_{el}	0.02	mm	electrolyte thickness
$h_{cath,el}$	0.03	mm	cathode electrode thickness
$h_{cath,ch}$	0.8	mm	cathode channel height
Cylindrical analogy			
$d_{an,ch}$	6.1	mm	anode channel diameter
$d_{cath,ch}$	8.6	mm	cathode channel diameter

The performance of the stack is significantly influenced by the geometrical dimensions of the channels. Due to the confidential geometrical structure, the stack configuration studied by Omer et al. is considered as a reference (Omer et al. 2023): the materials constituting the anode, electrolyte and cathode electrodes are among those most commonly used in commercial applications (Ni-YSZ, YSZ and LSM, respectively), and the geometrical characteristics are close to the external appearance of the stack present in the FBK laboratory. The geometrical parameters adopted in this case study are reported in Table 3.

It should be noted that the active cell area presented in the reference case is 100 cm^2, so the cell size has been scaled to match the 80 cm^2 of the case study, while maintaining the structure along the cell axis perpendicular to the flow direction. The fraction of catalyst in the anode channel f_{cat}, presented in subsection 2.3 for the ammonia decomposition reaction, is unknown for the test bench, but

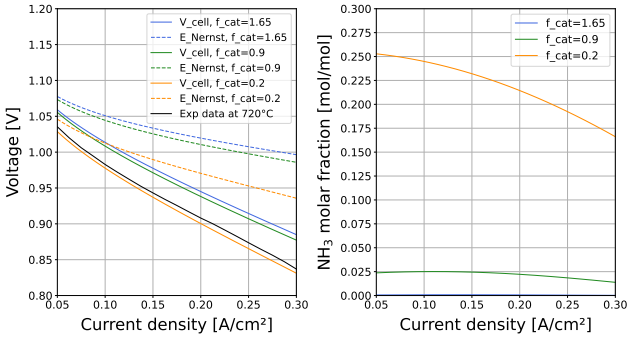


Figure 5. Simulated Nernst voltage and cell voltage (left) of the stack model at 710 °C as a function of varying catalyst fractions within the anode channel. Corresponding ammonia molar fraction at the anode outlet (right).

DOI 10.3384/ecp218405

it has a strong influence on the H_2 availability, which then determines the Nernst voltage (Equation 8). The volume of the anode electrode V_{cat} was calculated using the data provided in Table 3 following the procedure described in subsection 2.1 on ammonia decomposition and neglecting the catalyst porosity. The result, $f_{cat} = 1.65$, leads to the almost complete ammonia conversion, causing issues when calculating the Nernst voltage, as shown in Figure 5: the higher the ammonia converted, the more hydrogen available to the cell, resulting in an upward shift of the polarisation curve. Therefore, $f_{cat} = 0.2$ is used; calibrating this parameter would have been possible also by measuring the ammonia outlet of the anode channel with a mass spectrometer.

The substack model uses Equation 5 for the pressure drop calculation, so the calculation of the Reynolds number Re is greatly influenced by the anode and cathode channel diameters in Table 3; they are determined by cylindrical analogy, based on the channel's height only, ignoring the electrode's height, which is the same procedure used for the ammonia decomposition reaction calibration in subsection 2.1.

2.4 Stack Model

The experimental test bench consists of six planar oxygen-ion conducting cells electrically connected in series. In its nominal state, the stack operates at ambient pressure, 0.78 V cell voltage, and 400 A/cm^2 current density for a total nominal power of 150 W. The stack is enclosed in a furnace containing electric heaters to bring the temperature of the air near the stack to a set temperature T_{set}, which is then maintained by a proper temperature control; the anodic and cathodic gases are heated to this temperature before entering the stack. Therefore, no thermal insulation has been considered in the stack model and both gases have the set temperature presented in Table 4.It would be possible to calculate dispersion through the furnace insulation at a later stage by taking into account the difference between the average furnace temperature and the ambient temperature. However, this is beyond the scope of this study.

The `FuelCell.Stacks.SOFC.Stack` model had to be simplified because it only accepts a minimum number of substacks equal to 2, whereas in the case study there is only a short stack of 6 cells. Therefore, the $nbr_{substack}$ dependent variables are simplified, and the manifold models are removed. This stack model then extends the electrical, thermal and mass flow connections and incorporates the substack model and the heat box model, which would represent the conductive, convective and radiative heat losses from the furnace to the environment.

The operating conditions adopted during the experimental analysis are reported in Table 4; the performance of the stack is evaluated for the five temperatures set within the ohmic region of the polarisation curve, i.e., where the voltage decreases with a linear trend, thus from $J = 0.05\ A/cm^2$ to $J = 0.30\ A/cm^2$ in Figure 6.

Table 4. Operating conditions adopted for the tests conducted on the test bench in the FBK's laboratory.

Var.	Value	Unit	Description
$T_{set,1}$	650	$°C$	furnace set temperature
$T_{set,2}$	680	$°C$	furnace set temperature
$T_{set,3}$	710	$°C$	furnace set temperature
$T_{set,4}$	740	$°C$	furnace set temperature
$T_{set,5}$	760	$°C$	furnace set temperature
U_{fuel}	0.75	-	utilization rate at 32A
U_{air}	0.10	-	utilization rate at 32A
$\dot{V}_{NH_3}$	1.189	Nl/min	volume flow rate
$\dot{V}_{air}$	31.876	Nl/min	volume flow rate
$\dot{m}_{NH_3}$	1.506e-5	kg/s	mass flow rate
$\dot{m}_{NH_3}$	6.862e-4	kg/s	mass flow rate

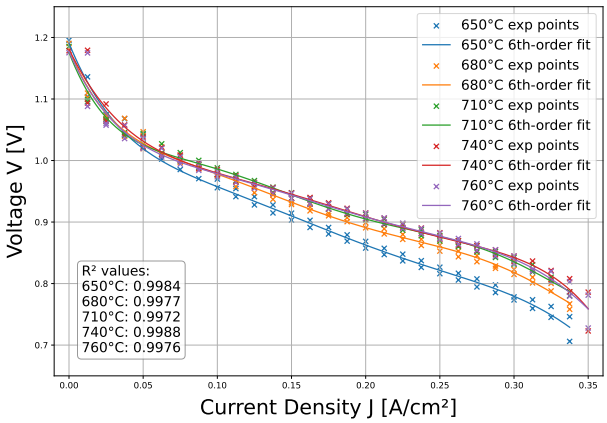


Figure 6. These are the experimental polarisation curves obtained from operating the test bench. The continuous lines were obtained by interpolating the average values of the experimental points using a sixth-order polynomial equation. The coefficient of determination R^2 for each temperature is shown to demonstrate how well the fitted values represent the experimental data.

3 Results and Discussion

The aim of the dynamic simulations, conducted for each of the models presented in the previous subsections, is to obtain the same performance as that obtained experimentally, with the smallest possible relative error ($\varepsilon_{\%}$) for each time step.

$$\varepsilon_{\%} = \left| \frac{V_{exp} - V_{sim}}{V_{exp}} \right| \tag{12}$$

3.1 Ammonia Decomposition Reaction Calibration

The amount of converted ammonia was calculated by transforming the NH_3 volumetric flow into mass flow using the ideal gas law under normal conditions. This step was necessary to compare the simulation results with those found in the literature; the conversion percentages are derived as the ratio between the decomposed ammo-

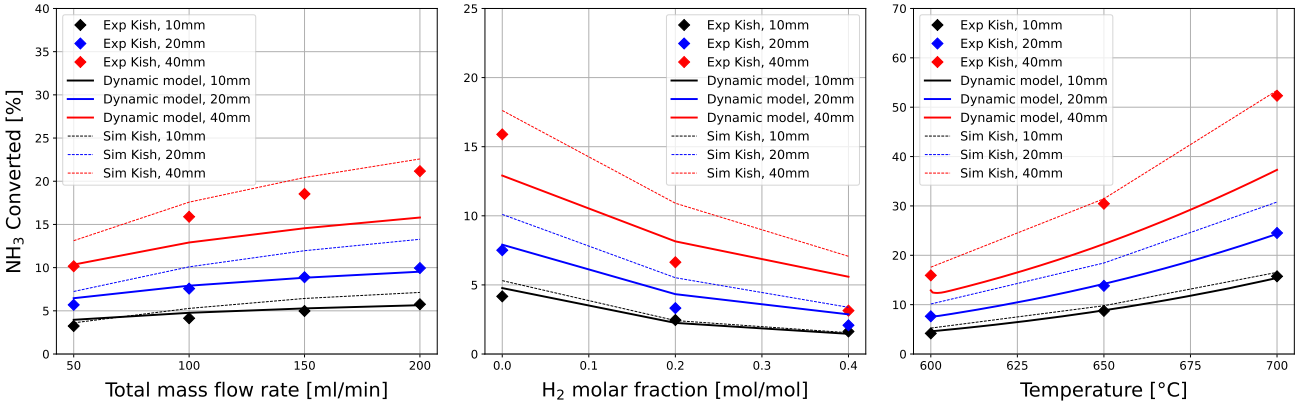


Figure 7. Comparison between the performance of the ammonia decomposition model and that obtained from simulations of the same operating conditions as described in the literature (Kishimoto et al. 2017). Discrepancies are calculated considering the experimental results with Equation 12, for both the simulation results obtained by Kishimoto et al. and the results obtained with the dynamic reaction channel model in the current case study.

nia volumetric flow and the incoming one. The calibration process results, shown in Figure 7, are obtained using $CE = 0.85$ (Equation 3). It is important to note that the discrepancy was obtained by comparison with the experimental results found in the reference article and not those from their simulations (Kishimoto et al. 2017).

It is evident that the reaction model aligns with the ammonia conversion curve for catalyst lengths of 10 mm and 20 mm, exhibiting smaller discrepancies compared to the results reported in (Kishimoto et al. 2017). However, higher discrepancies are observed when the catalyst is 40 mm long. This discrepancy, indicating a reduced conversion performance with a catalyst length of 40 mm, can be attributed to the observation that the gas exhibits increased dilution within the terminal segment of the catalyst and that the increased presence of hydrogen also reduces the ammonia conversion rate R_{dec}. This phenomenon could be attributed to the application of a lumped pseudo-1D model; however, this assertion needs to be validated by reproducing the experimental apparatus and also verifying that no potential factor has been overlooked by the authors of this case study.

This calibration process is considered a necessary step in obtaining a reaction that, when incorporated into the final SOFC stack model, can simulate the internal ammonia cracking.

3.2 Choice of ASR Values

It is important to comment on the value of the $ASR(T)$ coefficients that are used in the stack model. The experimental values are derived from the polarisation curve in the linear region, thus making $ASR_{exp}(T)$ dependent only on the average temperature of the stack, as shown in Figure 4. Consequently, all types of losses that result in a cell voltage drop from the open-circuit voltage value $OCV = E_0|_{I=0A}$ are included. However, it has been observed that, during the simulation, the cell model iteratively solves Equation 8 using the local values of partial

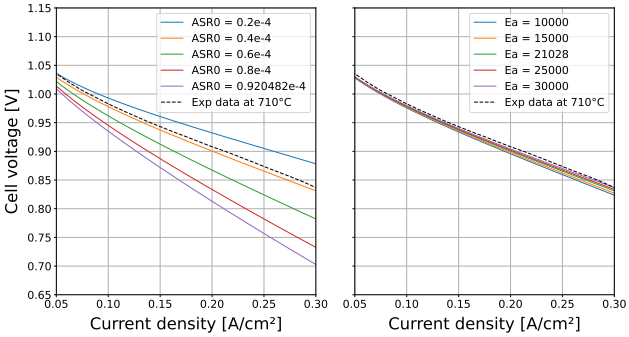


Figure 8. The two plots show a sensitivity analysis performed for operating the stack at 710 °C while keeping E_a, in the left plot, and ASR_0, in the right plot, at the experimental values.

pressures, which are determined by the cell performances: higher current requires higher hydrogen oxidation, thus the Nernst voltage decreases (see Figure 5). This decreasing effect is already taken into account in the calculation of the derivative of the polarisation curve to be calculated, so if the experimental values (Table 2) of ASR_0 and E_a were used in the model, the internal losses of the cell would be overestimated. A sensitivity analysis is presented in Figure 8 to explain the phenomenon above.

3.3 Stack Validation

Once the most influential parameters presented in subsection 2.4 are set, the stack operation is simulated under the operating conditions shown in Table 4. In Figure 9, V_{cell} values obtained from the dynamic simulation of the stack model are compared to the V_{cell} values derived from the polynomial fitting of experimental data in the ohmic current density range; the graph is represented with continuous curves to demonstrate, moment by moment, the discrepancy between the model's predictions and the actual cell voltage values. The stack model is considered validated, as the maximum value of the discrepancy is 2.18% at T=760 °C.

DOI
10.3384/ecp218405

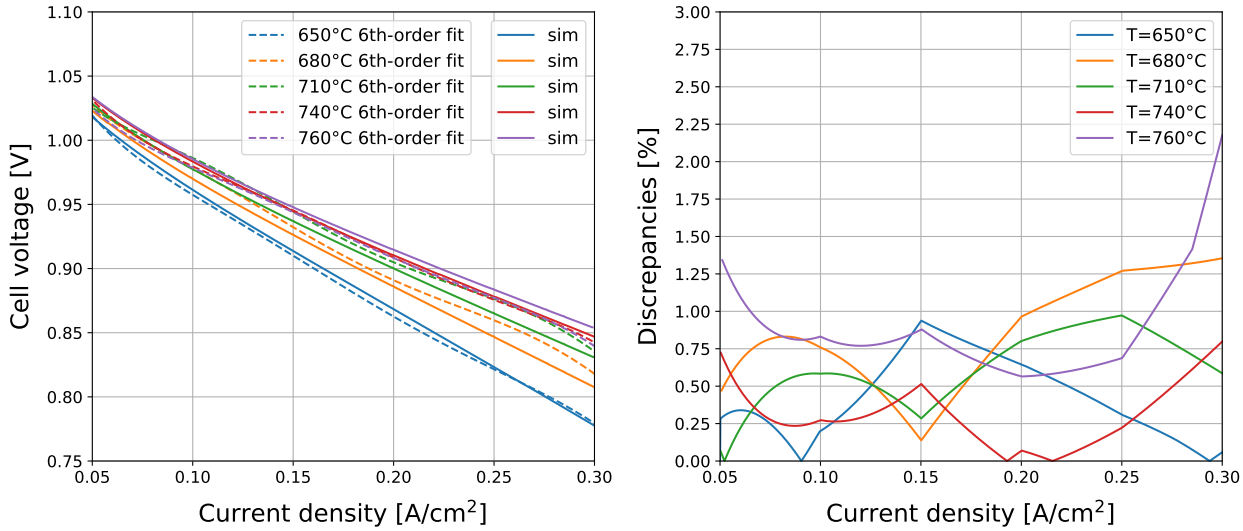


Figure 9. Comparison between the interpolated experimental polarisation curves and those obtained from the dynamic simualtion of the stack model over the selected ohmic range of current density. The discrepancy values (right) diagram are obtained with Equation 12 for each point on the polarisation curve, with a maximum value of 2.18% at T=760 °C.

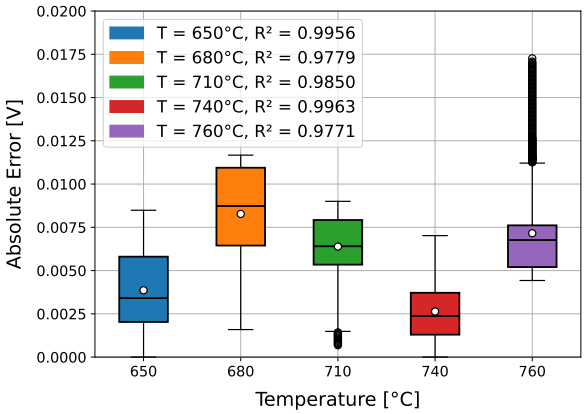


Figure 10. Boxplot of absolute voltage error distribution across temperatures. Each box shows the median, interquartile range and variability of the error between experimental and simulated values, with the mean error represented by a white circle.

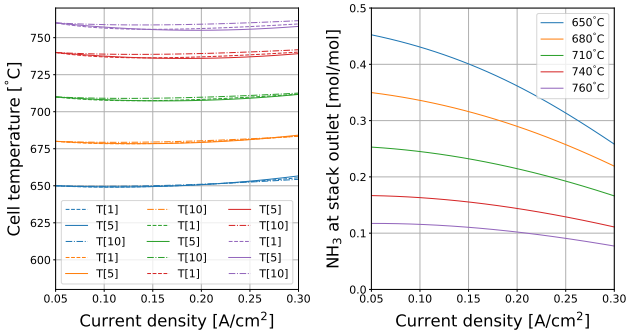


Figure 11. Cell temperature diagram (left) at the first, fifth (which represent the middle section of the cell) and tenth discretisation. Ammonia molar fraction in the anode off gases (right).

Figure 10 shows an analysis of the absolute error distribution to evaluate the performance of the model at different operating temperatures; the coefficient of determination R^2 has also been calculated and plotted for each temperature to help quantify the fit of the model to the experimental data. High R^2 values, such as 0.9956 at 650 °C and 0.9963 at 740 °C, indicate that the model can capture the variability in the experimental data. Lower values, such as 0.9779 at 680 °C and 0.9771 at 760 °C, are still considered good, as the model was built with some general assumptions that can be further investigated to improve the overall performance.

Figure 11 shows an overview of the cell performance in different sections along the longitudinal axis in terms of cell temperature and amount of ammonia left at the end of the anode channel. In fact, as explained in subsection 2.1, the ammonia conversion performance is strongly dependent on temperature; therefore, a higher amount of unconverted ammonia is expected at lower temperatures. This is confirmed by the fact that the experimental polarisation curve at 650°C (Figure 6) shows high concentration losses at high current density due to the low presence of hydrogen and hence lower ammonia conversion efficiency. As the temperature difference between the cells increases due to ammonia decomposition, thermal stress on the cell material is increased, leading to problems in cell operation and reduced lifetime. Installing a mass spectrometer at the end of the stack, makes it possible to quantify the amount of ammonia remaining and thus validate also the conversion performance of the stack. By simulating the operation of the stack for longer periods, the thermal inertia can be assessed, determining the performance at different temperatures, although continuous polarisation and depolarisation tests are not performed due to the need for further details on the test bench.

4 Conclusions and Future Works

The work presented in this paper resulted in the validation of two models built with a lumped pseudo-1D approach: firstly, an ammonia decomposition model built on equations presented and validated in the literature; secondly, a SOFC stack model fed directly with ammonia, built from the structure of a test bench in the laboratory and then validated with the operating parameters used in the laboratory and experimental and polarisation curves. The validated ammonia decomposition reaction model was necessary to implement the ammonia decomposition reaction that occurs within the anode channel of the stack model when it is operated at high temperatures.

The optimal performance of the ammonia decomposition reaction and the ability of the stack model to capture more than 97% of the experimental data, with a maximum relative error of 2.18%, prove the robustness and reliability of the model in investigating the performance of a solid oxide fuel cell stack fed directly with ammonia.

This high accuracy can be further improved by adding some considerations, previously simplified for the purpose of the article, such as recalibrating the ammonia decomposition reaction based on new experimental results or setting up an optimisation problem to obtain optimal values for the area specific resistance ASR_0 and the activation energy E_a; the same approach could also be used to estimate the best coefficients for the parameters of the activation, ohmic and concentration losses equations neglected in this case study. This would extend the current range of application of the stack. These considerations arise naturally from the fact that the material and geometric structure of the stack are covered by trade secrets. Therefore, this model is intended as a basis for refining the description of the test bench model and enhancing simulation performance.

The model's objective is to obtain a dynamic response from the stack to changes in operating conditions, such as the transition between nominal load and partial load. This enables the prediction of model outputs and the optimisation of stack performance. In addition, this highly accurate model can be used as a basis for scale-up investigations and can be implemented in a Balance of Plant (BoP) model, which represents all components in an industrial plant configuration, including control loops; a highly accurate stack model allows the evaluation of operating parameters to be adopted for optimal plant performance and the optimisation of control strategies in real cases.

Acknowledgements

The project is supported by the Clean Hydrogen Partnership and its members Hydrogen Europe and Hydrogen Europe Research, under Grant Agreement No. 101101521.

References

Andersson, Daniel et al. (2011). "Dynamic modeling of a solid oxide fuel cell system in Modelica". In: *8th Modelica Conference*. DOI: http://dx.doi.org/10.3384/ecp11063593.

EuropeanCommission (2022). *Green Deal: key to a climate-neutral and sustainable EU*. Tech. rep. International Renewable Energy Agency. URL: https://www.europarl.europa.eu/topics/en/article/20200618STO81513/green-deal-key-to-a-climate-neutral-and-sustainable-eu.

IEA (2021). *Ammonia Technology Roadmap*. Tech. rep. International Energy Agency. URL: https://iea.blob.core.windows.net / assets / 6ee41bb9 - 8e81 - 4b64 - 8701 - 2acc064ff6e4 / AmmoniaTechnologyRoadmap.pdf.

IRENA (2024). *World Energy Transitions Outlook 2024: 1.5°C Pathway*. Tech. rep. International Renewable Energy Agency. URL: https://www.irena.org/Publications/2024/Nov/World-Energy-Transitions-Outlook-2024.

Kishimoto, Masashi et al. (2017-01). "Formulation of ammonia decomposition rate in Ni-YSZ anode of solid oxide fuel cells". In: *International Journal of Hydrogen Energy* 42, pp. 2370–2380. ISSN: 03603199. DOI: 10.1016/j.ijhydene.2016.11.183.

Lyu, Zhen-Hua et al. (2023). "Design of ammonia oxidation electrocatalysts for efficient direct ammonia fuel cells". In: *EnergyChem* 5.3, p. 100093. ISSN: 2589-7780. DOI: https://doi.org/10.1016/j.enchem.2022.100093.

McBride, Bonnie J., Michael J. Zehe, and Sanford Gordon (2002-09). *NASA Glenn Coefficients for Calculating Thermodynamic Properties of Individual Species*. Tech. rep. NASA. URL: https://ntrs.nasa.gov/citations/20020085330.

NIST (2023). *NIST Ammonia properties*. Tech. rep. NIST. URL: https://webbook.nist.gov/cgi/cbook.cgi?ID=C7664417.

Omer, Ahmed et al. (2023-07). "Design and performance optimization of a direct ammonia planar solid oxide fuel cell for high electrical efficiency". In: *Journal of Power Sources* 573, p. 233135. ISSN: 0378-7753. DOI: 10.1016/j.jpowsour.2023.233135.

Wan, Zhijian et al. (2021). "Ammonia as an effective hydrogen carrier and a clean fuel for solid oxide fuel cells". In: *Energy Conversion and Management*. ISSN: 0196-8904. DOI: https://doi.org/10.1016/j.enconman.2020.113729.

Wang, Bin et al. (2022). "Ammonia as a green energy carrier: Electrochemical synthesis and direct ammonia fuel cell - a comprehensive review". In: *Fuel Processing Technology* 235, p. 107380. ISSN: 0378-3820. DOI: https://doi.org/10.1016/j.fuproc.2022.107380.

Zendrini, Michele et al. (2021). "Assessment of ammonia as energy carrier in the use with reversible solid oxide cells". In: *International Journal of Hydrogen Energy*. DOI: https://doi.org/10.1016/j.ijhydene.2021.06.139.

Zheng, Chunzheng et al. (2023). "Research Status, Optimization Strategies, and Future Prospects of Ammonia Decomposition Catalysts for COx-Free Hydrogen". In: *Industrial & Engineering Chemistry Research*. DOI: https://doi.org/10.1021/acs.iecr.3c01261.

Dynamic Simulation of a PEM Electrolysis System

Markus Pollak[1] André Thüring[1] Wilhelm Tegethoff[1,2]

[1]TLK-Thermo GmbH, Germany, {m.pollak, a.thuering}@tlk-thermo.com
[2]Institut für Thermodynamik, Technische Universität Braunschweig, Germany,
{w.tegethoff}@tu-braunschweig.de

Abstract

Electrolysis systems are the key technology to produce hydrogen from renewable energy sources. The modeling and simulation of proton exchange membrane (PEM) electrolyzers and the connected power grid present challenges due to highly fluctuating capacities in the available electrical power. Our study focuses on the modeling of the electrolyzer and its balance of plant with varying power supply resulting from renewable energy sources. The balance of plant contains a water supply system, the electrical power supply and processes for treating the produced hydrogen, in particular drying. The electrolyzer model is validated using steady-state and dynamic data from the literature and it is shown that the developed model accurately represents those measurement results. Furthermore, we implement an empirical degradation model to assess the effects of different use-cases on the efficiency of the electrolyzer.

Keywords: PEM electrolysis, hydrogen, dynamic simulation, two-phase flow, degradation

1 Introduction

A water electrolyzer is a device that is used to split water into hydrogen and oxygen using electrical energy. The production of hydrogen is essential for the chemical industry, because it is the feedstock of many processes. Moreover, hydrogen is also widely discussed as a future energy carrier (Hancke, Holm, and Ulleberg 2022; Hou and Yang 2024). To produce hydrogen in a sustainable way, the combination of electrolyzer and renewable energy sources is the key (Hou and Yang 2024; Reimann, Kohlenbach, and Röntzsch 2023). The operation characteristics of renewable energy sources, such as wind energy farms and photovoltaics, are inherently dynamic. Therefore, an electrolyzer connected to a power grid with a large share of renewable energy sources should be able to handle dynamic capacities.

There are various water electrolyzer technologies on the market, each with specific properties. Our study focuses on proton exchange membrane (PEM) electrolyzers because of the following arguments. This type of electrolyzer is able to operate dynamically due to its low temperature requirements. As well, low operating temperatures allow for a fast start-up. A further advantage of the PEM electrolyzer is that it can be operated at high pressures and therefore reduces the power required to compress hydrogen for storage applications (Hancke, Holm, and Ulleberg 2022).

To assess the performance of different electrolyzers or design an electrolzyer for a specific application, detailed simulation models are needed. Electrolyzers are generally operated in combination with other systems of various engineering disciplines. Therefore, the use of Modelica as a modeling language is ideal as many Modelica libraries focusing on the modeling of energy system are available and can be used to investigate future energy systems.

PEM electrolyzer models with different levels of detail have been published in the literature (Engel et al. 2025; Hancke, Holm, and Ulleberg 2022; Reimann, Kohlenbach, and Röntzsch 2023; Krenz et al. 2024). A novelty of our approach presented in this work is that we combine a detailed physics-based electrolyzer system model with an empirical degradation model published by (Schofield et al. 2025).

In the current study, an electrolyzer is investigated under steady-state and dynamic operating conditions. To assess the effects of fluctuating power supply on the efficiency and degradation of an electrolyzer, the electrolyzer is connected to a wind turbine as a power supply and meteorological data are used to generate operation profiles. The combination of an electrolyzer with a renewable energy source is arguably a relevant use case for hydrogen production in the future.

In the following section, the working principle and the main features of the PEM electrolyzer are introduced. The third section discusses the function and modeling of the specific components selected to build the electrolyzer system model. In the fourth section, we focus on the simulation of steady-state and dynamic operation and study the predicted degradation. The results obtained are compared with data from the literature.

2 PEM Electrolyzer Model for System Simulations

The PEM electrolyzer model discussed in this study was developed using the commercially available Modelica libraries TIL, TILMedia and the TIL3_Addon_HydrogenEnergySystems each in version 2025.1 (TLK-Thermo GmbH 2024). Dymola 2024x is used as modeling and simulation environment

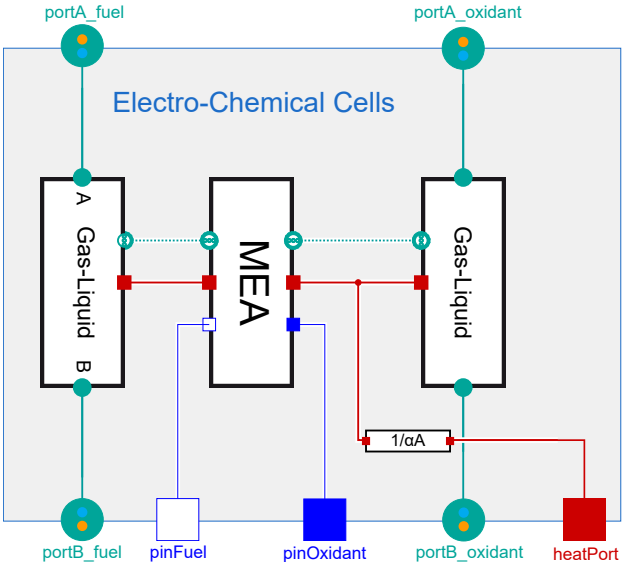


Figure 1. Structure of the stack model divided into fuel and oxidant channels and a membrane electrode assembly. Turquoise lines represent mass transfer. Dashed lines symbolize diffusive mass transfer and full lines convective mass transport. Heat transfer is marked by red lines and electrical current by blue lines.

(Dassault Systèmes SE 2024). A novelty of our approach is to combine a physics-based model with an empirical degradation model to assess the effects of degradation in realistic operating scenarios. The electrolyzer model is divided into three subdomains representing the fuel channel, the membrane electrode assembly (MEA) and the oxidant channel, as shown in Fig. 1. To calculate spatially resolved results, the model is divided into smaller discretization elements in the flow direction, but keeping the structure given in the figure. This way, the local formation and transport of product gases and water can be observed. To simulate a full stack consisting of multiple cells, one representative cell is simulated in detail and the results scaled-up by multiplying the results of the single cell by the number of cells in the stack, as described by (Tang 2016) for a fuel cell.

2.1 Electrochemical Foundation and Balances

To describe the operation of an electrolyzer, we start by introducing the relevant balance equation, and then we derive the most important metrics of the electrolyzer. On the oxidant side of the electrolyzer, water is supplied to the stack oxidant port A, as shown in Fig. 1. As typical for PEM electrolyzers, water is supplied to the anode channels of the electrolyzer only. Because the membrane is permeable to water, water is transported towards the cathode, as described in the next section. On the anode electrode water is split up into protons and oxygen:

$$\text{Anode}: \quad \mathrm{H_2O} \rightarrow 2\mathrm{H^+} + 2\mathrm{e^-} + \frac{1}{2}\mathrm{O_2} \tag{1}$$

On the cathode, protons and electrons form hydrogen molecules:

$$\text{Cathode}: \quad 2\mathrm{H^+} + 2\mathrm{e^-} \rightarrow \mathrm{H_2} \tag{2}$$

In summary, the overall reaction reads:

$$\mathrm{H_2O} \rightarrow \mathrm{H_2} + \frac{1}{2}\mathrm{O_2} \tag{3}$$

The products and water leave the system at the fuel port B and oxidant port B. Heat can be removed using a heat port. In the electrolyzer, a two-phase flow is typical since the water is supplied in liquid state and the products are gases. To split water into oxygen and hydrogen, electrical power must be supplied to the electrolyzer. The calculation of the required electrical power is described in the following paragraph. In the model, the current is supplied directly to the MEA as shown in Fig. 1.

To derive the minimal power required for the electrolysis, a steady-state operation is assumed in the following equations, but the model is formulated for dynamic operation. The following mass balance holds for the electrolyzer:

$$\dot{m}_{\mathrm{H_2O,in}} - \dot{m}_{\mathrm{H_2O,out}} - \dot{m}_{\mathrm{H_2,out}} - \dot{m}_{\mathrm{O_2,out}} = 0 \tag{4}$$

The first law of thermodynamics for the electrolyzer reads:

$$\dot{H}_{\mathrm{H_2O,in}} - \dot{H}_{\mathrm{H_2O,out}} - \dot{H}_{\mathrm{H_2}} - \dot{H}_{\mathrm{O_2}} + \dot{Q} + P = 0 \tag{5}$$

The second law of thermodynamics is stated as follows:

$$\dot{S}_{\mathrm{H_2O,in}} - \dot{S}_{\mathrm{H_2O,out}} - \dot{S}_{\mathrm{H_2}} - \dot{S}_{\mathrm{O_2}} + \frac{\dot{Q}}{T} = \dot{S}_{\mathrm{prod}} \tag{6}$$

We assume an isobaric and isothermal reaction, so that the reactants and products have the same pressure and temperature.

Combing the first and the second law results in a new equation:

$$\dot{G}_{\mathrm{H_2O,in}} - \dot{G}_{\mathrm{H_2O,out}} - \dot{G}_{\mathrm{H_2}} - \dot{G}_{\mathrm{O_2}} + P = T\dot{S}_{\mathrm{prod}} \tag{7}$$

Here, the relation for the Gibbs free energy is used:

$$G = H - TS \tag{8}$$

A special relation holds when no entropy is produced: The minimum work needed to split a water molecule is given by the Gibbs free energy of the reaction ΔG_R:

$$\Delta G_R = \Delta H_R - T\Delta S_R \tag{9}$$

It is the difference of the reaction enthalpy ΔH_R and the product of the cell temperature T and reaction entropy ΔS_R. The terms of the equation are explained below. The Gibbs free energy of the reaction can be directly linked

DOI
10.3384/ecp218413

to the voltage between the electrodes using the number of transferred electrons z and the Faraday constant F:

$$U_{\text{rev}} = -\frac{\Delta G_R}{zF} \tag{10}$$

The Gibbs free energy of the reaction depends on the temperature, the pressure and the composition of reactants supplied to the electrolyzer.

Another important voltage, the so-called thermoneutral voltage, can be derived from the reaction enthalpy:

$$U_{\text{TN}} = -\frac{\Delta H_R}{zF} \tag{11}$$

When operated at this voltage an adiabatic electrolyzer will operate at constant temperature (Bernt 2019), assuming no temperature gradients in the electrolyzer. Otherwise, the reversible heat of the reaction leads to a change in the cell temperature. The reversible heat is calculated using the electrical current and the thermoneutral and reversible voltages:

$$\dot{Q}_{\text{rev}} = I(U_{\text{rev}} - U_{\text{TN}}) \tag{12}$$

In addition to the reversible heat, the losses occurring in an operating cell produce irreversible heat flows. Those losses are directly related to the entropy production rates, as presented by (Struchtrup 2014). The major losses to be considered in an electrolyzer cell are activation, Ohmic, and mass transfer losses and are introduced in the following section.

2.2 Electrochemical Transport Models

In the following, the transport in the MEA is presented. The water content λ_{W} represents the amount of water in the membrane according to (Springer, Zawodzinski, and Gottesfeld 1991). In the electrolyzer, the membrane is always equilibrated with liquid water on the oxidant side and therefore water content is assumed to be constant according to (Springer, Zawodzinski, and Gottesfeld 1991):

$$\lambda_{\text{W}} = 22 \tag{13}$$

The proton conductivity of the membrane is directly affected by its water content and is modeled as stated by (Springer, Zawodzinski, and Gottesfeld 1991):

$$\sigma_{\text{mem}} = (0.5139\lambda_{\text{W}} - 0.326)\exp\left(1268\,\text{K}\left(\frac{1}{303\,\text{K}} - \frac{1}{T}\right)\right) \tag{14}$$

Using the proton conductivity σ_{mem}, the membrane thickness δ_{mem} and the active area A, the electrical resistance of the proton conduction can be calculated:

$$R_{\text{mem}} = \frac{\delta_{\text{mem}}}{\sigma_{\text{mem}} A} \tag{15}$$

The irreversible voltage loss is then calculated by the following relation:

$$\Delta U_{\text{ohm}} = R_{\text{mem}} I \tag{16}$$

The activation loss is calculated using a Tafel equation with the parameters crossover current density i_{X} and the exchange current density i_0:

$$\Delta U_{\text{act}} = \frac{RT}{\alpha z F}\log\left(\frac{i + i_{\text{X}}}{i_0}\right) \tag{17}$$

Since the losses on the fuel side are comparatively small, they are neglected and only the oxidant side is considered.

Furthermore, the mass transfer losses are accounted for employing a limiting current density of $i_L = 6\,\text{A/cm}^2$ (Crespi et al. 2023):

$$\Delta U_{\text{conc}} = \frac{RT}{\alpha z F}\log\left(\frac{i_L}{i_L - i}\right) \tag{18}$$

The degradation of electrolzyer cells during operation is a current topic of research and of great importance when addressing economic viability questions. In the literature, different models for catalyst and membrane degradation can be found. A major drawback of the currently published models is their high computational demand. Therefore, an empirical model fitted by (Schofield et al. 2025) is implemented in the electrolyzer model.

$$\frac{d\Delta U_{\text{deg}}}{dt} = \begin{cases} 30\,\mu\text{V/h} & \text{if } 0 < i < 1\,\text{A/cm}^2 \\ 30\,(\mu\text{V cm}^4)/(\text{A}^2\,\text{h})\,[i(t)]^2 & \text{otherwise} \end{cases} \tag{19}$$

The data used for the calibration of the degradation model originate from accelerated stress tests and, therefore, might overpredict the rate of degradation under real operating conditions.

The cell voltage is then calculated using the losses ΔU introduced before and the reversible cell voltage U_{rev}:

$$U_{\text{cell}} = U_{\text{rev}} + \Delta U_{\text{ohm}} + \Delta U_{\text{act}} + \Delta U_{\text{conc}} + \Delta U_{\text{deg}} \tag{20}$$

With the cell voltage, the heat release from the reaction can be calculated:

$$\dot{Q} = I(U_{\text{cell}} - U_{\text{TN}}) \tag{21}$$

The main functions of the membrane in the electrolyzer are proton conduction, electric insulation and separation of the species. However, a real membrane is not fully impermeable to the species in the electrolyzer. Therefore, both the produced gases and water can partially travel through the membrane. The transport of the produced gases is driven by a partial pressure difference, while the transport of water occurs due to an absolute pressure gradient and because of electro-osmotic drag (EOD).

The permeation is modeled with the following equation and using parameters published by (Ahluwalia and Wang 2007) :

$$\dot{n}_{\text{drag,i}} = \frac{K_i A}{\delta_{\text{mem}}}\Delta p_i \tag{22}$$

For the calculation of the EOD, a constant drag coefficient $\varepsilon_{\text{EOD}} = 2.5$ is used (Ma et al. 2021). The water transfer is then calculated using the electric current:

$$\dot{n}_{\text{EOD}} = \frac{\varepsilon_{\text{EOD}} I}{F} \tag{23}$$

The heat produced by the reaction is transported to the fluid channels as described in the next section.

2.3 Fluid Channels

Fluid channels are required to supply the anode with water and remove reaction products from the electrolyzer. The channels are connected to both sides of the MEA as shown in Fig. 1. The electrochemical reaction produces gaseous products that mix with liquid water on the anode side. Therefore, it is necessary to consider two-phase flow characteristics when modeling the flow inside the fluid channels. In this study, we use a phase-separate modeling approach that allows us to model a thermal and chemical non-equilibrium of the gas and liquid phases. The relevant equations for describing the flow in the electrolyzer are outlined below.

First, the most important quantities of the two-phase flow are introduced. The void fraction is defined as the volume ratio of gas and total volume:

$$\varepsilon = \frac{V_g}{V} \tag{24}$$

The overall volume is the sum of the volumes occupied by gas and liquid phases:

$$V = V_g + V_l \tag{25}$$

The mass balance for the gas phase, denoted by the index g, is given by the following relation:

$$\varepsilon \frac{d\rho_g}{dt} V - \rho_g \frac{dV_l}{dt} = \dot{m}_{g,in} + \dot{m}_{g,out} + \dot{m}_{PC} + \sum \dot{m}_{MT,g,i} \tag{26}$$

Here, the time derivative of the liquid volume is given by

$$\frac{dV_l}{dt} = \beta \frac{dT}{dt} \frac{m_l}{\rho_l} + \frac{1}{\rho_l} \frac{dm_l}{dt} \tag{27}$$

with β being the isobaric expansion coefficient. Using a smooth limiter, it is always guaranteed that small amounts of liquid and gas phase are present in each control volume to avoid singularity.

Using an explicit formulation of time derivative of the density avoids symbolic manipulation of the equation system and thereby simplifies the solution (Richter 2008):

$$\frac{d\rho}{dt} = \left(\frac{d\rho}{dh}\right)_{p,\xi} \frac{dh}{dt} + \left(\frac{d\rho}{dp}\right)_{h,\xi} \frac{dp}{dt} + \left(\frac{d\rho}{d\xi}\right)_{p,h} \frac{d\xi}{dt} \tag{28}$$

For the liquid phase the mass balance reads:

$$\frac{dm_l}{dt} = \dot{m}_{l,in} + \dot{m}_{l,out} - \dot{m}_{PC} + \dot{m}_{MT,l} \tag{29}$$

Both equations are explicitly coupled by a possible phase change given by the common term $\dot{m}_{PC}$.

For the pressure calculation, a mechanical equilibrium of both phases is assumed:

$$p_g = p_l \tag{30}$$

The pressure is derived from the gas phase. The momentum balance is formulated in steady-state.

The gas phase consists of a mixture of hydrogen, oxygen and water vapor.

$$\begin{aligned} m_g \frac{d\xi_i}{dt} = {} & (\xi_{i,in} - \xi_i)\dot{m}_{g,in} + (\xi_{i,out} - \xi_i)\dot{m}_{g,out} \\ & + (\xi_{\text{condensed,i}} - \xi_i)\dot{m}_{PC} + \xi_i \dot{m}_{MT,g,i} \end{aligned} \tag{31}$$

Here, the mass fraction vector $\xi_{\text{condensed,i}}$ contains zeros except for a one at the position representing water. Since the liquid phase consists of water as one pure species in the model only, no explicit formulation of the species balance is required for the liquid.

The energy balance is formulated for each phase separately. However, for the current study, a thermal equilibrium is assumed:

$$T_g = T_l \tag{32}$$

For the gas phase, the energy balance reads:

$$\begin{aligned} m_g \frac{dh_g}{dt} = {} & (h_{g,in} - h_g)\dot{m}_{g,in} + (h_{g,out} - h_g)\dot{m}_{g,out} + (h_l - h_g)\dot{m}_{PC} \\ & + \sum (h_{i,MT} - h_g)\dot{m}_{MT,g,i} + \dot{Q} + \varepsilon V \frac{dp}{dt} \end{aligned} \tag{33}$$

The liquid phase energy balance is as follows:

$$\begin{aligned} m_l \frac{dh_l}{dt} = {} & (h_{l,in} - h_l)\dot{m}_{l,in} + (h_{l,out} - h_l)\dot{m}_{l,out} - (h_l - h_g)\dot{m}_{PC} \\ & + (h_{MT} - h_l)\dot{m}_{MT,g,i} + \dot{Q} \end{aligned} \tag{34}$$

The heat transfer from fluid to membrane and the bipolar plate is calculated using a temperature difference and a fixed heat transfer coefficient:

$$\dot{Q} = \alpha A \Delta T \tag{35}$$

For the forced convection a constant heat transfer coefficient of $\alpha = 2000\,\mathrm{W/(m^2K)}$ is used, reflecting that the exact geometry of the channels in the bipolar plate is unknown. Furthermore, the heat loss to the environment is calculated using the stack surface area and the heat transfer correlations for free convection from the literature (Klan and Thess 2013). The heat transfer area was calculated according to (Tjarks 2017).

Analogously to heat transfer, the mass transfer processes are calculated using a mass transfer coefficient and a concentration difference.

$$\dot{m} = \beta A \Delta c \tag{36}$$

The interface mass transfer of water is driven by a concentration difference between actual and saturated state:

$$\dot{m}_{\text{PC}} = \beta_{\text{PC}} A (c_{\text{sat}} - c) \tag{37}$$

To simplify the calculation, an ideal transport can be assumed, whereby the concentration c and the saturation concentration c_{sat} are set equal. Due to intense mixing, the produced gases are typically saturated with water.

DOI
10.3384/ecp218413

3 Electrolyzer System Model

The modeled electrolyzer system is shown in Fig. 2. To produce hydrogen of defined quality in a PEM electrolyzer, a water supply system, an electrical connection to a power supply, and a hydrogen treatment system are necessary. The modeled hydrogen treatment system consists of two drying stages, thermal drying and adsorptive drying, followed by a compressor. In the following, the subsystems and their specifications are discussed in detail.

Stack

An electrolyzer previously analyzed in the literature (Crespi et al. 2023) is modeled and simulated in this work. A summary of the most relevant stack parameters is given in Table 1. Two separators are directly connected to the stack to remove liquid water from the gas flows on fuel and oxidant side. On the oxidant side, the removed water is fed to the liquid loop again. The oxygen produced by the electrolyzer is released to the environment. On the fuel side, further treatment of the liquid water was not modeled. However, the produced hydrogen gas, which is typically moisture-laden, undergoes further treatment in the subsequent steps.

Table 1. Parameters of the modeled electrolyzer stack.

Quantity	Symbol	Unit	Value
Nominal Current	I	A	410
Nominal Power	P_{el}	kW	60
Number of cells	n_{C}	1	65
Active Area	A_{MEA}	cm^2	214
Membrane Thickness	δ_{mem}	mm	0.322
Thermal Capacity	C_{Stack}	kJ/K	155.0

Water Supply and Cooling Loop

The water supply system circulates water in the cathode loop and feeds to the electrolyzer to replace water consumed by the electrochemical reaction. In addition, the circulated water is employed for the thermal management of the electrolyzer. Therefore, a chiller is integrated into the water loop that can reject heat to a cooling circuit. In addition, a storage tank is included to buffer water consumption and act as a thermal capacity. To refill water a pump is modeled. The supply water is assumed to be deionized.

Hydrogen Treatment System

The purification unit consists of two different drying processes. In the first step, water is removed from the produced hydrogen by lowering its temperature below the saturation point using a heat exchanger connected to a cooling circuit. In order to fulfill the typically strict requirements for the hydrogen purity, an adsorptive drying process is implemented downstream of the thermal drying process. In this study, a temperature swing adsorption (TSA) system is modeled as suggested by (Tjarks et al. 2018) using the TIL Adsorption Library (TLK Energy GmbH 2024). The TSA system consists of two adsorbers that are operated alternately. Both adsorbers are filled with Zeolith as an adsorbent (Tjarks et al. 2018). When hydrogen is produced, one adsorber is used to remove water from the fuel gas. The adsorption process is driven by a gradient in the chemical potential of water in the fuel gas and adsorbed water. During operation, water accumulates in the adsorber until it reaches a defined loading state. When this loading state is reached, the adsorbers are switched and the adsorber that was in the drying operation is then regenerated using a recirculation system. In the recirculation loop, hydrogen is heated up to 160 °C then enters the adsorber and gets humidified inside. A blower is used to recirculate hydrogen with a defined flow rate. The water is removed from the hydrogen after leaving the adsorber by lowering the temperature of hydrogen below its saturation point using the environment or a cooling circuit as the heat sink. To switch between the two adsorbers, eight control valves are utilized.

After the purification stages, hydrogen is compressed to a defined level. This step is necessary to increase the volumetric energy density of hydrogen.

While the electrolyzer is modeled in detail as described above, the pumps and compressors are modeled with constant efficiencies. For the heat exchangers, a finite-volume-approach was employed (Richter 2008). The heat transfer coefficients are assumed to be constant. In addition, a constant separation efficiency is assumed in the water separator models shown in Fig. 2. All efficiencies and other significant parameters are summarized in Table 2. The pressure levels and the nominal coolant flow rate are adjusted according to (Crespi et al. 2023).

Table 2. Parameters of the component models in the electrolyzer system.

Quantity	Symbol	Unit	Value
Compressor Efficiency	$\eta_{\text{comp,isen}}$	1	0.7
Recirculation Blower	$\eta_{\text{b,isen}}$	1	0.4
Pump Efficiency	P_{el}	kW	0.97
Separator Efficiency	η_{sep}	1	0.99
Hydrogen Storage Pressure	p	bar	200
Cathode Pressure	p	bar	30
Anode Pressure	p	bar	3.5
Volume Tank	V	L	200
Nominal Coolant Flow Rate	$\dot{m}$	kg/s	0.63
Heat Transfer Resistance			
Chiller	R_{th}	K/W	0.001
Condenser	R_{th}	K/W	0.003
Heater	R_{th}	K/W	0.003

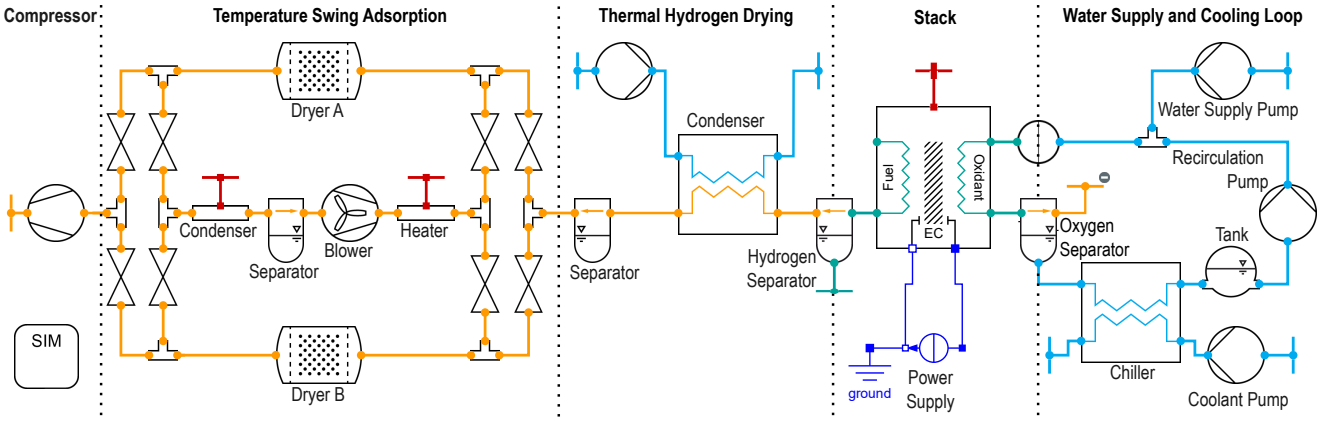


Figure 2. Electrolyzer with its main supply and hydrogen purification systems. Gas connections are colored in orange, liquid in blue, two-phase flow in turquoise and electrical current in dark blue. Heat flows are represented by red lines.

4 Steady-State and Dynamic Operation of PEM Electrolyzer

This section provides an overview of the simulation results obtained with the proposed model. First, steady-state operation and calibration of the stack are discussed. Next, dynamic operation scenarios are investigated. Finally, the degradation of the stack during dynamic operation is explored.

4.1 Steady-State Operation

Data from the literature data published by Crespi et al. (2023) is used to calibrate and validate the model discussed in the previous section. The operating conditions are given in Table 2. First, the geometry of the model is adjusted according to the data given in the paper of Crespi et al. (2023) also shown in Table 1. Next, a calibration is performed using steady-state measurement data of the same publication recorded at different operating temperatures and current densities. Data used to validate the model are not used in the fitting process.

The exchange current density $i_0 = 1.05 \times 10^{-7}\,\mathrm{A/cm^2}$, the effective membrane thickness $\delta_{\mathrm{mem,eff}} = 455\,\mu\mathrm{m}$ and the crossover current density $i_X = 0.02\,\mathrm{A/cm^2}$ have been used in a fitting process to obtain an accurate representation of the electrolyzer data published by Crespi et al. (2023) . Only data for an operating temperature of 50 °C were used for the fitting. The other part of the data, at operating temperatures of 45 °C and 60 °C, were used for validation.

In Fig. 3, the results of the calibration and validation can be found. Overall, the model accurately reflects the measurement data. The highest deviation between the measurement and the simulations is observed for small current densities. The mean absolute deviation is 0.013 V for all simulated operating points. The maximum deviation between simulation and measurement is 0.035 V .

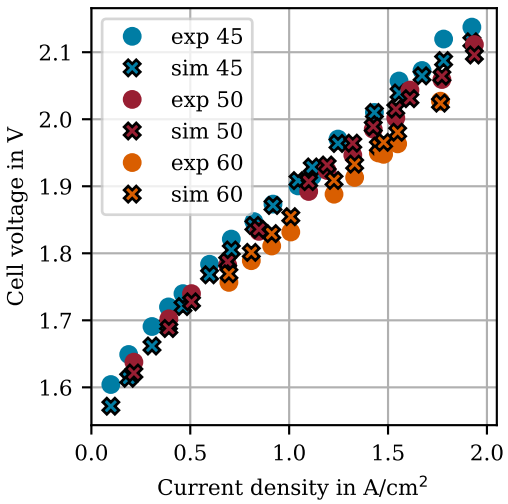


Figure 3. Comparison of measured and simulated voltages at different temperatures and current densities. The other operating conditions are set according to Table 2.

4.2 Dynamic Operation

One application of a PEM electrolyzer is the direct connection to a renewable energy source, such as a wind farm. Renewable energy sources such as wind and photovoltaics are highly fluctuating due to the weather conditions at the location of the electrolyzer. The aim of this section is to investigate whether the developed model can accurately simulate the response of the electrolyzer to a time-varying power supply. First, the calibration of the thermal capacity of the cooling circuit is conducted. Next, the gas volumes on the fuel side are calibrated and the pressure build-up investigated. Finally, the dynamic operation is studied with regard to efficiency, safety and degradation of the electrolyzer.

Warm-Up of the Electrolyzer

For an electrolyzer connected to a renewable and weather-dependent energy source, it is important to see how fast it can reach the desired operating temperature and produce

DOI 10.3384/ecp218413

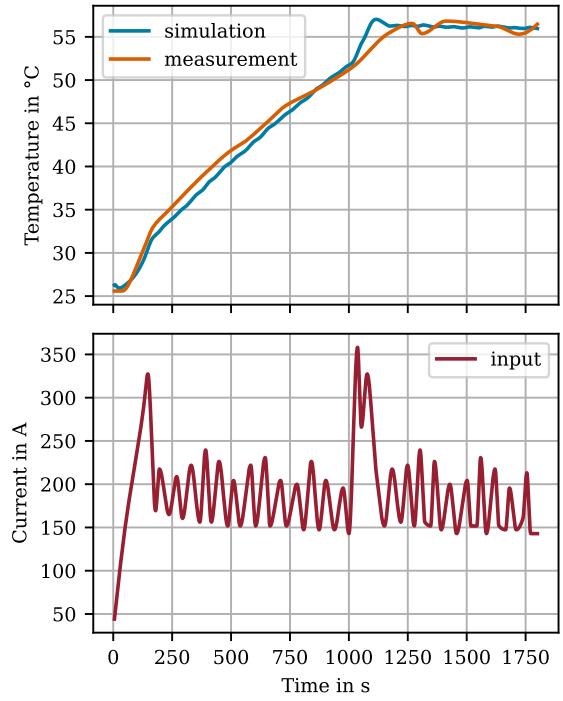


Figure 4. Comparison of simulated and measured temperatures during start-up and operation of the electrolyzer with varying current input with an average of I = 188 A.

hydrogen with high efficiency. In the work of Crespi et al. (2023), the warm-up of an electrolyzer system under given power capacity profiles is shown. The heat produced in the electrochemical reaction is utilized to warm up the electrolyzer from ambient temperature. The measurements are shown in Fig. 4. A mean current of 188 A is supplied to the electrolyzer. During warm-up the pump in the cooling circuit is deactivated so that no heat is drawn from the system. The amount of heat produced by the reaction depends on the voltage of the cell and thereby on the current supplied to the electrolyzer. The warm-up scenario measured by Crespi et al. (2023) is used in this study to calibrate the model parameters. The adjusted parameter is the water mass in the cooling circuit.

The results of the measurement and the simulation with different parameter values for the water mass are shown in Fig. 4. A steady-state is reached after about 1100 s. In the electrolyzer simulation results, it becomes obvious that the mass of water has a significant influence on the warm-up process due to the high heat capacity of water. The water mass in the liquid loop is adjusted to 50 kg in the calibration process.

The simulation of another warm-up cycle is shown in Fig. 5. Compared to the warm-up shown in Fig. 4, the average current supplied to the electrolyzer is 211 A. A quasi-steady-state temperature is reached after approximately 800 s after the start-up, both in simulation and measurement. In addition to the temperature curve, a comparison of the simulated and measured cathode pressures is shown over time. A total volume of 9.5 L was assumed for

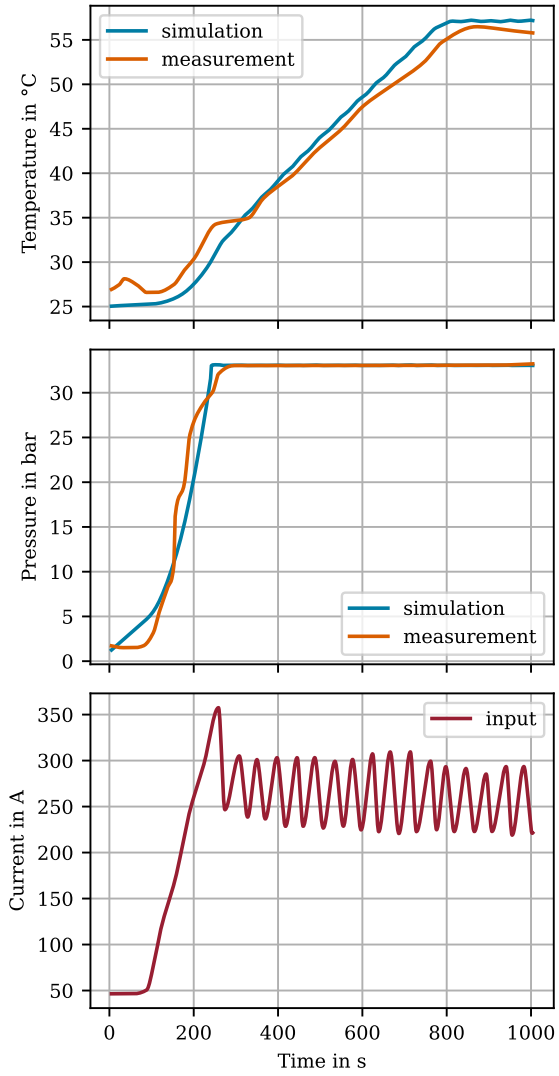


Figure 5. Comparison of simulated and measured temperatures during start-up and operation of the electrolyzer with varying current input with an average of I = 211 A. Simulated and measured pressure build-up are also shown.

the gas on the fuel side of the electrolyzer. The pressure build-up is predicted to be more uniform by the simulation model than in the measurement. In the curve representing the measurement, different slopes are visible that cannot be linked to the input of the current density. In particular, the pressure plateau at the beginning of the recorded data does not appear to be in agreement with the input current. However, quasi-steady-state of the cathode pressure is reached in measurement and simulation after about 240 s.

Electrolyzer Connected to a Wind Turbine

As future energy systems become more dynamic, flexibility in operation is crucial to adapt to fluctuating demands and decentralized production.

In order to operate the electrolyzer model with a dynamic electrical power supply, the scenario of a wind turbine is selected. Therefore, a power supply profile is cre-

ated on the basis of wind data from the DWD for Braunschweig (German Meteorological Service (DWD) 2025). The hourly data indicate the wind speed at ground level. For the calculation of the electrical power generation by the wind turbine, a simplified approach is employed based on (Heier 2022). The wind turbine has a hub height of 45 m and a rotor diameter of 44 m. Using the simplified model, the hourly electrical power supply from the wind turbine to an electrolyzer is calculated for one week in March 2023. The ambient temperature is set to $T_{\text{amb}} = 8\,^{\circ}\text{C}$, which is the average for this month.

Table 3. Electrolysis system simulation results for a week of operation in March 2023 for an electrolyzer with and without grid connection.

Quantity	Symbol	Unit	Value
With Grid Connection			
Hydrogen Produced	m_{H_2}	kg	83.51
Electrical Energy Consumption	E_{el}	kWh	4534.9
Average Efficiency	η	1	0.6138
Without Grid Connection			
Hydrogen Produced	m_{H_2}	kg	79.9
Electrical Energy Consumption	E_{el}	kWh	4367.59
Average Efficiency	η	1	0.6097

To study the effect of different operation strategies, we analyzed two scenarios in terms of the efficiency that can be achieved and the amount of hydrogen that can be produced. For the first scenario, a grid connection of the electrolyzer is assumed. If the available power from the wind turbine falls below 5 kW, it is assumed that the electrolyzer operates at a minimum power that is supplied by the electrical grid. This power is the minimum to keep the electrolyzer at operating temperature. In the second scenario with no grid connection, the electrolyzer is turned off if the available electrical power from the wind turbine falls below a 1 kW threshold. As a result, the electrolyzer temperature decreases due to heat losses to the cold environment. In both scenarios, the maximum power is limited to 60 kW.

The simulation results of both operation scenarios are plotted in Fig. 6. In the figure, the stack temperature, the hydrogen production rate, and the power supply are plotted against time. In the stack temperature, differences are visible in the times when no electrical power is available from the wind turbine. Large temperature decreases can be seen during times when no power is supplied by the wind turbine. The grid-connected electrolyzer remains at an operating temperature of roughly 50 °C due to the additional power supplied by the grid. In both hydrogen production and in the power supply, the effect of the grid connection is shown as an offset when there is no electricity from the wind turbine.

To calculate the efficiency of the system, the hydrogen production rate is integrated and multiplied by the lower heating value (LHV) of hydrogen $\text{LHV}_{\text{H}_2} = 120\,\text{MJ/kg}$. The result is then divided by the time integral of electrical power consumed by the stack, pumps, blower and compressor. This yields the efficiency of the system:

$$\eta = \frac{m_{\text{H}_2}\,\text{LHV}_{\text{H}_2}}{E_{\text{el}}} \tag{38}$$

The integral results of the calculated scenarios are summarized in Table 3. Due to the energy supplied by the grid, which is used when no electricity is available from the wind turbine, the hydrogen production of the grid-connected electrolyzer system is about 4.5 % higher than the electrolyzer without a grid connection. For the same reason, the electrical energy consumption of the grid-connected system is also higher. Moreover, the efficiency of the grid-connected system is higher. This is because hydrogen production at low current densities is very efficient and on the other hand the grid-connected electrolyzer benefits from higher operating temperatures, where operation is slightly more efficient.

Furthermore, the simulated results of the hydrogen concentration on the oxidant side were monitored to check if the flammability limit of hydrogen in oxygen is exceeded at certain operating points. The hydrogen concentration on the oxidant side depends on the permeation and the recirculation rate. It could be observed in the simulation results that the hydrogen concentrations on the anode are higher for the electrolyzer without a grid connection. In the case of the electrolyzer without grid connection, the flammability limit is exceeded at times when the power is reduced rapidly. As a result, countermeasures would have to be taken. For the grid-connected electrolyzer, the limit is never exceeded. Generally, for an accurate description of the transient accumulation of hydrogen during operation, the model needs further parameters, such as volumes of components and materials.

The simulation times for both the grid-connected and the non-connected electrolyzer scenario shown in Fig. 6 are about 75 s on a laptop equipped with an Intel i7-1360P processor.

Prediction of the Degradation in the Electrolyzer

An empirical degradation model (eq. 19) published for the PEM electrolyzer was implemented in the electrolyzer model. To assess the degradation predicted by the empirical model, the simulation of the electrolyzer connected to the wind turbine was conducted again, but for the entire period of March 2023. The grid-connected electrolyzer was selected due to its higher efficiency.

After one month of simulated operation, the voltage of an average single cell in the electrolyzer stack increased by approximately 0.045 V. Extrapolating this result over a period of the year would result in an increase of voltage

DOI
10.3384/ecp218413

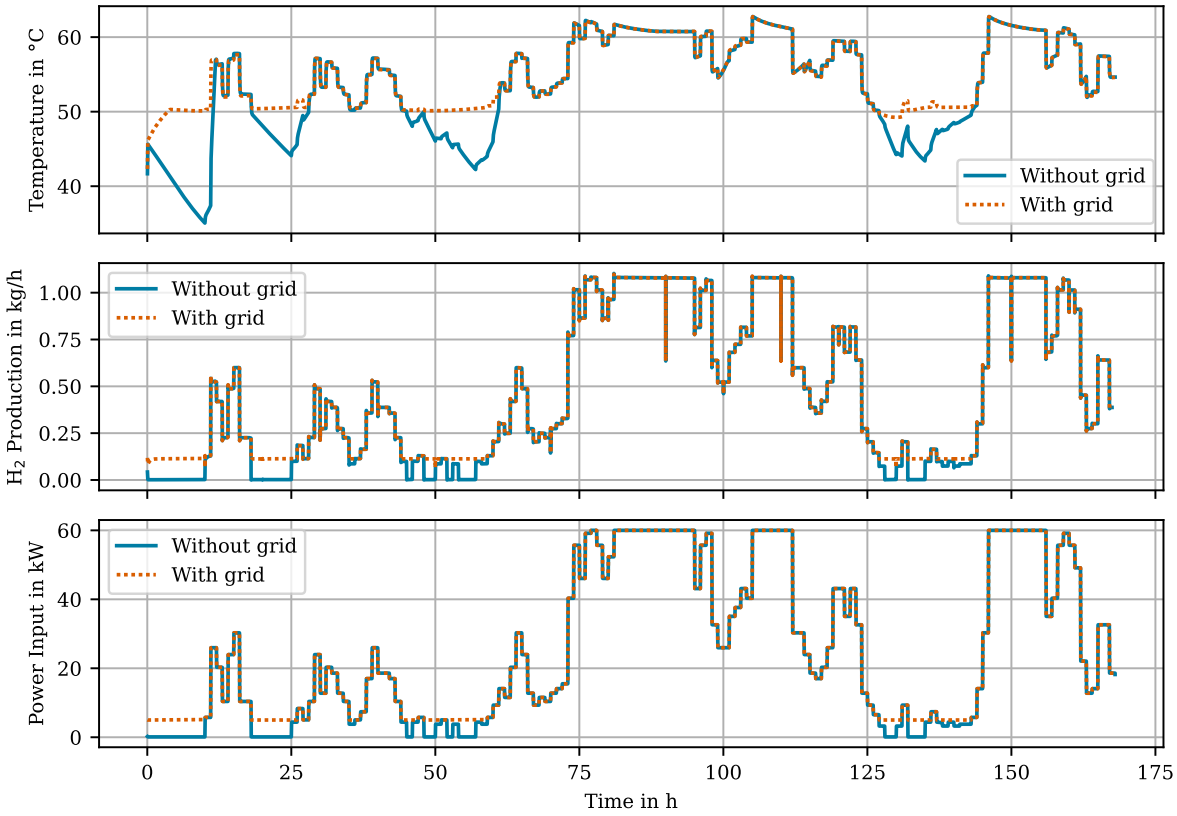


Figure 6. Simulation results of a 60 kW electrolyzer connected to a wind turbine with and without connection to the electric grid. The operating temperature, hydrogen production and power input of both scenarios are compared for the first week of March 2023.

by 0.54 V per single cell. When comparing this extrapolated result to the operating voltages published by (Crespi et al. 2023), the increase of voltage due to degradation would lower the efficiency drastically. In comparison to irreversible degradation results published by (Krenz et al. 2024), the degradation rates reported here for the electrolyzer model are high and result from the fact that irreversible and reversible degradation are not separated in the model. More detailed degradation models are reported in the literature and might help to improve the predictions (Rakousky 2016; Krenz et al. 2024).

The obtained result also demonstrates how important dynamic physics-based simulation models are to check the plausibility of calculated degradation results and to verify degradation models developed using lab-scale electrolyzers. In particular, the ability of physics-based simulation models to study realistic operating scenarios in short time frames makes them invaluable.

5 Conclusion and Outlook

In this study, a newly developed electrolyzer model was presented and investigated under steady-state and dynamic operating conditions. The dynamic model was validated for steady-state operating and dynamic scenarios using measurement data.

First, the model was parametrized according to Crespi et al. (2023) and then unknown parameters were fitted to measurement data of the same publication. The fitted model agrees well with the measured polarization curve for three different operating temperatures. Next, the dynamic response of the electrolyzer model was adjusted by fitting the liquid mass and the gas volume of the fuel to measurement data. Again, a good agreement of the measurements and simulation is achieved. In both investigated operation points, the simulated stack temperatures match the measurements accurately. Furthermore, the pressure build-up in the cathode volume of the stack is predicted accurately by the simulation model.

In the last step, a dynamic load profile was generated based on real-world wind data. Two operating scenarios for the generated load profile were analyzed, one with and another without grid connection of the electrolyzer. The efficiency of grid-connected electrolyzer is higher than that of the standalone electrolyzer. The difference can be attributed to the operating phases with relatively low load, where the efficiency of hydrogen production is high. Furthermore, it was found that the hydrogen accumulation in the electrolyzer not connected to the grid is problematic in phases with low power supply. Finally, the degradation rate under dynamic load conditions was investigated, again using the load profile of the wind turbine. Here, an overprediction of the degradation rate by the implemented model was identified using the system simulation. In the future, more detailed degradation models, such as the one proposed in (Krenz et al. 2024), could be implemented.

Nomenclature

Abbreviations

MEA	Membrane Electrode Assembly
PEM	Proton Exchange Membrane
TSA	Temperature Swing Adsorption

Latin Symbols

A	Area, m^2
c	Concentration, $mol/(m^3)$
C	Heat capacity, J/K
F	Faraday Constant, C/mol
G	Gibbs free energy, J
$\dot{G}$	Gibbs free energy flow rate, W
h	Specific enthalpy, J/kg
H	Enthalpy J
$\dot{H}$	Enthalpy flow W
i	Current Density, A/cm^2
I	Current, A
M	Molar mass, kg/mol
n	Number, *dimensionless*
$\dot{m}$	Mass flow rate, kg/s
$\dot{n}$	Molar flow rate, mol/s
p	Pressure, Pa
P	Power, W
$\dot{Q}$	Heat Flow Rate, W
R	Heat transfer resistance, K/W
S	Entropy, J/K
$\dot{S}$	Entropy Flow Rate, W/K
t	Time, s
T	Temperature, K
V	Volume, m^3
U	Voltage, V
z	Number of electrons

Greek Symbols

α	Heat transfer coefficient, $W/(m^2K)$
α	charge transfer coefficient
β	Mass transfer coefficient, m/s
δ	Thickness, m
ε	Void Fraction
ε	Electro-osmotic Drag Coefficient
η	Efficiency
λ	Water Content mol/mol
ξ	Mass fraction, kg/kg
ρ	Density, kg/m^3
σ	Proton Conductivity, S/m

Subscript

act	Activation
conc	Concentration
deg	Degradation
eff	Effective
g	Gas
H_2	Hydrogen
H_2O	Water
is	Isentropic
l	Liquid
L	Limit
mem	Membrane
O_2	Oxygen
perm	Permeation
R	Reaction
rev	Reversible
sat	Saturation
TN	Thermoneutral
x	Crossover

References

Ahluwalia, R.K. and X. Wang (2007-09). "Buildup of nitrogen in direct hydrogen polymer-electrolyte fuel cell stacks". en. In: *Journal of Power Sources* 171.1, pp. 63–71. ISSN: 03787753. DOI: 10.1016/j.jpowsour.2007.01.032.

Bernt, Maximilian Philipp (2019). "Analysis of Voltage Losses and Degradation Phenomena in PEM Water Electrolyzers". PhD thesis. Technische Universität München.

Crespi, Elena et al. (2023-02). "Experimental and theoretical evaluation of a 60 kW PEM electrolysis system for flexible dynamic operation". en. In: *Energy Conversion and Management* 277, p. 116622. ISSN: 01968904. DOI: 10.1016/j.enconman.2022.116622.

Dassault Systèmes SE (2024). *Dymola 2024x*. https://www.3ds.com/products-services/catia/products/dymola/.

Engel, Franziska K. et al. (2025-04). "Dynamic multi-physics 1D-model of a proton exchange membrane electrolysis cell". en. In: *International Journal of Hydrogen Energy* 119, pp. 56–72. ISSN: 03603199. DOI: 10.1016/j.ijhydene.2025.03.116.

German Meteorological Service (DWD) (2025). *Climate Data Centers*. https://opendata.dwd.de/.

Hancke, Ragnhild, Thomas Holm, and Øystein Ulleberg (2022-06). "The case for high-pressure PEM water electrolysis". en. In: *Energy Conversion and Management* 261, p. 115642. ISSN: 01968904. DOI: 10.1016/j.enconman.2022.115642.

Heier, Siegfried (2022). *Windkraftanlagen: Systemauslegung, Netzintegration und Regelung*. de. Wiesbaden: Springer Fachmedien Wiesbaden. ISBN: 978-3-658-36569-1 978-3-658-36570-7. DOI: 10.1007/978-3-658-36570-7. URL: https://link.springer.com/10.1007/978-3-658-36570-7.

Hou, Junbo and Min Yang (2024-06). *Green Hydrogen Production by Water Electrolysis*. en. 1st ed. Boca Raton: CRC Press. ISBN: 978-1-00-336893-9. DOI: 10.1201/9781003368939.

Klan, Herbert and André Thess (2013). "F2 Wärmeübertragung durch freie Konvektion: Außenströmung". In: *VDI-Wärmeatlas*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 757–764. ISBN: 978-3-642-19981-3. DOI: 10.1007/978-3-642-19981-3_38.

Krenz, Tobias et al. (2024-12). "Reversible Degradation Phenomenon in PEMWE Cells: An Experimental and Modeling Study". en. In: *Journal of The Electrochemical Society* 171.12, p. 124501. ISSN: 0013-4651, 1945-7111. DOI: 10.1149/1945-7111/ad96e4.

Ma, Zhiwen et al. (2021-05). "A comprehensive modeling method for proton exchange membrane electrolyzer development". en. In: *International Journal of Hydrogen Energy* 46.34, pp. 17627–17643. ISSN: 03603199. DOI: 10.1016/j.ijhydene.2021.02.170.

Rakousky, Christoph (2016). "Langzeitstabilität der Polymerelektrolyt-Wasserelektrolyse bei reduziertem Iridiumgehalt". Dissertation. Rheinisch-Westfälische Technische Hochschule Aachen.

Reimann, Ansgar, Paul Kohlenbach, and Lars Röntzsch (2023-12). "Development of a novel quasi-2D PEM Electrolyzer Model in Modelica". en. In: pp. 63–72. DOI: 10.3384/ecp20463.

Richter, Christoph (2008-04). "Proposal of New Object-Oriented Equation-Based Model Libraries for Thermodynamic Systems". PhD thesis. Technische Universität Braunschweig. DOI: 10.24355/dbbs.084-200806100200-3.

Schofield, Landon et al. (2025-01). "Dynamic optimization of proton exchange membrane water electrolyzers considering usage-based degradation". en. In: *AIChE Journal* 71.1, e18635. ISSN: 0001-1541, 1547-5905. DOI: 10.1002/aic.18635.

Springer, T. E., T. A. Zawodzinski, and S. Gottesfeld (1991-08). "Polymer Electrolyte Fuel Cell Model". en. In: *Journal of The Electrochemical Society* 138.8, pp. 2334–2342. ISSN: 0013-4651, 1945-7111. DOI: 10.1149/1.2085971.

Struchtrup, Henning (2014). *Thermodynamics and energy conversion*. Springer.

Tang, Tian (2016). "Analyse der Energieeffizienz eines Brennstoffzellenstapels im Kaltstartbetrieb". PhD thesis. Technische Universität Braunschweig.

Tjarks, Geert (2017). *PEM-Elektrolyse-Systeme zur Anwendung in Power-to-Gas Anlagen*. de. Jülich: Forschungszentrum Jülich GmbH, Zentralbibliothek. ISBN: 978-3-95806-217-7.

Tjarks, Geert et al. (2018-05). "Energetically-optimal PEM electrolyzer pressure in power-to-gas plants". en. In: *Applied Energy* 218, pp. 192–198. ISSN: 03062619. DOI: 10.1016/j.apenergy.2018.02.155.

TLK Energy GmbH (2024). *TIL Adsorption*. https://https://tlk-energy.de/en/software/til-suite/adsorption.

TLK-Thermo GmbH (2024). *TIL Suite*. https://www.tlk-thermo.com/en/software/til-suite.

DOI
10.3384/ecp218413

Model-Based Control Design for a Multi-Stacks SOC System

Axelle Hégo[1] Félix Bosio[1] Sylvain Mathonnière[1]

[1]Univ. Grenoble Alpes, CEA/LITEN, 17 Rue des Martyrs, 38054 Grenoble Cedes 9, France, {axelle.hego, felix.bosio, sylvain.mathonniere}@cea.fr

Abstract

In response to growing interest in high-temperature electrolysis, CEA-Liten has developed multi-stack Solid Oxide Cell (SOC) testing equipment comprising four electrolyser stacks. Each stack includes multiple cells, and together they form a module housed within a shared thermal enclosure and connected to a balance of plant. This setup supports the development of innovative control strategies, essential due to the stacks' sensitivity to operating conditions. To mitigate risks and costs associated with testing new strategies directly on the physical system, a Model in the Loop approach was implemented. The model replicates the real module's characteristics and operational capabilities, allowing safe and efficient design and validation of control strategies. Various transition scenarios between operating conditions—tailored to diverse production needs and constraints—were developed and validated using the model before real-world implementation. This paper presents the model-based control methodology and compares experimental results with simulations.

Keywords: system modeling, FMU, control design, Solid Oxide Electrolysis Cell SOEC

Acronyms

BoP Balance of Plant.
FMI Functional Mock-up Interface.
FMU Functional Mock-up Unit.
HEX Heat EXchanger.
HTE High Temperature Electrolysis.
MiL Model in the Loop.
PFD Process Flow Diagram.
PLC Programmable Logic Controller.
SC Steam Conversion.
SOC Solid Oxide Cell.
SOEL Solid Oxide Electrolyser.

1 Introduction

Hydrogen (H_2) production based on water electrolysis can contribute significantly to the decarbonisation objectives of the energy and industry sectors. High Temperature Electrolysis (HTE) based on Solid Oxide Cell (SOC) technology is currently in industrialisation phase with the start of MW scale demonstrators (Iyer et al. 2024). Solid Oxide Electrolyser (SOEL) systems are composed of modules containing several stacks in the same thermal enclosure and supplied by the Balance of Plant (BoP) with gases at the required conditions (composition, temperature, pressure) and with electrical current. In this context, CEA-Liten has developed a highly instrumented four stacks module presented in (Cubizolles et al. 2024). The operation of such modules is complex due to high sensitivity for some operating parameters, such as pressure or current. It is also time-consuming and cost-expensive in terms of human resources and equipment. Using multi-physic modelling is interesting to investigate behavior of the system and test new control strategies without risking to damage the real components. Once validated on the model, the new control strategies can be implemented in the real Programmable Logic Controller (PLC) and validated on the real module.

System modelling has two broad applications : design and sizing at the early phase of a project and model-based control development (Iyer et al. 2024). For this second application, the Model in the Loop (MiL) methodology is used. This methodology aims at developing low (PID) to high-level (Model Predictive Control (MPC)) control laws by leveraging the modelling of actuators and using return states of the modelled system as input to the control models (Aoun et al. 2019). Extensions of this methodology are Software in the Loop (SiL) where the algorithm control itself is compiled and tested with a simulated model and then Hardware in the Loop (HiL) in which the hardware itself (controller hardware in this case) is included, most of the time requiring real-time simulations (Zimmer 2019). This paper presents a MiL methodology based on a plant modelling using Modelica language, an export of this model as a Functional Mock-up Unit (FMU) (Modelica Association 2014) and then a co-simulation in Simulink (The MathWorks Inc. 2025a) where the control development has been implemented. This workflow can be used on almost all types of system such as in (Cech et al. 2017) with a steam turbine model, (Wang et al. 2017) with a model of refrigerant system for vehicle, and (Ensbury, Horn, and Dempsey 2020) with a vehicule model. Concerning model-based control for SOC, most studies are focuses on the coupling with other systems (nuclear, solar, etc) (Iyer et al. 2024). To the authors' knowledge, few studies have investigated specific control. In (Botta et al. 2019), SOFC and SOEC 1D dynamic stacks are modelled using Modelica and then PI controllers are defined to manage temperature of these two modes. In (Pohjoranta et al. 2015), MPC

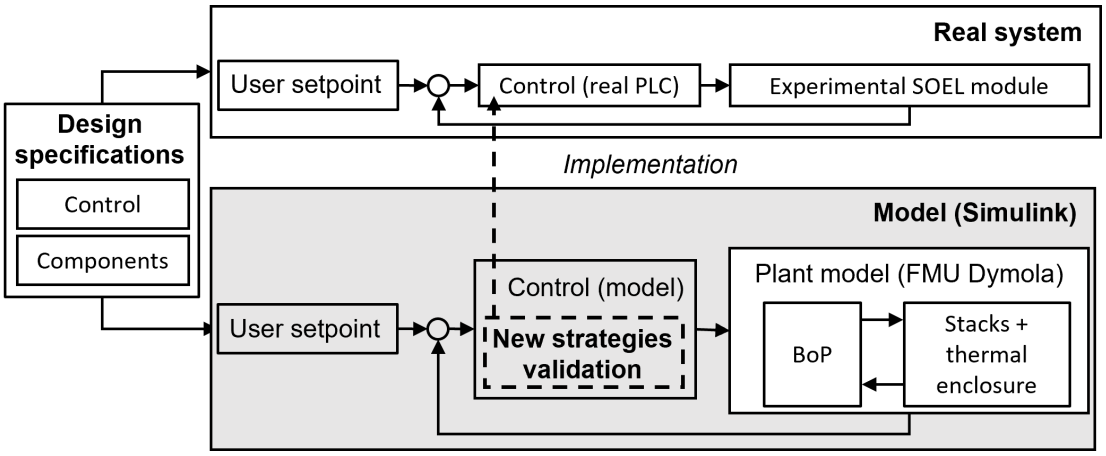


Figure 1. Overview of model-based design for control applied on a SOEL module

approach is applied on SOFC stack temperature based on experimental data of a SOFC system. In the present work, control strategies to change operating conditions are applied on a complete one-module system model and its real system counterpart. These strategies involve stacks' temperatures, gases flow rates (thus composition), current and voltage regulation.

This paper describes the methodology applied to the multi-stacks one-module system of the CEA-Liten by presenting a global model overview in Section 2. The represented system is briefly described as well as the corresponding Dymola model. The control model and control strategies are also presented. Illustration of high-level control strategies are presented in Section 3 towards a comparison between experimental and simulated data. Section 4 concludes with perspectives and challenges of the study.

2 Model and methodology

2.1 Model overview

This study focuses on a model of the multi-stacks testing equipment presented in (Cubizolles et al. 2024). A complete system model is developed using different software tools as shown in Figure 1. The Modelica language is tailored to represent complex multi-physics dynamics whereas the Simulink software is ubiquitous for control development, therefore they were selected for the development of respectively the plant model and the control model. The Functional Mock-up Interface (FMI) standard is well suited for encapsulating systems of ordinary differential equations and more generally for Modelica models, therefore in order to exchange data between those two models, the plant model was encapsulated in a Functional Mock-up Unit (FMU) based on this standard to be co-simulated in Simulink software along with the control model.

The FMU of the plant model comprises about 30 inputs which correspond to the actuator commands provided by a control model. The FMU outputs are about 90 and represent state returns representing simulated sensors which describe model system behavior. All the components of this complete system model are configured using real system specifications such as datasheet of components, measured quantities and control loop implemented in real PLC.

2.2 Plant model on Dymola

The dynamic plant model is composed of three interconnected subsystems: fuel loop, air loop and hotbox. The BoP refers to the fuel loop and the air loop which supply the conditioned fluids to the hotbox subsystem. The hotbox comprises four stacks designed and manufactured by CEA (individual power of 2.5 kW_{DC}), heaters, Heat EXchanger (HEX) and the thermal enclosure (hotbox) in which they are contained. Figure 2 is a simplified Process Flow Diagram (PFD) of the electrolysis configuration of the system.

- The **fuel loop** corresponds to the steam and hydrogen conditioning upstream of the hotbox and downstream until venting. The HEXs in series (Figure 2 (A) and (B)) heat the inlet fuel using heat from fuel downstream from the stacks. The heater (C) is controlled to supply hotbox subsystem with fuel at required temperature. The unconverted steam/H_2 mix in the reactant line downstream from the stacks and HEX is condensed in (E) before venting.

- The **air loop** is fed by the blower (Figure 2 (b)) and a by-pass ensures the flow rate. Upstream from the blower, a HEX (a) removes excess humidity from the air with water cooling. Then, inlet air is heated up by HEX (c) using heat from air downstream from the stacks and the heater (d) before the hotbox. As

DOI 10.3384/ecp218425

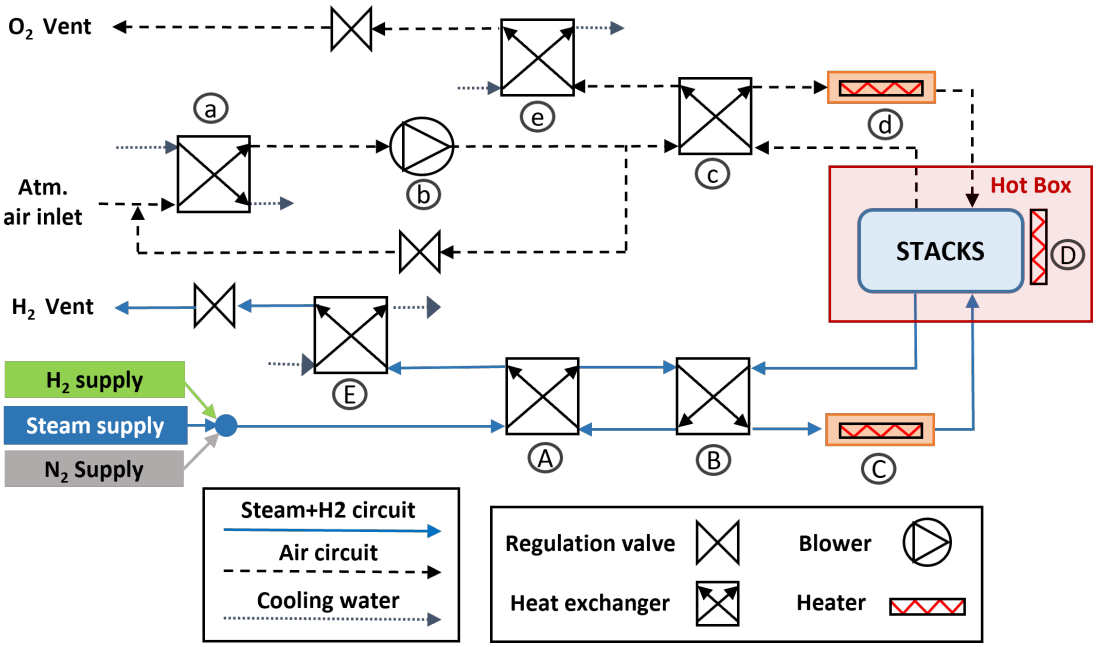


Figure 2. Process flow diagram of SOEL system from (Cubizolles et al. 2024)

with fuel, air downstream from stacks and HEX is cooled down before vent.

- **Gas feeding** of the loops are represented with gas sources of fixed pressures (Modelon Pressure boundary components) where media of Fuel Cell library media are used. Fuel media corresponds to "PreDefined.IdealGases.FastSteamHydrogen" composed of H_2O, H_2 and N_2. Air media corresponds to "PreDefined.IdealGases.FastMoistAir" composed of H_2O, N_2 and O_2.

- In the **hotbox subsystem**, fluids (air & reactant) flow through HEXs that connect the upstream and downstream flows from stacks for each medium. Then, the air and reactant are distributed respectively at the anode and cathode of the four stacks. The four stacks are positioned in a 2×2 vertical grid in a thermal enclosure comprising three heaters (Figure 2 (D)). In order to model the hotbox subsystem, the four stacks are represented by a vector of stack model based on SOFC stack component of a commercial Fuel Cell Library (Andersson et al. 2011) and connected fluidically and electrically in parallel. The stack model is composed of two reaction pipes (anode and cathode) which exchange mass and heat information with a electrochemical model. It corresponds to a reduced model based on experimental data and a CEA's Single Repeat Unit (SRU) model (Laurencin et al. 2011). Stacks' temperatures are computed through four thermal models (one per stack) taking into account conductive heat transfer, and, through a source term added to the heat equation, heat generated by electrochemical reaction and the gases enthalpy difference. The thermal model of the hotbox is composed of a network of thermal capacities representing stacks environment (heaters, stacks' supports) and the walls (~130 heat capacitors) without the stacks themselves. The four stacks included in the hotbox are each represented with 41 continuous state variables that are used to perform a thermal mapping of the stack. The stacks environment, its interactions with the stacks and interactions between stacks are computed with the network of heat capacitors linked by conductive and radiative transfers. Those heat transfer models are all finely tuned through external 3D Computational Fluid Dynamic (CFD) simulations.

Plant model had been implemented using Dymola software v2023x Refresh 1 (Dassault Systèmes 2025) as integrated development environment for Modelica Standard Library (MSLv3.2.3) and Modelon (MBLv3.7) - Fuel Cell Library (FCLv1.13) (Modelon AB 2021a; Modelon AB 2021b). The model is composed of:

- valves[1] with a quasi-static isenthalpic flow neglecting mass and energy storage, based on IEC 534/ISA S.75 standards for valve sizing.
- semi-empirical compressor models[2], implementing polytropic processes and quasi-static substance mass and energy balances are used. Both centrifugal and reciprocating compressors models have been chosen to best fit the experimental hardware.
- heat exchangers, for heat recovery from stack downstream fluid (Fig. 2 (A), (B) and (C)), represented

[1] Modelon.ThermoFluid.Valves.ValveCompressible
[2] Modelon.ThermoFluid.Compressors.DynamicCompressor and PositiveDisplacementCompressor

with dynamic NTU lumped exchanger using Modelon convective heat transfer model[3] and effectiveness functions[4] or value adapted to real components.
- heat exchangers[5], for cooling (Fig. 2 (a), (e) and (E)), represented with lumped model of indirect cooler using the logarithmic mean temperature difference for heat transfer calculation.
- friction models represented with static isenthalpic flow incorporating lumped pressure loss without energy or mass storage.

Valve parameters (opening flow characteristics and flow coefficient), compressor parameters (efficiency and flow maps) are set using components' data sheet and, in some models, based on experimental data.

The complete model once translated contains about 500 continuous time states. The hotbox thermal model dynamic being of paramount interest for HTE, it contains more than 300 continuous time states. The remaining states are distributed between the thermofluidic states of the fluids (~60 for the air loop, ~70 for the fuel loop and ~30 for the hotbox) and the stacks' electrochemistry.

This plant model has been encapsulated in FMU, v2.0, in co-simulation using implicit solver (Radau) which is stable for stiff thermofluidic models with a tolerance of 1e-4.

2.3 Control model on Simulink

The control of complex systems can be organized in two main parts:

- The low-level control corresponds to the signals applied directly to actuators. These signals could come from direct inputs (an opening value for a valve for example) or from set-point drived controllers, like the PID ones. This type of controllers compares a measured signal to a set-point and computes a correction signal to send to the actuator which impacts the measured variable.
- The high-level control corresponds to set-points definition. It can be as basic as direct values or more elaborate, with user-defined functions describing the logic to take into account, using a set of parameters, inputs, limitations, laws, etc.

For SOEL systems, the main goal is to produce a certain amount of H_2. To do this, the main variables to control are:

- the gases flow rates,
- the gases composition,
- the current density,
- the stacks' temperature.

In addition, other important variables need to be monitored in order to take into account the stacks limitations or users' requirements. Some are from direct measurements, as the cells voltage U, others are build from other variables. From this last category, the Steam Conversion (SC) is one of the most important and is defined as:

$$SC = \frac{j}{2F} * \frac{1}{\dot{n}_{H_2O}} \tag{1}$$

with the current j in A, F the Faraday's constant $(96485\, A \cdot s \cdot mol^{-1})$ and the molar flow rate of inlet steam $\dot{n}_{H_2O}$ in $mol \cdot s^{-1}$.

The nominal way of controlling the considered system for a constant H_2 production demand is to have the current density and the SC to the highest safe values possible. With these parameters fixed, Equation 1 leads to the flow rates values calculation. Then, the last parameter to determine is the stacks' operating temperature. In the current framework, it is controlled by the stacks' mean voltage value. The voltage target is the thermoneutral voltage (~1.287 V for one cell) where heat sources generated by the electrochemical process irreversibilities (such as omhic losses) are counterbalanced by the endothermicity of the electrolysis reaction. For a given flow rate and current density, only one temperature leads to operating the stacks at their thermoneutral mean voltage. The aim is to maintain the stacks voltage close to this value during their lifetime, by adjusting the temperature to compensate the voltage increase due to degradation phenomena.

It is expected that SOEL systems are operated in stationary mode, some punctual modulation of the production rate can be required. It could come from the final H_2 consumer or from inputs limitation as the electricity or steam availability.

Different strategies can be imagined to adapt the production rate of the system but they will all lead to modifications of the main parameters set-points (flow rates, current density and stacks' temperature). The way of applying these modifications can be critical for the stacks integrity and can cause irreversible damages. In order to limit the risks taken on the real system, the model-based control design approach is a pertinent solution for the test, the comparison and the validation of these strategies.

This study focuses on the comparison of two different strategies for an operating point transition:

- The slow transition, described by the state transition control diagram in Figure 3, aim to modulate the H_2 production with a constant SC. The first step

[3] Modelon.ThermoFluid.Volumes.HeatTransfer.LumpedHT_Dittus-Boelter and LumpedHT_constAlpha
[4] Modelon.ThermoFluid.HeatExchangers.Functions.parallelFlow and counterFlowEps
[5] FuelCell.HeatExchangers.GasWater.GasDryer

DOI 10.3384/ecp218425

is to start a ramp on the temperature (T ramp) to a higher or lower set-point (respectively for more or less H_2 production). The temperature evolution impacts the stacks' voltage value U and the current j can be increased or decreased by Δj to stay at the thermoneutral voltage U_{THN}. The current evolution would modify the SC value so the steam flow rate is adjusted to keep it constant. The last step corresponds to the stop of the temperature ramp. It is reached when the current is at its target value.

- The fast transition, as outlined in the state transition control diagram in Figure 4, enables the SC to adjust quickly by evolving the flow rate in order to reach the new current set-point as quickly as possible, while keeping the stacks at the thermoneutral voltage. To achieve this, a flow rate ramp (Q ramp) is initiated simultaneously with the temperature ramp (T ramp). It leads to a faster evolution of the voltage, compared to the temperature ramp only in the slow transition, that can be compensated by current variations. Once the new current set-point is reached, the flow rate ramp is halted, and the SC is gradually returned to its initial value by incrementing the flow rate, as the temperature change influences the voltage.

The described control model is based on the same logic and laws implemented in the real PLC of the test bench. These control laws were defined during the design phase of the multi-stacks testing equipment and aim to provide the desired conditions to the stacks, guaranteeing the safety of the installation and its components.

The control model has been developed using Simulink R2023b and the Stateflow library (The MathWorks Inc. 2025b). FMIKit v3.1 was used to to the plant's FMU integration instead of the native FMU integration block (Dassault Systèmes 2023). The FMU is co-simulated in Simulink and use its intern solver. Simulink model is simulated with a discrete fixed-step solver with a time-step of 1 s to be representative of real-PLC behavior. Communication time step between the FMU and the control model is inherited from Simulink solver (1 s).

These strategies have been tested on plant model presented in Section 2.2 in order to explore their limits before an implementation on the real system. The following section presents the simulation results obtained and a comparison with the corresponding experimental sequence.

3 Results

3.1 Initial conditions and results configuration

Before this study, the four stacks were operated in various conditions during several thousands of hours and as the degradation phenomena has impacted them differently, the temperature corresponding to the thermoneutral voltage (for a given set of operating conditions j and SC) is not the same for each stack. Because there is no individual temperature control implemented in this system, it was chosen to control the hotbox temperature on the stacks' mean voltage.

To assess the response of the multi-stack testing equipment to the two transition strategies and to ensure a significant variation in H_2 production (approximately 10 %), the operating condition changes outlined in Table 1 were selected for the experiments. These conditions take into account the limits on the real system such as the pressure or the maximal temperature allowed in the stacks and the maximum SC value. Moreover, the system efficiency is not in the scope of this study and regarding the level of degradation at the time of the test, the conditions are representative of a real system operation.

Table 1. Operating points of the study, normalized by cell's active area for the current density and the fuel flow rate

	OP71	*OP63*	*OP67*
Current density i $[A.cm^{-2}]$	0.71	0.63	0.67
Steam Conversion SC [%]	55	55	55
Inlet fuel flow rate $[NmL.min^{-1}.cm^{-2}]$	10.05	8.88	9.43
H_2O / H_2 [%] mol. frac.	90 / 10	90 / 10	90 / 10

3.2 Results description

The results shown in Figure 5 represent a succession of four transition sequences (two per strategy type). For each one, it starts from a nominal H_2 production, goes to a lower one before going back to the nominal. Two fast transition sequences are represented between 0 and 6 hours with variations from OP71 to OP63 to OP67 to OP71. Two slow transition sequences are represented between 6 and 18 hours with variations from OP71 to OP63 to OP71. Each sequence is highlighted with gray vertical area.

The current density is proportional to the H_2 production rate. In the transition from OP63 to OP71, a step at OP67 was needed because of pressure limitations on the system due to a high fuel flow rate value (low SC). All the changes on the parameters leading to a voltage evolution are done with the objective of maintaining the stacks' average voltage around the thermoneutral voltage, between 1.287 and 1.29 V (green area). The studied temperature is an average of fuel outlet temperature of the stacks. This average is displayed subtracting the initial temperature in

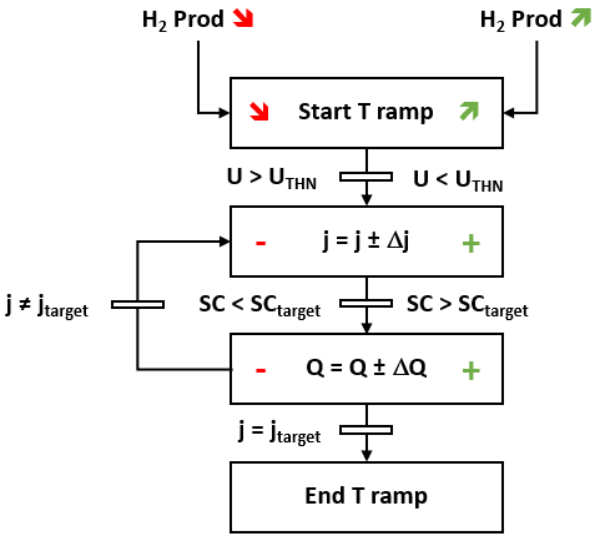


Figure 3. Slow operating point transition control state diagram

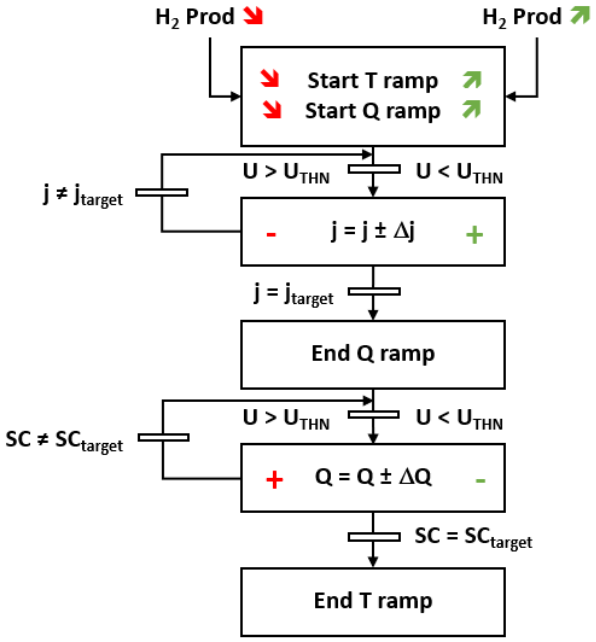


Figure 4. Fast operating point transition control state diagram

order to highlight the variation between the model and the real data, in which the absolute temperature initial values are different (degradation mechanisms are not all represented in the electrochemical model used). The fuel inlet flow rate is adjusted following the SC targeted at 55 %.

The described conditions have been reproduced in simulation (Figure 5). Computation time to simulate the sequences is about 10h20 with CPU 8th Gen Intel(R) Core(TM) i5-8365U 1.60 GHz and 16 Go RAM.

3.3 Comparison between experimental and model results

As a preliminary remark, it is important to mention that model is not fitted to real system. Main dynamics are represented by the model and it is used to test and debug control transitions. Model fitting using experimental data is a work in progress.

Overall, the model results are in accordance with the experimental data. One of the main difference between the model and real system is the thermal behavior. Even if the model describing the thermal phenomena occurring inside the hotbox, the stacks and between them is accurate comparing to 3D CFD simulations, some parameters are difficult to set precisely (as the thermal capacities for the conduction or the thermal emissivity of the materials for the radiation). It leads to differences, especially in term of dynamic behavior, which induce time shifts in the transitions, but also in term of absolute temperature value. However, it is worth noting that the main trends are represented and the order of magnitude of the temperature difference between an operating point to another is consistent (about 10 $°C$ for experimental data and about 7 $°C$ for simulation). Inertia phenomena are observed on temperatures between sequences and the inverse trends are also observed in voltages, which is the expected behavior as proved by the experimental data. Once more, trends and variation orders are consistent between model and experimental data. Another main difference is observed on SC and fuel inlet flow rate during the second sequence (fast transition from OP63 to OP67). This important difference, about 3 $NmL.min^{-1}.cm^{-2}$ for flow rate and 10 % for SC, can be explained by the cyclic pressure variations occurring on the test bench steam generator during the re-filling with cold water. The fuel flow rate, hence the SC, is directly impacted. These variations are not represented in the model which can also explain some time shift observed during transition phases. As these flow rate variations are periodic, it could be interesting to reproduce them in the model.

Despite these differences between the model and the real system, the model allows to validated the control behavior for the proposed transition strategies by pointing out the limits of these strategies.

3.4 Results analysis and transition methods comparison

The objective of the slow strategy is to reach the new production point by maintaining the steam conversion while the temperature ramp is affecting the voltage. As the overall dynamic is led by the temperature, the flow

DOI
10.3384/ecp218425

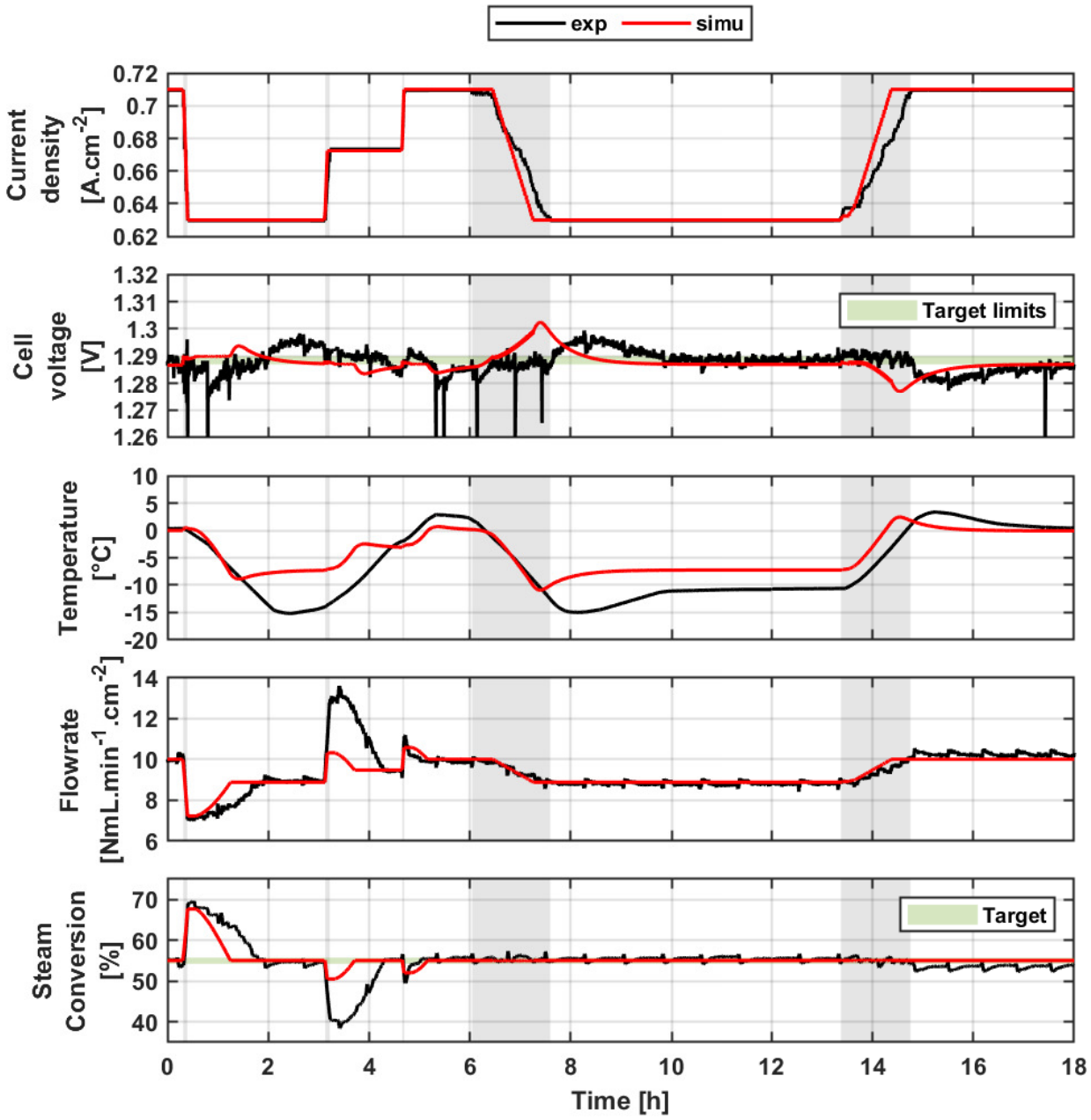


Figure 5. Experimental (black line) and simulated (red line) data for operating point transitions.

rate transition is smooth. It ensures safe conditions for the stacks.

The objective of the fast transition is to reach a new production point as soon as possible. To do so, and because the mean voltage is required to be maintained at its thermoneutral value, the only possible way is to allow variations on the flow rate (and so the SC).

The expected behavior for each strategy is clearly visible on the results. With the fast transition, the targeted production is reached ~20 time faster than with the slow one (5 minutes versus 1 hour 35 minutes).

In any case, both strategies led to a similar time for having the required operating point conditions (current density and SC). This is because the thermal dynamic is the slowest and therefore the one leading the system behavior.

To conclude this comparison, the advantages and drawbacks of each strategy can be sum-up in the following lists:

- Slow transition: The targeted production rate is reached slowly, but with a constant SC during the transition that limits the risks of having a critical value for the stacks (too high SC for less H_2 production and too high pressure due to a too low SC for more H_2 production).

- Fast transition: The targeted production rate is reached quickly. The SC is variable during the transition so the system limits need to be carefully characterized, and the safety actions adapted to avoid critical damages to the stacks.

The fast transition allows a faster way to accommodate the production rate of the system and can be of interest for some applications where the load profile is not fully stable or in a scenario with variable current input such as in some renewable energy sources. On the other side, it is less smooth than the slow transition and could be the source of accelerating stacks' degradation.

4 Conclusions and discussions

This paper describes a Model in the Loop methodology to de-risk the implementation of new control strategies for a real SOEL system. A dynamic multi-physic model of the real system has been developed using Modelica language (supported by Dymola software). Then, this model has been exported as a FMU for being co-simulated in Simulink software where the control model was developed. This approach has been used to validate strategies of operating point transitions and corresponding results are discussed.

It is worth noting that this approach is sensitive to the level of implementation of the FMI standard of each software used. This is exemplified by its application to another SOEL system model, implemented in Modelica using another proprietary library and comprising more components. Even though the plant model run with its own control without errors in Dymola, trying to implement the same control in Simulink and applying it to the corresponding FMU of the Modelica plant model resulted in difficulties during simulation (extreme slowness and generally ending up with errors). Those difficulties have been observed and the identification of bottlenecks is non-trivial task. Future studies focused on model stability are considered.

Future objective for this model is Hardware in the Loop and/or Control Hardware in the Loop approach. Thus, many challenges can be addressed on the model such as fitting, adaptation and simplification to be compatible for real-time computer with a consistent representativeness. Concerning the control model, specific toolbox enabling a transfer of control developed in Simulink to real PLC can be tested. Moreover, in the present work, robust control has not been explored considering uncertainties in the model parameters. This aspect could be investigated to improve performance under such uncertainties.

Acknowledgements

The authors gratefully acknowledge the financial support by CEA, GENVIA and partially by the REVERSI ADEME project.

References

Andersson, Daniel et al. (2011-03-20). "Dynamic modeling of a solid oxide fuel cell system in Modelica". In: *Proceedings 8th Modelica Conference*. Dresden, Germany, pp. 593–602. DOI: http://dx.doi.org/10.3384/ecp11063.

Aoun, Nadine et al. (2019-02). "Dynamic Simulation of Residential Buildings Supporting the Development of Flexible Control in District Heating Systems". In: *Proceedings of the 13th International Modelica Conference, Regensburg, Germany, March 4–6, 2019*. Vol. 157. Linköing University Electronic Press, pp. 129–138. DOI: 10.3384/ecp19157129.

Botta, G. et al. (2019). "Dynamic modeling of reversible solid oxide cell stack and control strategy development". In: *Energy Conversion and Management* 185, pp. 636–653. ISSN: 0196-8904. DOI: https://doi.org/10.1016/j.enconman.2019.01.082. URL: https://www.sciencedirect.com/science/article/pii/S0196890419301360.

Cech, Martin et al. (2017). "Novel tools for model-based control system design based on FMI/FMU standard with application in energetics". In: pp. 416–421. DOI: 10.1109/PC.2017.7976250.

Cubizolles, Géraud et al. (2024-10). "Development of a Versatile and Reversible Multi-Stack Solid Oxide Cell System Towards Operation Strategies Optimization". In: *Journal of The Electrochemical Society* 171.10, pp. 104–511. DOI: 10.1149/1945-7111/ad85fa.

Dassault Systèmes (2023). *FMIKit-Simulink*. URL: https://github.com/CATIA-Systems/FMIKit-Simulink.

Dassault Systèmes (2025). *Dymola*. URL: https://www.3ds.com/products/catia/dymola.

Ensbury, Theodor, Nate Horn, and Mike Dempsey (2020). "Dymola and Simulink in Co-Simulation: A Vehicle Electronic Stability Control case study". In: American Modelica Conference 2020. URL: https://www.claytex.com/wp-content/uploads/2021/06/Dymola-and-Simulink-in-Co-Simulation_-a-case-study.pdf.

Iyer, Siddharth et al. (2024-06). "Review of experimental and modelling investigations for solid oxide electrolysis technology". In: *International Journal of Hydrogen Energy* 72, pp. 537–558. DOI: 10.1016/j.ijhydene.2024.05.312.

Laurencin, J. et al. (2011). "Modelling of solid oxide steam electrolyser: Impact of the operating conditions on hydrogen production". In: *Journal of Power Sources* 196.4, pp. 2080–2093. DOI: 10.1016/j.jpowsour.2010.09.054.

Modelica Association (2014). *Functional Mock-up Interface for Model Exchange and Co-Simulation version 2.0*. Tech. rep. Modelica Association. URL: https://fmi-standard.org.

Modelon AB (2021a). *Fuel Cell Library Version 1.13*. Tech. rep. Modelon AB. URL: https://modelon.com/library/fuel-cell-library/.

Modelon AB (2021b). *Modelon Base Library Version 3.7*. Tech. rep. Modelon AB. URL: https://modelon.com/modelon-library-suite-modelica-libraries/.

Pohjoranta, Antti et al. (2015-03). "Model predictive control of the solid oxide fuel cell stack temperature with models based

on experimental data". In: *Journal of Power Sources* 277, pp. 239–250. DOI: 10.1016/j.jpowsour.2014.11.126.

The MathWorks Inc. (2025a). *MATLAB/Simulink*. URL: https://mathworks.com/products/simulink.html.

The MathWorks Inc. (2025b). *Stateflow*. URL: https://fr.mathworks.com/products/stateflow.html.

Wang, Kai et al. (2017). "Integration of complex Modelica-based physics models and discrete-time control systems: Approaches and observations of numerical performance". In: 12th International Modelica Conference, Prague, Czech Republic, May 15-17, 2017, pp. 527–532. DOI: 10.3384/ecp17132527.

Zimmer, Dirk (2019-02). "Towards Hard Real-Time Simulation of Complex Fluid Networks". In: *Proceedings of the 13th International Modelica Conference, Regensburg, Germany, March 4–6, 2019*. Vol. 157. Linköing University Electronic Press, pp. 579–588. DOI: 10.3384/ecp19157579.

DOI
10.3384/ecp218425

Towards Integration of PeN-ODEs in a Modelica-based workflow

Andreas Hofmann[1] Lars Mikelsons[1]

[1]Chair of Mechatronics, University of Augsburg, Germany,
{andreas.hofmann,lars.mikelsons}@uni-augsburg.de

Abstract

Hybrid modeling – the combination of first-principle models and machine learning – offers the potential to increase model accuracy while reducing modeling effort. Although approaches for creating hybrid models from system simulation models exist, the unique characteristics of Modelica-based, object-oriented models – such as modularity and reusability – can, as of today, not be utilized. In this contribution, we explore approaches for bridging this gap to enable the use of hybrid models with Modelica. Key challenges of architecture definition, training environment and reintegration of the trained machine learning parts into a Modelica model are addressed. To illustrate our approach, we present a case study involving a SCARA robot. This example demonstrates a partially integrated workflow for hybrid modeling, intended to serve as a foundation and motivation for further research.

Keywords: hybrid modeling, PeN-ODE, SciML, NeuralFMU, Julia, machine learning

1 Introduction

1.1 Motivation

In recent years, data-driven models, i.e. machine learning (ML) models, and hybrid models, which combine first-principle modeling (FPM) and data-driven approaches, have gained wide attention in the field of system simulation. Data-driven models promise to eliminate efforts for traditional modeling tasks, while additionally increasing the fidelity of models based on real-world data. Alongside those benefits, however, data-driven models are limited by their high demand for data to approximate the underlying physics, their black-box nature, and limited extrapolation capabilities, thus restricting their application. While early architectures recurrently mapped states from a point in time to the next one, the more sophisticated approach of the NeuralODE paper (Chen et al. 2018), which gained a lot attention, confined data-driven models to the system dynamics and combined them with well-known numerical solvers.

Hybrid models, in contrast to their purely data-driven counterparts, try to incorporate ML-approaches into models that are built upon laws of physics and well-established empirical approaches. Since those models aim to only learn unmodelled physical effects, their data requirements are typically way more economic. Also, extrapolation capabilities and explainability of hybrid models are usually better compared to solely data-driven models due to the underlying FPM nature.

While the term NeuralODE is widely established in the field of data-driven approaches, various names and architectures for hybrid models abound, such as NeuralFMU (Thummerer, Kircher, and Mikelsons 2021), Universal Differential Equation (Rackauckas, Ma, et al. 2021), et cetera. Hence, since there is no common wording for hybrid models, their application, especially in industrial scope, is shortened. Therefore, a concise definition for hybrid models (for system simulation) has been developed in the the ITEA OpenSCALING project[1], based on earlier definitions by Thummerer and Mikelsons (2023), Thummerer, Kolesnikov, et al. (2023) and Kamp, Ultsch, and Brembeck (2023). The definition will be presented in the next section.

Modelica's object-oriented nature makes it especially well-suited for hybrid modeling, as it enables efficient reuse of models. On one hand, integrating machine learning models into component equations introduces a high degree of flexibility. This allows for modification of arbitrary mathematical relationships, like algebraic equations and state derivative expressions. Furthermore, a trained hybrid component can be reused across different modeling scenarios. For example, it is often advantageous to model an entire test rig, including both the device under test and the dynamic behavior of the testbed components. Using recorded data, a hybrid model of the device under test can be trained, subsequently extracted from the testbed model, and then embedded into a high-fidelity model of another system. This reusability helps justify the considerable effort involved in the training phase of hybrid modeling. On the other hand, object-oriented models retain information about algebraic relationships, which facilitates the direct integration of measurement data. As long as a relationship between the measured and augmented quantities exists, the model can be trained. This approach is particularly valuable when working with field data from systems in operation, where typically only a limited set of dedicated signals is available.

However, developing hybrid models from Modelica-based systems and reintegrating them into a Modelica-centered workflow presents several challenges. Key obstacles include the training process – typically performed

[1]project page: https://openscaling.org/

within machine learning frameworks developed in Python or Julia – and the integration of Neural Networks, which are among the most commonly used machine learning models, into the Modelica environment.

1.2 Outline

In the following section, a short overview of the PeN-ODE architecture and a further partitioning based on training strategies is provided. Following that, main challenges for building and training PeN-ODEs and utilizing those in a Modelica-based workflow are depicted. In the subsequent chapter, a workflow for PeN-ODEs is demonstrated, utilizing an object-oriented model of a Scara-Robot, however starting from the Julia programming language, avoiding the transfer of the model from a Modelica-based tool. Finally, the paper closes with an outlook on further developments and challenges.

2 Definition of PeN-ODE

As mentioned before, several methods for combining first principle models and machine learning have been available for many years. To make the concept of hybrid models accessible for wider industrial application, a common definition for hybrid models, or at least for the distinct class of system simulation, has been developed within the ITEA4 OpenSCALING project under the name PeN-ODE. Although this concise definition is limited to a model architecture, a further distinction, based on training strategies, can be made, which facilitates overall communication on required data, et cetera.

2.1 PeN-ODE Architecture

A PeN-ODE, short for Physics-enhanced Neural Ordinary Differential Equation, is an ODE[2] model that combines FPM and ML models, typically Neural Networks. As illustrated in Figure 1, no particular restrictions are imposed on the interaction between the FPM and ML subsystems, allowing maximum flexibility in tailoring the hybrid model to individual use cases.

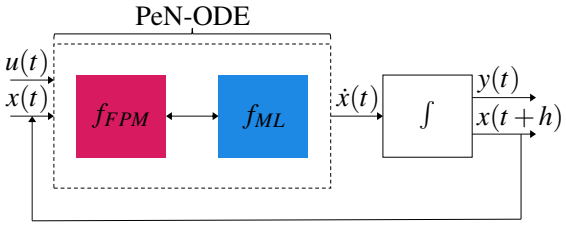


Figure 1. Architecture of a PeN-ODE with physical subsystem (f_{FPM}) and machine learning part (f_{ML}) within the solution process.
$x(t)$ - system states
$u(t)$ - inputs
$y(t)$ - outputs
h - solver step size

While this definition might seem arbitrary at first, it clearly differentiates the PeN-ODE from NeuralODEs and other data-driven approaches, where the complete dynamic of the system is represented using Neural Networks. Nonetheless, other common approaches for hybrid models are fully or partially incorporated: NeuralFMUs, the combination of Functional Mockup-Units and Neural Networks are a subset of the PeN-ODE, where the FPM is represented using the FMI standard. Universal Differential Equations (UDEs) describe a rich framework for ODEs, PDEs, SDEs, etc. and allow to build and train hybrid models but also models without any FPM (therefore similar to NeuralODEs). Also, training strategies such as Physics Informed Neural Networks are featured in the UDE framework. In this context, PeN-ODEs cover a dedicated subset of UDEs, focusing on the relevant parts for system simulation. There are far more approaches that fit into the PeN-ODE definition, though not necessarily from the field of technical system, such as Integrated Neural Network(Su et al. 1992) or methods without dedicated names (Quaghebeur, Nopens, and De Baets 2021), (Oliveira 2004). Sorourifar et al. (2023) also refer to their architecture as "Physics-Enhance Neural Ordinary Differential Equation", combining first-principle modelling and Neural Networks in chemical reaction systems.

Please note that in the PeN-ODE definition no training strategies, i.e. the training of the Neural Network, are explicitly defined. As a result, a pre-trained ML model augmenting an FPM is a valid PeN-ODE model.

2.2 Partitioning of PeN-ODE by training strategies

Although the PeN-ODE definition imposes no restrictions on the training process itself, different training strategies come with varying requirements for training data. Accordingly, alongside the definition of the PeN-ODE architecture, two distinct training strategies are delinated. It is important to note that, following successful training and integration, the specific training strategy used for the PeN-ODE becomes indistinguishable in the final model.

Closed-loop trained PeN-ODE

The first training strategy involves parameter optimization of the ML part by computing gradients through the solution of the ODE. Because this requires calculating gradients across the transient behavior of the system, aspects such as time and state events, as well as the numerical solution process itself, must be taken into account. For further details we refer to Thummerer, Olsson, et al. (2025), though focusing on NeuralFMUs.

In addition to the commonly used method of automatic differentiation, mathematical techniques, such as the Continuous Adjoint method or the Forward Sensitivity analysis for ODEs, have regained attention in recent years. The first approach performs the gradient calculation by tracing every operation on the discretized system, often referred to as "discretize-then-optimize". In contrast, the latter constructs an additional set of differential equations which are

[2]including DAEs and discretized PDEs

DOI 10.3384/ecp218435

solved alongside the original system. Since gradients are derived from the time-discrete solution values, it is often referred to as "optimize-then-discretize". A comprehensive review covering the details of gradient computation for ODEs can be found in (Sapienza et al. 2024).

The cost function of the optimization problem can aggregate any related quantity of the ODE system associated to the equations under consideration, i.e. alongside the states also algebraic quantities. Hence, considering the substantial effort for gradient computation, the constraints on available measurements from the (real-world) data are typically low. In Figure 2 the training strategy for closed-loop training is depicted.

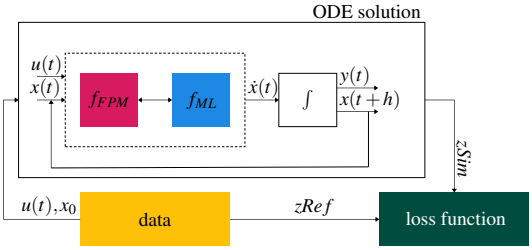


Figure 2. Closed-loop training procedure for loss calculation
$x(t)$ - system states
$u(t)$ - inputs
$y(t)$ - outputs
$zSim$ - relevant solution values from simulation
$zRef$ - reference solution values from data
x_0 - initial states
h - solver step size

Open-loop trained PeN-ODE

In contrast to the closed-loop training strategy, the open-loop procedure does not consider a transient cost function accumulating over the ODE solution. Instead, it uses state derivative quantities and algebraic variables to train the ML part based on the right-hand side (RHS) of the ODE, cf. Figure 3.

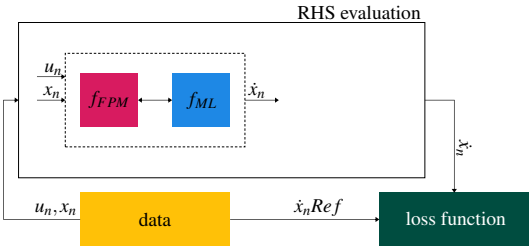


Figure 3. Open-loop training procedure for loss calculation, based on evaluation of the RHS of the ODE system
x_n - system states at time n
u_n - inputs at time n
$\dot{x}_n$ - state derivatives at time n
$\dot{x}_nRef$ - reference data at time n

As a result, the computational effort for gradient calculation is reduced and is typically handled via automatic differentiation. However, this approach places higher demands on the training data, as all relevant quantities must be explicitly available. Moreover, deviations during training on derivative level typically accumulate to increasing errors due to integration in the simulation of the trained PeN-ODE.

Although Figure 3 illustrates a joint evaluation, it is important to note that the machine learning component can also be trained independently. This corresponds to conventional data-driven approaches, as depicted in Figure 4.

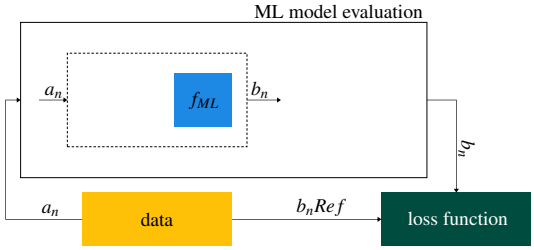


Figure 4. Open-loop training procedure for loss calculation, based on solely evaluating the ML model.
a_n - ML input at time n
b_n - ML output at time n
b_nRef - reference data at time n

3 Challenges for derivation and training of Modelica-based PeN-ODEs

As briefly mentioned in the introduction, utilizing PeN-ODEs in Modelica poses various challenges, which will be examined in more detail below. The primary challenges can be summarized as follows:

- **Training of the hybrid model:** Bridging the gap between Modelica-based modeling and common machine learning training frameworks, such as Julia
- **PeN-ODE architectures:** Limiting access to available signals and defining appropriate PeN-ODE architectures within the modular, object-oriented structure of Modelica.
- **Representation of hybrid architecture in Modelica:** Methods for incorporating trained ML components into Modelica models in a way that preserves their functionality and compatibility.

3.1 Training of the hybrid model

The first challenge in establishing an integrated workflow for PeN-ODEs in Modelica is bridging the gap to the ML training environment. Python and its extensive ecosystem of ML frameworks and libraries are indisputably the most common environment for machine learning tasks. Nevertheless, noting that many other suitable programming languages exist, we want to focus in the following passage on the Julia programming language. Julia offers a wide ecosystem for scientific machine learning (SciML) and does provide some initial approaches for bridging ML training environment and Modelica-based tools.

Tinnerholm et al. (2021) present a reimplementation of the OpenModelica compiler in Julia, called OM.jl, that can generate models for the Julia packages DifferentialEquations.jl (Rackauckas and Nie 2017), which provides methods for efficiently solving a wide range of differential equation problems, such as ODEs, DAEs, or SDEs. Additionally OM.jl can generate models for the Julia-specific environment for object-oriented modeling, ModelingToolkit.jl (Ma et al. 2021). In an additional step, similar to a Modelica compiler, ModelingToolkit symbolically processes the system to an executable set of differential-algebraic equations. Unfortunately, the corresponding Github-repositories (OM.jl, OMBackend.jl, ...) have not been maintained for years.

Similar to ModelingToolkit.jl, Modia (Elmqvist et al. 2021) offers capabilities of object-oriented modelling and simulation in Julia. Being strongly embedded into the Modelica community, the developers see Modia as an environment to test new algorithms and extensions towards further adaptions of Modelica. A translator has been developed by Otter, Elmqvist, et al. (2019), allowing to source-to-source translate a (subset) of the Modelica language to Modia. Unfortunately, the repository presented in their paper is no longer available. Bruder and Mikelsons (2021) utilized Modia to build an equation-based, hybrid non-linear single track model. However, like OM.jl, Modia is still based on an older version of Julia, released in Nov. 2021.

The Base Modelica initiative (Kurzbach et al. 2023) provides another approach for translating arbitrary, though only flattened, Modelica models to Julia. A dedicated package in the Julia SciML ecosystem exists, named BaseModelica.jl[3] , promising to parse and convert the Base Modelica model to the previously mentioned object-oriented modeling environment ModelingToolkit.jl. Please note that the initial Modelica change proposal for Base Modelica is currently in development. Once available, the pipeline could be an auspicious way of translating Modelica models to suitable simulation models for hybrid training in Julia. However, as of today, it remains unclear whether ModelingToolkit.jl is capable of supporting all relevant features of Modelica. Furthermore, in our experience, many Modelica models require specific numerical algorithms, which could cause further challenges.

Looking at the eFMI standard (Lenord et al. 2021), a representation reminiscent to the Algorithm Code, however without implemented solver arithmetic, could pose as a valid mutual baseline, for code generation to various languages and tools. From a Modelica compiler perspective, this would represent the hybrid[4] DAE system after the symbolic transformation. In contrast to Base Modelica, this representation would not require any additional transformation process. However, numerical challenges, as mentioned before, would still not be resolved by this intermediate representation.

Utilizing Julias' SciML ecosystem, while using the Functional Mockup Interface instead of Modelica directly, is possible through the packages FMIFlux.jl[5] and FMI.jl[6], which build the foundation of NeuralFMUs. FMI.jl allows to load and simulate FMUs in Julia, while FMIFlux.jl manages the gradient calculation through the FMU model. However, since all object-oriented information from a Modelica model is removed during FMU generation, it is not feasible to unroll the ML model of the trained NeuralFMU back to the object-oriented components.

Although we focused this section on model transfer from Modelica to Julia, the Enzyme project offers a solution to calculate the relevant gradients directly on LLVM level, see (Moses and Churavy 2020). LLVM supports a wide range of languages, however, changes to existing Modelica-based tools, that we cannot evaluate, might be needed. Furthermore, dedicated training frameworks, possibilities to define ML models, etc., within the Modelica tools would still be missing.

3.2 PeN-ODE architectures

Many hybrid model architectures, such as the NeuralFMU with its FMU foundation, grant access to the complete set of states, state derivatives, and algebraic quantities of the FPM. Consequently, hybrid models using arbitrary FPM signals in the ML model can be derived. Therefore, NeuralFMUs can be reused within other simulation models as a single component. However, separating the ML part from the physical model and re-introducing it in the original model and, therefore, also reusing partial models is not feasible. From an object-oriented modeling perspective, the use of arbitrary model quantities contradicts the paradigm of knowledge encapsulation, see (Cellier 1996). Consequently, the ML part of a hybrid component in an object-oriented model can only be supplied with internal quantities or existing interfaces of the model or additional signals must explicitly be exposed through model adaptions. Since encapsulation of knowledge enables the composition of hierarchical models, relying solely on available quantities renders reusing trained individual components possible.

As already stated in the PeN-ODE definition, arbitrary architectures for interaction between the FPM and ML part are applicable. Furthermore, ML models are adapted during their training phase and often optimized via hyperparameter optimization. Therefore, it is useful to only define the minimum necessary architectural parts, typically input and output features, within Modelica and shift all other architecture considerations to the ML side, where their optimization can take place. However, a Modelica compiler needs to know the relations of quantities to perform the symbolic transformation. The Modelica lan-

[3] https://github.com/SciML/BaseModelica.jl

[4] in this very case of this paper hybrid refers to continuous-discrete rather than to combination of FPM and ML

[5] https://github.com/ThummeTo/FMIFlux.jl

[6] https://github.com/ThummeTo/FMI.jl

DOI 10.3384/ecp218435

guage specification (Modelica Association 2023) provides the constructs of function inlining, which enables a placeholder functionality sufficient for the previously described considerations. Still, different types of augmenting the FPM equations exist.

The easiest way for setting up a hybrid architecture by place holder in a Modelica models is parallel execution on equation level, see Listing 1. Missing information in the equation is compensated by adding the function *fun_1*, which will later be replaced by some ML architecture. Only relevant inputs and outputs are provided in the Modelica model. Through the annotation "Inline=false" the function is not symbolically processed in the compiler and can be replaced during code generation.

Listing 1. parallel architecture on equation level

```
model SimpleParallelArchitecture
    function fun_1 "dummy function"
        input Real b, c, d;
        output Real dx;

    algorithm
        dx = b*c+d; // some operations that
        // will be replaced
    annotation(Inline=false);
    end fun_1;

    Real x "state variable";
    Real a,b,c,d;
    ...

equation
    ...
    // adding missing parts of the equation
    // by a Neural Network
    der(x) = a * x +  fun_1(b,c,d);
    ...
end SimpleParallelArchitecture;
```

The parallel architecture offers a coherent augmentation of a physical equation. However, to enable arbitrary architectures on ML side, further adaptions to existing model equations are required. This is done by splitting the relation of interest into two parts, as shown in Listing 2.

Listing 2. universal architecture on equation level

```
model UniversalArchitecture
    Real a "some algebraic qoi";
    Real a_fpm "the known relation of the
        qoi";
    Real b,c,d,x; // some additional
        variables
    ...
equation
    ...
    // a = b*x; // original equation
    a_fpm = b*x;
    a = fun_2(a_fpm, c,d);
    a = sin(time);
end UniversalArchitecture;
```

A detailed discussion on combination of FPM and ML, though in the context of NeuralFMUs, can be found in (Thummerer and Mikelsons 2025).

The relation between the algebraic variable a and the quantities b and x in Listing 2 shall be enhanced by some data-driven approach. Therefore, the first principle knowledge is rearranged into a new variable *a_fpm*. The original equation is replaced by the place holder function *fun_2*, that utilizes the existing knowledge of *a_fpm* and further employs the information from variables c and d. On ML side arbitrary combinations for interaction of the variables are possible, including the parallel architecture described before. Since the augmentation takes place on equation level rather than variable level, non-linear systems can arise from the function inline annotation during the symbolic transformation of the model. This is depicted in Listing 2 by explicitly providing a value to variable a, since time is the independent variable of a Modelica model. Solving the nonlinear system introduces additional evaluations during simulation. This is necessary because the placeholders used at this stage do not yet contain the actual relationships, which are only incorporated at a later point in the workflow. Obviously, on the machine learning side, the training framework must be capable of solving the nonlinear system and computing the corresponding gradients. To our current understanding, there are no further limitations imposed by Modelica itself, since the language specification has a rich extensiveness, like providing derivative annotations for the place holder function, et cetera.

If the architecture of the ML model is known before training, an implementation in Modelica could be possible. Some approaches will be discussed in the next section. However, since all equations from the ML model are processed by a Modelica compiler, additional effort is added to the symbolic transformation. Furthermore, the benefit of the symbolic representation of the ML model within Modelica remains unclear.

With experience, at least with ModelingToolkit.jl, we would recommend to only define the minimum required interfaces in Modelica and process all other parts in the ML environment. Having trainable parameters available as arrays, contrary to current Modelica compiler implementations, provides a convenient workflow in combination with hyperparameter optimization.

3.3 Representation of hybrid architecture in Modelica

As stated before, the final ML model is typically derived throughout the training in a hyperparameter optimization. Once the final structure has been established, the ML model and parameters must be reintegrated to the Modelica environment. Since Neural Network are today's dominating approach, we will limit approaches to this ML model type.

The most naive approach is implementing the corresponding tensor operations directly as a Modelica model, since the Modelica language provides features for multidimensional arrays. However, this is typically limited to simple architectures such as multilayer perceptions

(MLPs). A more sophisticated approach, while keeping Modelica language features, is provided in the NeuralNetwork Library[7] developed at Hochschule Bielefeld , that allows to automatically generate a model from Tensorflow. However, since array equations are, as of today, scalarized throughout the model transformation in a Modelica compiler, large Neural Networks might cause high compilation times or even overstrain available computer memory. Certainly, algorithms for keeping array structures throughout the symbolic pipeline of a Modelica compiler have been developed and will mitigate those issues, e.g. (Abdelhak, Casella, and Bachmann 2023), (Otter and Elmqvist 2017), (Fioravanti et al. 2023). Nonetheless, as stated before, the benefit of symbolic neural network representation in a Modelica model remains unclear.

Alternatively, standardized formats for exchanging deep learning models among tools, such as the Open Neural Network Exchange (ONNX) or the Neural Network Exchange Format (NNEF), could be utilized, for example the SmartInt Library[8] , which provides ONNX support using Modelica's "external C" feature. Therefore, Neural Networks with high number of parameters are be applicable, without any impact on a Modelica compiler.

3.4 Summary on Findings

There exist a wide range of different solution approaches which, in combination, help to prepare, create, and implement PeN-ODEs in Modelica. Modelica language features can be used to define interfaces in a convenient way and libraries and tools are available to embed ML models in Modelica. Nonetheless, the transfer of the (processed) Modelica model to a training environment such as Julia, poses an unsolved challenge. Once Base Modelica, is available, its application with ModelingToolkit.jl should be further investigated. Having a new implementation, positioned between Base Modelica (flattened) and the Algorithm Code from eFMI (including solver arithmetic) could pose a useful new representation, also for other applications. Further experiments with Modia or OM.jl might enable a first integrated workflow bridging the gap between Modelica and Julia, at the cost of limitations from the machine learning environment, due to their dependency on an old version of the Julia language.

We are confident, once PeN-ODEs have proven their usefulness, even against the current issues, a joint solution for bridging the gap can be achieved, suitable for users and tool vendors at the same time.

4 Scara-Robot Example

Based on the previously described solution approaches for integration PeN-ODEs in a Modelica workflow, we implemented an example application featuring a SCARA robot. However, since there exists no up-to-date solution for bridging the gap from Modelica to Julia while preserving the object-oriented structure of the model, we bypass this step in that we start from an object-oriented model within ModelingToolkit.jl. We hope that this example will encourage people to approach the findings from the previous chapter and to apply the idea of Modelica-based PeN-ODEs.

4.1 Overview on the SCARA Robot Example

The example is a modification of the NeuralFMU workshop (Thummerer and Mikelsons 2024) adapted for object-oriented application. A SCARA robot is used to write text on a sheet of paper by actuating its two revolute joints. A marker pen is mounted at the end effector, which can be pressed against the paper for writing or lifted for non-writing movements. The joints are actuated by DC motors and controlled using simple proportional controllers. An inverse kinematics model converts tabular end-effector position data – linearly interpolated between setpoints – into the corresponding joint angles.

The SCARA robot is modeled as 2D mechanical system and the writing process is modeled by applying a defined normal force, rather than implementing actual pen movement and collision considerations. Most importantly, no friction between the paper and the pen is model, but is supposed to be learned within the PeN-ODE. An equivalent Modelica model is illustrated in Figure 5. As reference data, simulation results from the robot containing Stribeck friction between end effector and paper, writing the words "train" and "validate" are available. To render a realistic scenario, we will only use motor currents as the available measurement outputs, since those signals are commonly available quantities and most importantly contain the friction force resp. acceleration information.

4.2 Model generation

As already mentioned at the beginning of this section, we employ ModelingToolkit.jl, to actually start from an object-oriented simulation model of the SCARA robot. Apart from own model implementations, models from the Modeling Toolkit Standard Library[9] were employed. Utilizing the Modelica PlanarMechanics library (Zimmer 2012) as a blueprint, necessary components for the mechanical system were implemented in ModelingToolkit.jl. All files reproducing the example can be found in the corresponding github repository[10]

As proposed previously, we insert a place holder function in the end effector model and consider only the known quantities within the component, i.e. end effector velocity in x- and y-direction and the normal force to the paper plane, to allow for the reuse of the trained subsystem, see Listing 3. Please note that "$\sim$" is used as the equality sign in ModelingToolkit.jl and the Modelica "der()"-operator corresponds to the "D()"-operator in the listing. NN_dummy refers to the place holder function that will be replaced after code generation. Its implementation is

[7] https://github.com/AMIT-HSBI/NeuralNetwork
[8] https://github.com/xrg-simulation/SMArtInt
[9] https://github.com/SciML/ModelingToolkit.jl
[10] https://github.com/AndreasHofmann217/ScaraRobotExample

DOI 10.3384/ecp218435

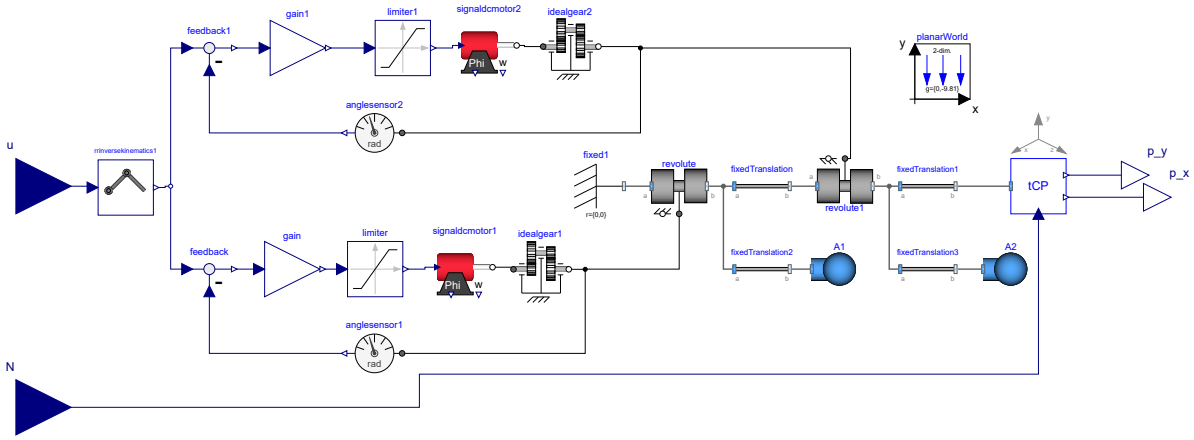


Figure 5. Illustration of the SCARA robot as Modelica model

similar to the previously mentioned workflow with annotation "Inline=false" from the Modelica specification.

ModelingToolkit.jl does generate an ODE system for models that can be represented by a structure incidence matrix with lower triangular form after transformation and a DAE system in mass matrix form otherwise. For the SCARA robot model a DAE system is generated, whose code is further adapted by us. The mass matrix, parameter information, and modified right-hand side functions are exported to a new Julia file, that will later be used within our training pipeline.

Listing 3. trainable end effector model

```
@mtkmodel TrainableTCP begin
    @variables begin
        x(t)
        y(t)
        vx(t)
        vy(t)
        f(t)[1:2]
    end
    @components begin
        frame_a = Frame()
        zForceIn = Blocks.RealInput()
        posXOut = Blocks.RealOutput()
        posYOut = Blocks.RealOutput()
    end
    @equations begin
        frame_a.tau ~ 0.0
        frame_a.fx ~ f[1]
        frame_a.fy ~ f[2]
        frame_a.x ~ posXOut.u
        frame_a.y ~ posYOut.u
        frame_a.x ~ x
        frame_a.y ~ y
        D(x) ~ vx
        D(y) ~ vy
        f ~ NN_dummy(1.0,vx,vy,zForceIn.u)
    end
end
```

4.3 ML model and PeN-ODE training

For learning the friction forces from the applied normal force and the velocities at the end effector, a Neural Network is employed as ML model. It is built using an MLP with three hidden layers, 32 neurons and tanh as the activation function. To scale input features (velocity in x- and y-direction, normal force onto the paper) as well as the output features (friction forces in x- and y-direction), a layer adding z-score standardization resp. another layer providing inverse transformation for the output features are added. However, those layers do not contain fixed values for means and standard deviations but are rather trainable parameters as well. As a last measure, a trainable gate, initialized with start value of zero is added. This guarantees that the randomly initialized network does provide stable output during the first training steps. The complete architecture is depicted in Figure 6.

The model is trained using a single trajectory, with the word "train" written by the SCARA robot, cf. Figure 7

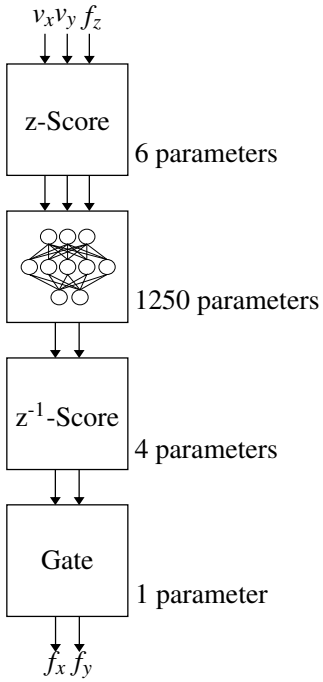


Figure 6. Employed ML model for the SCARA robot and the individual number of trainable parameters for the individual parts

Currently, only single-step solvers can be employed for the hybrid DAE of the SCARA during gradient calculation[11]. Therefore, we use Rodas5P, a fifth order Rosenbrock method from within DifferentialEquations.jl, as numerical method.

The trajectory is split into samples of 50 ms length and gradients are calculated over the solution of randomly selected single samples. The sampling is done for various reasons. On the one side, it helps to increase training speed, since gradients over a short elapsed time can be calculated faster and have lower requirements for computer hardware. Additionally, using stochastic examples helps to better progress in the beginning of the optimization, while calculating only gradients for a few examples, cf. (Goodfellow, Bengio, and Courville 2016). On the other side, since the calculated gradients combine all deviations across the simulated period, shorter time spans support capturing highly dynamic details. The Adam optimizer (Kingma and Ba 2015) with default settings was used for updating parameters. For loss accumulating Mean Absolute Error (MAE) aggregation was chosen, considering motor currents as quantities as available signals, as stated above. As algorithm for deriving the gradients of the DAE ReverseDiffAdjoint from the package SciMLSensitivity.jl (Rackauckas, Ma, et al. 2021) was selected.

The system was trained for 5000 steps on a laptop with i7-11850H CPU, 32 GB RAM and Win11 as OS, which took about two hours. Details on the used Julia packages can be found in the corresponding GitHub repository.

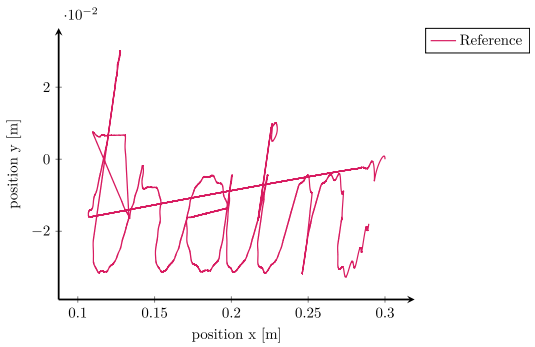


Figure 7. Training trajectory (end effector) for the SCARA robot (without distinguishing actually applied forces for writing)

4.4 Training results

Please note that no extensive effort, e.g. through hyperparameter analysis, was spent on the model. The results are purely derived to motivate and validate a PeN-ODE workflow, rather than to provide the best possible model outcome. Nonetheless, to anticipate some results of the trained model, the MAE for the combined model states is reduced by a factor greater than 4, while the Mean Squared Error (MSE) is reduced by a factor larger than 11, each for the chosen test trajectory.

As test data for the SCARA robot application, the word "validate" has been selected. In Figure 8, the model with and without training are compared to the reference data for the end effector position. The further the SCARA arms is extended, the more impact from friction is visible. The common error metrics MAE and MSE show a significant reduction in deviations. An overview for the aggregated model and individual states is collected in Table 1.

Table 1. Error metrics of the trained PeN-ODE compared to the original model (in brackets) for the test data trajectory.

state	MAE	MSE
complete model	7.76e-03	4.47e-04
$[-]$	(3.61e-02)	(5.19e-03)
$motor1.\varphi$	1.69e-04	1.17e-07
$[rad]$	(1.26e-03)	(2.87e-06)
$motor1.i$	1.40e-02	1.04e-03
$[A]$	(9.39e-02)	(1.61e-02)
$motor1.\omega$	5.59e-03	2.34e-04
$[rad/s]$	(1.62e-02)	(1.27e-03)
$motor2.\varphi$	3.40e-04	3.92e-07
$[rad]$	(1.67e-03)	(5.31e-06)
$motor2.i$	1.63e-02	9.58e-04
$[A]$	(7.60e-02)	(1.11e-02)
$motor2.\omega$	1.01e-02	4.55e-04
$[rad/s]$	(2.76e-02)	(2.70e-03)

A more detailed insight on the expressiveness of the PeN-ODE can be gained from the plots of the motor currents, see Figure 9 and Figure 10, as well as from the applied friction forces in the model, cf. Figure 11 and Figure 12. The model still holds some limitation w.r.t. force in x-direction, probably limited by the scalar gate. Furthermore, the friction forces during the start of the simulation, although no normal force is present, would require some further investigation on the training. Also, no further study on the impact of partly implementing physical effects, such as explicitly providing Coloumb friction, was conducted. This remains to be investigated in detail in the future. Yet, a first study on partially modelling friction, based on NeuralFMUs, was executed by Thummerer, Kolesnikov, et al. (2023).

From an object-oriented modeling perspective, the machine learning component can be seamlessly reintegrated into the end effector by replacing the placeholder function with the actual implementation containing the trained parameters. As previously noted, keeping these parameters independent of the compiler contributes to maintaining fast compilation times. Since only internal variables and inputs of the end effector component were utilized within the hybrid architecture, the component remains fully reusable in other applications – such as robots with different mechanical structures – without requiring further modifications.

[11] this is due to the discontinuous setpoint data and a bug during initialization of the DAE

DOI
10.3384/ecp218435

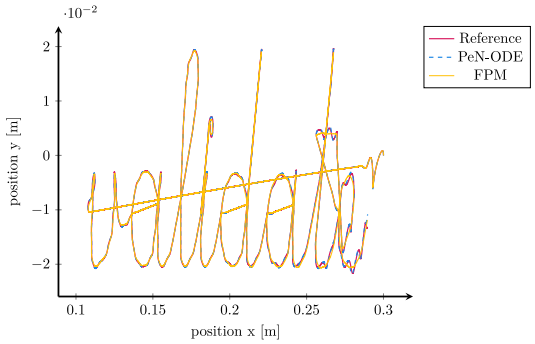


Figure 8. Comparison of the trained and untrained model writing the word validate (without distinguishing actually applied force)

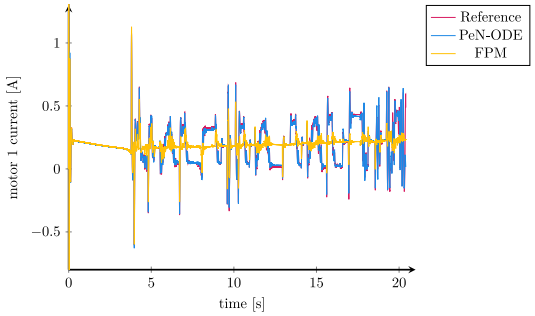


Figure 9. Comparison of Motor currents at Motor 1

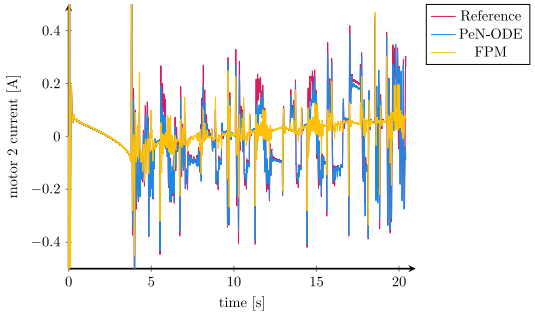


Figure 10. Comparison of Motor currents at Motor 2

5 Summary and Outlook

Although Modelica and the corresponding tools are natively incompatible with ML training environments such as the Julia programming language, many solution approaches for bridging the gap towards creating, training and applying of PeN-ODEs exist. Once a model is made available in Julia, all necessary steps can be applied, which was shown for the SCARA robot.

From the interfacing point of view, having an intermediate representation of the derived DAE system from a Modelica compiler, reminiscent of Base Modelica, would allow a universal starting point for arbitrary ML environments in Julia, Python or other languages but also for other applications. Furthermore, Base Modelica, once available, could offer a convenient way from Modelica to Julia, though it needs to be investigated if ModelingToolkit.jl is capable of supporting all relevant Modelica language features. Nonetheless, also tool vendor-specific implementations of Modelica compilers and simulation capabilities and particularities of their individual simulation runtimes might still pose challenges.

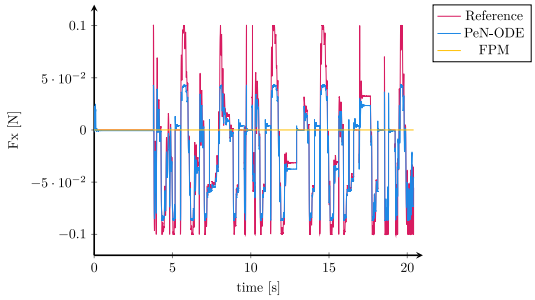


Figure 11. Comparison of friction force in x-direction

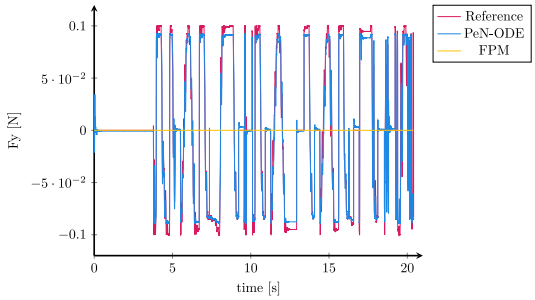


Figure 12. Comparison of friction force in y-direction

Apart from interfacing, new challenges, such as appropriate training strategies and best suitable ML models arise. While the example model was trained on a random selection of data samples, more sophisticated approaches such as dedicated scheduling algorithms for picking worst elements from the training set or different strategies, such as multiple shooting, growing horizon, et cetera, might accelerate and improve training.

In particular, the aspects of reusability of PeN-ODE components can facilitate the object-oriented approach from methods such as NeuralFMU, which in contrast does benefit from tool independence and bypasses the interfacing issues. A Modelica-based approach would allow to easily compose hierarchical systems of high fidelity PeN-ODEs.

Furthermore, the appropriate ratio of embedded physically-based white box models, such as known parts of friction, is an open research question for all hybrid models. Further studies are required to provide some indication, especially based on real-world problems.

Acknowledgements

This work has been partially funded through the ITEA4-project OpenSCALING (`www.openscaling.org`) by the German Federal Ministry of Education and Research (BMBF) under the grant number FKZ 01IS23062H.

References

Abdelhak, Karim, Francesco Casella, and Bernhard Bachmann (2023). "Pseudo Array Causalization". In: *Proceedings of the 15th International Modelica Conference 2023*. DOI: 10.3384/ecp204177.

Bruder, Frederic and Lars Mikelsons (2021). "Modia and Julia for Grey Box Modeling". In: *Proceedings of the 14th International Modelica Conference 2021*. DOI: 10.3384/ecp2118187.

Cellier, François (1996). "Object-Oriented Modeling: Means For Dealing With System Complexity". In: *Proceedings of the 15th Benelux Meeting on Systems and Control*.

Chen, Ricky T. Q. et al. (2018). "Neural Ordinary Differential Equations". In: *Proceedings in Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*. DOI: 10.48550/arXiv.1806.07366.

Elmqvist, Hilding et al. (2021). "Modia – Equation Based Modeling and Domain Specific Algorithms". In: *Proceedings of the 14th International Modelica Conference 2021*. DOI: 10.3384/ecp2118173.

Fioravanti, Massimo et al. (2023-07). "Array-Aware Matching: Taming the Complexity of Large-Scale Simulation Models". In: *ACM Trans. Math. Softw.* DOI: 10.1145/3611661.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press.

Kamp, Tobias, Johannes Ultsch, and Jonathan Brembeck (2023). "Closing the Sim-to-Real Gap with Physics-Enhanced Neural ODEs". In: *Proceedings of the 20th International Conference on Informatics in Control, Automation and Robotics*. DOI: 10.5220/0012160100003543.

Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization." In: *ICLR (Poster)*. URL: http://dblp.uni-trier.de/db/conf/iclr/iclr2015.html#KingmaB14.

Kurzbach, Gerd et al. (2023). "Design proposal of a standardized Base Modelica language". In: *Proceedings of the 15th International Modelica Conference 2023*. DOI: 10.3384/ecp204469.

Lenord, Oliver et al. (2021). "eFMI: An open standard for physical models in embedded software". In: *Proceedings of 14th Modelica Conference 2021*. DOI: doi.org/10.3384/ecp2118157.

Ma, Yingbo et al. (2021). *ModelingToolkit: A Composable Graph Transformation System For Equation-Based Modeling*. arXiv: 2103.05244 [cs.MS].

Modelica Association (2023). *Modelica – A Unified Object-Oriented Language for Systems Modeling. Language Specification Version 3.6*. Tech. rep. Linköping: Modelica Association.

Moses, William and Valentin Churavy (2020). "Instead of Rewriting Foreign Code for Machine Learning, Automatically Synthesize Fast Gradients". In: *Advances in Neural Information Processing Systems*. Vol. 33. DOI: 10.48550/arXiv.2010.01709.

Oliveira, R. (2004). "Combining first principles modelling and artificial neural networks: a general framework". In: *Computers Chemical Engineering* 28.5. DOI: 10.1016/j.compchemeng.2004.02.014.

Otter, Martin and Hilding Elmqvist (2017). "Transformation of Differential Algebraic Array Equations to Index One Form". In: *Proceedings of the 12th International Modelica Conference 2017*. DOI: 10.3384/ecp17132565.

Otter, Martin, Hilding Elmqvist, et al. (2019). "Thermodynamic Property and Fluid Modeling with Modern Programming Language Constructs". In: *Proceedings of the 13th International Modelica Conference 2021*. DOI: 10.3384/ecp19157589.

Quaghebeur, Ward, Ingmar Nopens, and Bernard De Baets (2021). "Incorporating Unmodeled Dynamics Into First-Principles Models Through Machine Learning". In: *IEEE Access* 9. DOI: 10.1109/ACCESS.2021.3055353.

Rackauckas, Christopher, Yingbo Ma, et al. (2021). *Universal Differential Equations for Scientific Machine Learning*. arXiv: 2001.04385 [cs.LG]. URL: https://arxiv.org/abs/2001.04385.

Rackauckas, Christopher and Qing Nie (2017). "DifferentialEquations.jl–a performant and feature-rich ecosystem for solving differential equations in Julia". In: *Journal of Open Research Software* 5.1.

Sapienza, Facundo et al. (2024). *Differentiable Programming for Differential Equations: A Review*. arXiv: 2406.09699 [math.NA]. URL: https://arxiv.org/abs/2406.09699.

Sorourifar, Farshud et al. (2023). "Physics-Enhanced Neural Ordinary Differential Equations: Application to Industrial Chemical Reaction Systems". In: *Industrial & Engineering Chemistry Research* 62.38. DOI: 10.1021/acs.iecr.3c01471.

Su, Hong-Te et al. (1992). "Integrating Neural Networks with First Principles Models for Dynamic Modeling". In: *IFAC Proceedings Volumes* 25.5. 3rd IFAC Symposium on Dynamics and Control of Chemical Reactors, Distillation Columns and Batch Processes (DYCORD+ '92), Maryland, USA, 26-29 April. DOI: 10.1016/S1474-6670(17)51013-7.

Thummerer, Tobias, Josef Kircher, and Lars Mikelsons (2021). "NeuralFMU: Towards Structural Integration of FMUs into Neural Networks". In: *Proceedings of 14th Modelica Conference 2021*. DOI: 10.3384/ecp21181297.

Thummerer, Tobias, Artem Kolesnikov, et al. (2023). "Paving the way for Hybrid Twins using Neural Functional Mock-Up Units". In: *Proceedings of the 15th International Modelica Conference 2023*. DOI: 10.3384/ecp204141.

Thummerer, Tobias and Lars Mikelsons (2023). *Using NeuralODEs in real life applications*. https://pretalx.com/juliacon2023/talk/EWL3LC/. Accessed: 2025-04-08.

Thummerer, Tobias and Lars Mikelsons (2024-08). *Scientific Machine Learning using Functional Mock-Up Units | Thummerer, Mikelsons | JuliaCon 2024*. https://www.youtube.com/watch?v=sQ2MXSswrSo.

Thummerer, Tobias and Lars Mikelsons (2025). *Learnable Interpretable Model Combination in Dynamical Systems Modeling*. URL: https://arxiv.org/abs/2406.08093.

Thummerer, Tobias, Hans Olsson, et al. (2025). "LS-SA: Developing an FMI layered standard for holistic efficient sensitivity analysis of FMUs". In: *Proceedings of the 16th International Modelica Conference 2025*.

Tinnerholm, John et al. (2021). "OpenModelica.jl: A modular and extensible Modelica compiler framework in Julia targeting ModelingTookit.jl". In: *Proceedings of 14th Modelica Conference 2021*. DOI: 10.3384/ecp21181109.

Zimmer, Dirk (2012). "A Planar Mechanical Library for Teaching Modelica". In: *Proceedings of the 9th International Modelica Conference*. DOI: 10.3384/ecp12076681.

Efficient Training of Physics-enhanced Neural ODEs via Direct Collocation and Nonlinear Programming

Linus Langenkamp [1] Philip Hannebohm [1] Bernhard Bachmann [1]

[1]Institute for Data Science Solutions, Bielefeld University of Applied Sciences and Arts, Germany,
{first.last}@hsbi.de

Abstract

We propose a novel approach for training Physics-enhanced Neural ODEs (PeN-ODEs) by expressing the training process as a dynamic optimization problem. The full model, including neural components, is discretized using a high-order implicit Runge-Kutta method with flipped Legendre-Gauss-Radau points, resulting in a large-scale nonlinear program (NLP) efficiently solved by state-of-the-art NLP solvers such as Ipopt. This formulation enables simultaneous optimization of network parameters and state trajectories, addressing key limitations of ODE solver-based training in terms of stability, runtime, and accuracy. Extending on a recent direct collocation-based method for Neural ODEs, we generalize to PeN-ODEs, incorporate physical constraints, and present a custom, parallelized, open-source implementation. Benchmarks on a Quarter Vehicle Model and a Van-der-Pol oscillator demonstrate superior accuracy, speed, generalization with smaller networks compared to other training techniques. We also outline a planned integration into OpenModelica to enable accessible training of Neural DAEs.

Keywords: Physics-enhanced Neural ODEs, Dynamic Optimization, Nonlinear Programming, Modelica, Neural ODEs, Universal Differential Equations

1 Introduction

Growing access to real-world data and advances in computational modeling have opened new possibilities for combining measured data with physics-based models. Neural Ordinary Differential Equations (NODEs) (Chen et al. 2018) represent a significant advancement in merging data-driven machine learning with physics-based modeling. By replacing the dynamics of an ODE with a neural network

$$\dot{\boldsymbol{x}}(t) = NN_{\boldsymbol{p}}(\boldsymbol{x}(t), \boldsymbol{u}(t), t), \tag{1}$$

where $\boldsymbol{x}(t)$ are states, $\boldsymbol{u}(t)$ is a fixed input vector, and $\boldsymbol{p}$ are the neural network parameters, NODEs bridge the gap between traditional differential equations and modern deep learning. After obtaining a NODE and given an initial condition $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$, the state trajectory is reconstructed by simulation with an arbitrary ODE solver

$$\boldsymbol{x}(t) := \text{ODESolve}\left(NN_{\boldsymbol{p}}(\boldsymbol{x}(t), \boldsymbol{u}(t), t), \boldsymbol{x}_0\right). \tag{2}$$

However, learning the full dynamics can be unstable, requires a lot of data, and can suffer from poor extrapolation (Kamp, Ultsch, and Brembeck 2023). As a result, hybrid modeling is an emerging field that combines the flexibility of neural networks with known physics and first principle models. Extensions like Universal Differential Equations (UDEs) (Rackauckas et al. 2020) or Physics-enhanced Neural ODEs (PeN-ODEs) (Kamp, Ultsch, and Brembeck 2023; Sorourifar et al. 2023) generalize this paradigm, allowing domain-specific knowledge to be incorporated into the model while still learning observable but unresolved effects. These approaches have demonstrated great success in various fields, including vehicle dynamics (Bruder and Mikelsons 2021; Thummerer, Stoljar, and Mikelsons 2022), chemistry (Thebelt et al. 2022), climate modeling (Ramadhan et al. 2023), and process optimization (Misener and L. Biegler 2023).

Training neural components typically involves simulating (2) for some initial parameters $\boldsymbol{p}$ and then propagating sensitivities of the ODE solver backward in each iteration. Afterward, the parameters are updated via gradient descent. This process is computationally expensive and results in long training times (Lehtimäki, Paunonen, and Linne 2024; Roesch, Rackauckas, and Stumpf 2021; Shapovalova and Tsay 2025), as explicit integrators are low-order and unstable, requiring small step sizes, while stable, implicit integrators involve solving nonlinear systems at each step, thus being computationally demanding.

To address these enormous training times several alternative procedures have been proposed: in (Roesch, Rackauckas, and Stumpf 2021) a collocation technique is introduced, which approximates the right hand side (RHS) of an ODE from data. The NN is then trained on the approximations with standard training frameworks. Further, in (Lehtimäki, Paunonen, and Linne 2024) model order reduction is used to accurately simulate the dynamics in low-dimensional subspaces. Very recent work presented in (Shapovalova and Tsay 2025) introduced global direct collocation with Chebyshev nodes, a method originating from dynamic optimization for optimal control and parameter optimization, for training Neural ODEs. The approach reduces the continuous training problem to a large finite dimensional nonlinear program (NLP) and shows fast and stable convergence, demonstrated on a typical problem, the Van-der-Pol oscillator.

In Modelica-based workflows, the training of Neural ODEs is typically performed externally by exporting a Functional Mock-Up Unit (FMU), subsequently training it in Python or Julia using standard machine learning frameworks and ODE solvers, and finally re-importing the hybrid model. While this approach introduces external dependencies and additional transformation steps, the *NeuralFMU* workflow (Thummerer, Stoljar, and Mikelsons 2022) demonstrates a practical method for integrating hybrid models into real-world applications.

Building on recent advances in direct collocation-based training of Neural ODEs (Shapovalova and Tsay 2025), we significantly extend the approach to PeN-ODEs. We formulate the training process as a dynamic optimization problem and discretize both neural and physical components using a stable, high-order implicit collocation scheme at flipped Legendre-Gauss-Radau (fLGR) points. This results in a large but structured NLP, allowing efficient, simultaneous optimization of states and parameters. Our custom, parallelized implementation leverages second-order information and the open-source NLP solver Ipopt (Wächter and L. T. Biegler 2006). It is designed for a future integration into the open-source modeling and simulation environment OpenModelica (Fritzson, Pop, Abdelhak, et al. 2020), thus providing an accessible training environment independent of external tools.

2 Dynamic Optimization for NODEs

In this section, we introduce a general class of dynamic optimization problems (DOPs) and formulate training for both NODEs and PeN-ODEs as instances of this class. We then discuss the transcription of the continuous problem into a large-scale nonlinear optimization problem (NLP). Finally, necessary considerations and key challenges are presented.

2.1 Generic Problem Formulation

Consider the DOP

$$\min_{\boldsymbol{p}} M(\boldsymbol{x}(t_0),\boldsymbol{x}(t_f),\boldsymbol{p}) + \int_{t_0}^{t_f} L(\boldsymbol{x}(t),\boldsymbol{p},t)\,\mathrm{d}t \tag{3a}$$

s.t.

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t),\boldsymbol{p},t) \qquad \forall t \in T \tag{3b}$$

$$\boldsymbol{g}^L \le \boldsymbol{g}(\boldsymbol{x}(t),\boldsymbol{p},t) \le \boldsymbol{g}^U \qquad \forall t \in T \tag{3c}$$

$$\boldsymbol{r}^L \le \boldsymbol{r}(\boldsymbol{x}(t_0),\boldsymbol{x}(t_f),\boldsymbol{p}) \le \boldsymbol{r}^U \tag{3d}$$

for a fixed time horizon $T = [t_0, t_f]$ with time variable $t \in T$. The states of the system are given by $\boldsymbol{x} : T \to \mathbb{R}^{d_x}$ and the goal is to find optimal time-invariant parameters $\boldsymbol{p} \in \mathbb{R}^{d_p}$, such that the objective (3a) becomes minimal and the constraints (3b)–(3d) are satisfied. These constraints are divided into the ODE (3b) and *path constraints* (3c), which both must be satisfied at all times on time horizon T, as well as *boundary constraints* (3d), which must only hold at the initial and final time points t_0, t_f. The objective is composed of a *Mayer* term M, that defines a cost at the boundary of T, and a *Lagrange* term L, that penalizes an accumulated cost over the entire time horizon. To ensure compatibility with typical nonlinear optimizers, all model functions must be twice continuously differentiable. This includes neural networks, their activation functions, and error measures. For completeness, the bounds of the constraints are given as $\boldsymbol{g}^L, \boldsymbol{g}^U \in (\mathbb{R} \cup \{-\infty, \infty\})^{d_g}$ and $\boldsymbol{r}^L, \boldsymbol{r}^U \in (\mathbb{R} \cup \{-\infty, \infty\})^{d_r}$.

2.2 Reformulation of PeN-ODE Training

PeN-ODEs embed one or more NNs with parameters $\boldsymbol{p}$ into known, possibly equation-based, dynamics $\dot{\boldsymbol{x}}(t) = \hat{\boldsymbol{\phi}}(\boldsymbol{x},\boldsymbol{u},t)$. The resulting differential equation has the form

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{u},t,NN_{\boldsymbol{p}}(\boldsymbol{x},\boldsymbol{u},t)), \tag{4}$$

where $\boldsymbol{u} : T \to \mathbb{R}^{d_u}$ is a fixed input vector and $NN_{\boldsymbol{p}}$ are enhancing NNs. This formalism aims to enhance systems that already express dynamics based on first principles, by further incorporating data-driven observables in form of neural components. Clearly, these components need not be NNs in general, and can be any parameter dependent expression. With additional information about the problem, one could use a polynomial, rational function, sum of radial basis functions or Fourier series.

The subsequent considerations also apply to the training of NODEs, where the goal is to learn the full dynamics without relying on a first principle model. This is evident from the fact that NODEs are a subclass of PeN-ODEs with

$$\dot{\boldsymbol{x}}(t) = NN_{\boldsymbol{p}}(\boldsymbol{x},\boldsymbol{u},t) = \boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{u},t,NN_{\boldsymbol{p}}(\boldsymbol{x},\boldsymbol{u},t)). \tag{5}$$

In this paper, we propose a formulation for training PeN-ODEs as a DOP (3a)–(3d), using known data trajectories $\hat{\boldsymbol{q}}$ and the corresponding predicted quantity $\boldsymbol{q}$. The DOP takes the form

$$\min_{\boldsymbol{p}} \int_{t_0}^{t_f} E\left(\boldsymbol{q}(\boldsymbol{x},\boldsymbol{u},t,NN_{\boldsymbol{p}}(\boldsymbol{x},\boldsymbol{u},t)),\hat{\boldsymbol{q}}(t)\right)\,\mathrm{d}t \tag{6a}$$

s.t.

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{\phi}(\boldsymbol{x},\boldsymbol{u},t,NN_{\boldsymbol{p}}(\boldsymbol{x},\boldsymbol{u},t)) \quad \forall t \in T \tag{6b}$$

for some smooth error measure E, e.g. the squared 2-norm $E(\boldsymbol{q},\hat{\boldsymbol{q}}) = \|\boldsymbol{q} - \hat{\boldsymbol{q}}\|_2^2$. This formulation represents the minimal setup.

By further incorporating the generic constraints (3c) and (3d), it is possible to impose an initial or final condition on the states as well as enforce desired behavior. For example, consider a NN approximation of a force element NN_F, with no force acting in its resting position. Therefore, the NN should have a zero crossing, i.e. $NN_F(0) = 0$. This can be trivially formulated as a constraint, without introducing a penalty term that may distort the optimization as in standard unconstrained approaches like (Kamp,

DOI 10.3384/ecp218445

Ultsch, and Brembeck 2023). Thus, the optimizer can handle the constraint appropriately.

As the loss in (6a) is a continuous-time integral, it enables an accurate and stable approximation using the same discretization employed for the system dynamics. In contrast to MSE loss as in (Shapovalova and Tsay 2025), which effectively corresponds to a first-order approximation of the integral, our formulation benefits from high-order quadrature, potentially preserving the accuracy of the underlying discretization.

2.3 Transcription with Direct Collocation

In the following, the general DOP (3a)–(3d) is reduced to a NLP using orthogonal direct collocation. Direct collocation approaches have proven to be highly efficient in solving DOPs and are implemented in a variety of free and commercial tools, such as PSOPT (Becerra 2010), CasADi (Andersson et al. 2019) or GPOPS-II (Patterson and A. V. Rao 2014), as well as in Modelica-based environments like OpenModelica (Ruge et al. 2014) or JModelica (Magnusson and Åkesson 2015). While OpenModelica only supports optimal control problems, the other frameworks allow for simultaneous optimization of static parameters. Recent work in the field of NODEs (Shapovalova and Tsay 2025) shows that learning the right hand side of small differential equations can be performed stably and efficiently using global collocation with Chebyshev nodes.

In direct collocation the states are approximated by piecewise polynomials, that satisfy the differential equation at so-called *collocation nodes*, usually chosen as roots of certain orthogonal polynomials. If the problem is smooth and with increasing number of collocation nodes, these methods achieve *spectral*, i.e. exponential, convergence to the exact solution. In this paper, the collocation nodes are chosen as the *flipped Legendre-Gauss-Radau points (fLGR)* rescaled from $[-1,1]$ to $[0,1]$. This rescaling is performed, so that the corresponding collocation method is equivalent to the Radau IIA Runge-Kutta method. Radau IIA has excellent properties, since it is A-, B- and L-stable and achieves order $2m-1$ for m stages or collocation nodes. These nodes c_j for $j = 1,\dots,m$ are given as the m roots of the polynomial $(1-t)P^{(1,0)}_{m-1}(2t-1)$, where $P^{(1,0)}_{m-1}$ is the $(m-1)$-th Jacobi polynomial with $\alpha = 1$ and $\beta = 0$. A detailed explanation of the method's construction based on quadrature rules is given in (Langenkamp 2024).

First, we divide the time horizon $[t_0, t_f]$ into $n+1$ intervals $[t_i, t_{i+1}]$ for $i = 0,\dots,n$ with length $\Delta t_i := t_{i+1} - t_i$. In each interval $[t_i, t_{i+1}]$ the collocation nodes $t_{ij} := t_i + c_j \Delta t_i$ for $j = 1,\dots,m_i$ as well as the first grid point $t_{i0} := t_i + c_j \Delta t_i$ with $c_0 = 0$ are added. Since the last node $c_{m_i} = 1$ is contained in any Radau IIA scheme, the last grid point of interval $i-1$ exactly matches the first grid point of interval i, i.e. $t_{i-1,m_{i-1}} = t_{i0}$. Furthermore, the states are approximated as $\boldsymbol{x}(t_{ij}) \approx \boldsymbol{x}_{ij}$ and for each interval i replaced by a Lagrange interpolating polynomial $\boldsymbol{x}_i(t) = \sum_{j=0}^{m_i} \boldsymbol{x}_{ij}\, l_j(t)$ of degree m_i, where

$$l_j(t) := \prod_{\substack{k=0 \\ k\neq j}}^{m_i} \frac{t - t_{ik}}{t_{ij} - t_{ik}} \quad \forall j = 0,\dots,m_i \tag{7}$$

are the Lagrange basis polynomials. Note that the parameters $\boldsymbol{p}$ are time-invariant and thus, need not be discretized. Each $\boldsymbol{x}_i$ must satisfy the differential equation (3b) at the collocation nodes t_{ij} and also match the initial condition $\boldsymbol{x}_{i0}$, which is given from the previous interval $i-1$. By differentiating we get the collocated dynamics

$$\mathbf{0} = \begin{bmatrix} D^{(1)}_{10} I & \dots & D^{(1)}_{1m_i} I \\ \vdots & \ddots & \vdots \\ D^{(1)}_{m_i 0} I & \dots & D^{(1)}_{m_i m_i} I \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{i0} \\ \vdots \\ \boldsymbol{x}_{im_i} \end{bmatrix} - \Delta t_i \begin{bmatrix} \boldsymbol{f}_{i1} \\ \vdots \\ \boldsymbol{f}_{im_i} \end{bmatrix} \tag{8}$$

with identity matrix $I \in \mathbb{R}^{d_x \times d_x}$, entries of the first differentiation matrix $D^{(1)}_{jk} := \frac{\mathrm{d}\tilde{l}_k}{\mathrm{d}\tau}(c_j)$, where

$$\tilde{l}_k(\tau) := \prod_{\substack{r=0 \\ r\neq k}}^{m_i} \frac{\tau - c_r}{c_k - c_r} \; \forall k = 0,\dots,m_i, \tag{9}$$

and the RHS of the ODE $\boldsymbol{f}_{ij} := \boldsymbol{f}(\boldsymbol{x}_{ij}, \boldsymbol{p}, t_{ij})$. The numerical values of $D^{(1)}_{jk}$ can be calculated very efficiently with formulas provided in (Schneider and Werner 1986).

Approximating L is analogous to discretizing the differential equation. This is done by replacing the integral with a Radau quadrature rule of the form

$$\int_{t_0}^{t_f} L(\boldsymbol{x}(t), \boldsymbol{p}, t)\,\mathrm{d}t \approx \sum_{i=0}^{n} \Delta t_i \sum_{j=1}^{m_i} b_j L(\boldsymbol{x}_{ij}, \boldsymbol{p}, t_{ij}), \tag{10}$$

where the quadrature weights are given by

$$b_j = \int_0^1 \prod_{\substack{k=1 \\ k\neq j}}^{m_i} \frac{\tau - c_k}{c_j - c_k}\,\mathrm{d}\tau \quad \forall j = 1,\dots,m_i. \tag{11}$$

M and the boundary constraints (3d) are approximated by replacing the values on the boundary with their discretized equivalents, i.e. $M(\boldsymbol{x}(t_0), \boldsymbol{x}(t_f), \boldsymbol{p}) \approx M(\boldsymbol{x}_{00}, \boldsymbol{x}_{nm_n}, \boldsymbol{p})$ and $\boldsymbol{r}(\boldsymbol{x}(t_0), \boldsymbol{x}(t_f), \boldsymbol{p}) \approx \boldsymbol{r}(\boldsymbol{x}_{00}, \boldsymbol{x}_{nm_n}, \boldsymbol{p})$, while the path constraints (3c) are evaluated at all nodes, i.e. $\boldsymbol{g}(\boldsymbol{x}(t_{ij}), \boldsymbol{p}, t_{ij}) \approx \boldsymbol{g}(\boldsymbol{x}_{ij}, \boldsymbol{p}, t_{ij})$.

2.4 Training with Nonlinear Programming

By flattening the collocated dynamics (8), we obtain the discretized DOP (3a)–(3d) of the form

$$\min_{\boldsymbol{x}_{ij}, \boldsymbol{p}} M(\boldsymbol{x}_{00}, \boldsymbol{x}_{nm_n}, \boldsymbol{p}) + \sum_{i=0}^{n} \Delta t_i \sum_{j=1}^{m_i} b_j L(\boldsymbol{x}_{ij}, \boldsymbol{p}, t_{ij}) \tag{12a}$$

s.t.

$$\mathbf{0} = \sum_{k=0}^{m_i} D_{jk}^{(1)} \boldsymbol{x}_{ik} - \Delta t_i \boldsymbol{f}(\boldsymbol{x}_{ij}, \boldsymbol{p}, t_{ij}) \quad \forall i, \forall j \geq 1 \quad (12b)$$

$$\boldsymbol{g}^L \leq \boldsymbol{g}(\boldsymbol{x}_{ij}, \boldsymbol{p}, t_{ij}) \quad \leq \boldsymbol{g}^U \quad \forall i, \forall j \geq 1 \quad (12c)$$

$$\boldsymbol{r}^L \leq \boldsymbol{r}(\boldsymbol{x}_{00}, \boldsymbol{x}_{nm_n}, \boldsymbol{p}) \leq \boldsymbol{r}^U \quad (12d)$$

This large-scale NLP (12a)–(12d) can be implemented and solved efficiently in nonlinear optimizers such as Ipopt (Wächter and L. T. Biegler 2006), SNOPT (P. E. Gill et al. 2007; Philip E. Gill, Murray, and Saunders 2005) or KNITRO (Byrd, Nocedal, and Waltz 2006). These NLP solvers exploit the sparsity of the problem as well as the first and second order derivatives of the constraint vector and objective function to converge quickly to a suitable local optimum.

The open-source interior-point method Ipopt requires the already mentioned first derivatives and, in addition, the Hessian of the augmented Lagrangian at every iteration. Since the derivatives only need to be evaluated at the collocation nodes, there is no need to propagate them as in traditional ODE solver-based training. Furthermore, training with ODE solvers usually limits itself to first order derivatives and therefore, does not utilize higher order information as in the proposed approach.

The resulting so-called *primal-dual* system is then solved using a linear solver for symmetric indefinite systems, such as the open-source solver MUMPS (Amestoy et al. 2001) or a proprietary solver from the HSL suite (HSL 2013), after which an optimization step is performed. This step updates all variables $\boldsymbol{x}_{ij}, \boldsymbol{p}$ simultaneously, allowing for direct observation and adjustment of intermediate values. In contrast, ODE solver training only captures the final result after integrating the dynamics over time, without the ability to directly influence intermediate states during the optimization process. Because this linear system must be solved in every iteration anyway, high order, stable, implicit Runge-Kutta collocation methods, e.g. Radau IIA, can be embedded with only limited overhead. As a result, the NLP formulation overcomes key limitations of explicit ODE solvers in terms of order, stability, and allowable step size. Moreover, since the solver performs primal and dual updates, the solution does not need to remain feasible during the optimization, in contrast to ODE solver approaches where the dynamics (3b) are enforced at all times through forward simulation. This results in both advantages and disadvantages: On the one hand, it enables more flexible and aggressive updates, potentially accelerating convergence. On the other hand, it may lead to intermediate solutions that temporarily violate physical consistency or produce invalid function evaluations, which require careful handling.

For a comprehensive overview of nonlinear programming, interior-point methods, and their application to collocation-based dynamic optimization, we refer interested readers to (L. T. Biegler 2010) and the Ipopt implementation paper (Wächter and L. T. Biegler 2006).

2.5 Challenges and Practical Aspects

We identify four main challenges in training PeN-ODEs using the proposed direct collocation and nonlinear programming approach. These challenges are closely related to those encountered in conventional PeN-ODE or general NN training.

2.5.1 Grid Selection

The choice of time grid $\{t_0, \ldots, t_{n+1}\}$ and the number of collocation nodes per interval m_i are crucial for both the accuracy and efficiency of the training process. In practice, the grid can either be chosen equidistant or tailored to the specific problem. While equidistant grids are straightforward to implement and often sufficient for well-behaved systems, non-equidistant grids may reduce computational costs while capturing the dynamics more efficiently. Placing more intervals with low degree collocation polynomials in regions of rapid state change can improve approximation quality without unnecessarily increasing the problem size. Similarly, in well-behaved regions, it is feasible to perform larger steps with more collocation nodes. Because the collocation scheme and grid are embedded into the NLP, these must be given *a-priori*. This leaves room for future developments of adaptive mesh refinement methods with effective mesh size reduction, which have already shown great success for optimal control problems (Zhao and Shang 2018; Liu, Hager, and A. Rao 2015).

2.5.2 Initial Guesses

Due to the size and possible nonlinearity of the resulting NLP, the choice of initial guesses has a strong influence on convergence behavior. Unlike classical NN training, where poor initialization primarily affects convergence speed, the constrained nature of the transcribed dynamic optimization problem can lead to poor local optima or even solver failure. It is therefore of high importance to perform informed initializations for the states $\boldsymbol{x}_{ij}$ and, if possible, for the NN parameters $\boldsymbol{p}$.

One practical approach to obtain the required parameter guesses is to first train the network on a small, representative subset of the full dataset using constant initial values for both the states and parameters. The optimized parameters resulting from this reduced problem then serve as informed initial guesses for the full training problem. Consequently, the states are obtained by simulation, i.e. $\boldsymbol{x}(t) := \text{ODESolve}\left(\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{u}, t, NN_{\boldsymbol{p}}(\boldsymbol{x}, \boldsymbol{u}, t), \boldsymbol{x}_0)\right)$ and $\boldsymbol{x}_{ij} := \boldsymbol{x}(t_{ij})$ for a given initial condition $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$. By construction, the collocated dynamic constraints (12b) are satisfied, leading to improved convergence and stability in the full NLP.

Clearly, this strategy does not work in general. However, in simple cases where the model can be decomposed and the NN's input-output behavior is observable, e.g. if model components should be replaced by a neural surrogate, the NN can be pre-trained using standard gradient descent. This yields reasonable initial guesses

DOI 10.3384/ecp218445

for the parameters and states by simulation, which can then be integrated into the constrained optimization problem. Another pre-training strategy could be the *collocation technique* proposed in (Roesch, Rackauckas, and Stumpf 2021). Still, developing general, effective strategies to obtain reasonable initial guesses is one of the key challenges and limiting factors we identify for the general application of this approach.

2.5.3 Batch-wise Training

Standard ML frameworks employ batch learning to efficiently split up data. This is not as straightforward when training with the approach described here. One might assume that the entire dataset must be included in a single discretized DOP. However, recent work (Shapovalova and Tsay 2025) demonstrates that batch-wise training is possible and promises significant potential. The Alternating Direction Method of Multipliers (ADMM) (Boyd et al. 2011) allows decomposing the optimization problem into smaller subproblems that can be trained independently, while enforcing consensus between them. This allows for memory-efficient training and opens up the possibility of handling larger models or learning from multiple data trajectories simultaneously.

2.5.4 Training of Larger Networks

While computing Hessians of NNs is generally expensive, it is tractable for small networks. To reduce computational effort for larger networks, it might be reasonable to use partial Quasi-Newton approximations such as SR1, BFGS or DFP (L. T. Biegler 2010) to approximate the dense parts of the augmented Lagrangian Hessian H, e.g. the blocks $H_{\boldsymbol{x}_{ij},\boldsymbol{p}}$ and $H_{\boldsymbol{pp}}$, or solely $H_{\boldsymbol{pp}}$. These blocks, which contain derivatives with respect to the NN parameters, are computationally expensive, while the block $H_{\boldsymbol{x}_{ij},\boldsymbol{x}_{ij}}$, which contains second derivatives with respect to the collocated states, is extremely sparse and comparably cheap. A Quasi-Newton approximation of the block $H_{\boldsymbol{x}_{ij},\boldsymbol{x}_{ij}}$ is therefore disadvantageous. The sparsity can be exploited by computing this block analytically and using it directly in the Quasi-Newton update. An implementation of this partial update using the SR1 Quasi-Newton method is straightforward. Instead of one expensive symmetric rank-one update for the entire Hessian H, one cheap symmetric rank-one update for $H_{\boldsymbol{pp}}$ and one general rank-one update for $H_{\boldsymbol{x}_{ij},\boldsymbol{p}}$ are needed.

This procedure significantly reduces the cost of the Hessian, while still providing fairly detailed derivative information. SR1 is particularly advantageous here, as it can represent indefinite Hessians, which is favorable when dealing with highly nonlinear functions.

Since our current examples perform well with small NNs, we do not explore larger networks in this paper. However, we anticipate that such Quasi-Newton strategies will be necessary in future work with larger networks. Very recent work (Lueg et al. 2025) independently expresses similar ideas, highlighting the potential of the approach.

3 Implementation

The generic DOP (3a)–(3d) and its corresponding NLP formulation (12a)–(12d) are implemented in the custom open-source framework GDOPT (Langenkamp 2024), which is publicly available.[1] For neural network training the code has been extended, including predefined parametric blocks such as neural networks, support for data trajectories and parallelized optimizations. This extended, experimental version is also publicly available.[2]

3.1 GDOPT

GDOPT (General Dynamic Optimizer) consists of two main components: an expressive Python-based package *gdopt* and an efficient C++ library *libgdopt*. The Python interface provides an user-friendly modeling environment and performs symbolic differentiation and code generation. Symbolic expressions are optimized using common subexpression elimination via SymEngine[3], and the resulting expressions together with first and second derivatives are translated into efficient C++ callback functions for runtime evaluation. In the present implementation, all expressions are flattened, resulting in large code, especially for the Hessian. Note that keeping NNs vectorized and employing symbolic differentiation rules with predefined NN functions offers significant advantages.

The library *libgdopt* implements a generalized version of the NLP (12a)–(12d) using Radau IIA collocation schemes, while also supporting optimal control problems. It is interfaced with Ipopt to solve the resulting nonlinear programs. Both symbolic Jacobian and Hessian rely on exact sparsity patterns discovered in the Python interface. Additional functionality includes support for nominal values, initial guesses, runtime parameters, mesh refinements, plotting utilities, and special functions. A detailed overview of features and modeling is provided in the GDOPT User's Guide[4].

Nevertheless, GDOPT lacks important capabilities that established modeling languages and tools offer, such as object-oriented, component-based modeling and support for differential-algebraic equation (DAE) systems. It is possible to model DAEs by introducing control variables for algebraic variables. However, this approach increases the workload and size of the NLP and may lead to instabilities.

3.2 Parallel Callback Evaluations

Since in every optimization step, the function evaluations as well as first and second derivatives of all NLP components (12a)–(12d) must be provided to Ipopt, an efficient callback evaluation is crucial to accelerate the training.

[1] https://github.com/linuslangenkamp/GDOPT
[2] https://github.com/linuslangenkamp/GDOPT_DEV
[3] https://github.com/symengine/symengine
[4] https://github.com/linuslangenkamp/GDOPT/blob/master/usersguide/usersguide.pdf

Note that these callbacks themselves consist of the continuous functions evaluated at all collocation nodes. For simplicity we write $\boldsymbol{z}_{ij} := [\boldsymbol{x}_{ij}, \boldsymbol{p}]^T$ for the variables at a given collocation node and, in addition,

$$\boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{p}, t) := [L(\cdot), \boldsymbol{f}(\cdot), \boldsymbol{g}(\cdot)]^T \tag{13}$$

for the vector of functions that are evaluated at all nodes. Clearly, the required callbacks

$$\boldsymbol{\psi}\big|_{z_{ij}} \quad \nabla\boldsymbol{\psi}\big|_{z_{ij}} \quad \nabla^2\boldsymbol{\psi}\big|_{z_{ij}} \quad \forall i\, \forall j \geq 1$$

are independent with respect to the given collocation nodes t_{ij} and thus, allow for a straightforward parallelization. In the extension of GDOPT, we use OpenMP (Chandra et al. 2001) to parallelize the callback evaluations required by Ipopt. Depending on the specific callback, i.e. objective function, constraint violation, gradient, Jacobian, or Hessian of the augmented Lagrangian, a separate `omp parallel for` loop is used to evaluate the corresponding components at all collocation nodes. This design results in a significant reduction in computation time, especially for the comparably expensive dense derivatives of neural components.

4 Performance

In order to test the proposed training method and parallel implementation, two example problems are considered. The first example is the Quarter Vehicle Model (QVM) from (Kamp, Ultsch, and Brembeck 2023), where an equation-based model is enhanced with small neural components, such that physical behavior is represented more accurately. The second example is a standard NODE, where the dynamics of a Van-der-Pol oscillator (Roesch, Rackauckas, and Stumpf 2021) are learned purely from data. In both cases, the experimental setup closely follows the configurations used in the respective paper. Training is performed on a laptop running Ubuntu 24.04.2 with Intel Core i7-12800H (20 threads), 32 GB RAM and using GCC v13.3.0 with flags `-O3 -ffast-math` for compilation, while MUMPS (Amestoy et al. 2001) is used to solve linear systems arising from Ipopt (Wächter and L. T. Biegler 2006). All dependencies are free to use and open-source.

4.1 Quarter Vehicle Model

We follow the presentation in (Kamp, Ultsch, and Brembeck 2023) for an overview of the model and the data generation process. The Quarter Vehicle Model (QVM) captures the vertical dynamics of a road vehicle by modeling one wheel and the corresponding quarter of the vehicle body. It consists of two masses connected by spring-damper elements representing the suspension and tire dynamics. The linear base model is described by the differential equations $\dot{z}_r = u$, $\dot{z}_b = v_b$, $\dot{z}_w = v_w$, and

$$\dot{v}_b := a_b = m_b^{-1}\left(c_s \Delta z_s + d_s \Delta v_s\right) \tag{14a}$$

$$\dot{v}_w := a_w = m_w^{-1}\left(c_t \Delta z_t + d_t \Delta v_t - c_s \Delta z_s - d_s \Delta v_s\right) \tag{14b}$$

where $\Delta z_s = z_w - z_b$, $\Delta v_s = v_w - v_b$, $\Delta z_t = z_r - z_w$, $\Delta v_t = u - v_w$. In addition, m_w is the mass of the wheel, m_b is mass of the quarter body, and c_s and d_s are the coefficients of the linear spring-damper pair between these masses, modeling the suspension. Furthermore, c_t and d_t define an additional linear spring-damper pair between the tire and ground. The state vector $\boldsymbol{x} = [z_b, z_w, v_b, v_w, z_r]^T$ contains the positions of the body z_b and wheel z_w, their velocities v_b and v_w, and the road height z_r. The differential road height $\dot{z}_r = u$ is given as an input and the observable outputs $y = [a_w, a_b]$ measure the wheel and body accelerations. A corresponding Modelica model of the linear QVM is depicted in Figure 1. (Kamp, Ultsch, and Brembeck 2023)

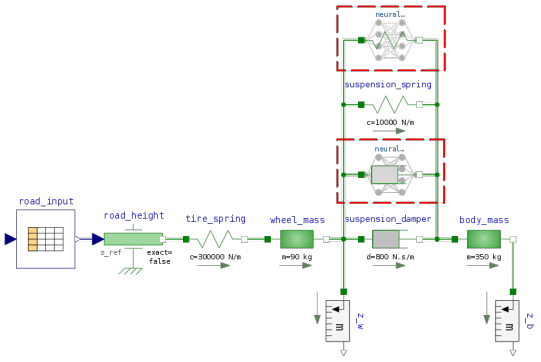


Figure 1. Modelica Models of the Linear (without boxes) and Neural (with boxes) QVM. Modified from T. Kamp.

4.1.1 Data Generation

The linear model is extended by introducing two additional nonlinear forces between both masses, a translational friction force F_{fr} and a progressive spring characteristic F_{pr}. These nonlinear components are introduced only for data generation, creating a more complicated, nonlinear model whose behavior deviates from the known base dynamics. Therefore, the differential equations (14a) and (14b) become

$$\dot{v}_b = m_b^{-1}\left(c_s \Delta z_s + d_s \Delta v_s + F_{pr}(\Delta z_s) + F_{fr}(\Delta v_s)\right) \tag{15a}$$

$$\dot{v}_w = m_w^{-1}\left(c_t \Delta z_t + d_t \Delta v_t - c_s \Delta z_s - d_s \Delta v_s - F_{pr}(\Delta z_s) - F_{fr}(\Delta v_s)\right) \tag{15b}$$

To generate data, a simulation of the nonlinear model for an imitation of a realistic, rough road (ISO8608, Type D (Múčka, Peter 2018)) is performed. The observables a_b and a_w as well as the states are disturbed by random Gaussian noise and sampled with 1000 Hz for a 42 s trajectory as in (Kamp, Ultsch, and Brembeck 2023).

4.1.2 Training Setup

The nonlinear components $F_{fr}(\Delta v_s)$ and $F_{pr}(\Delta z_s)$ introduced for data generation are replaced by two neural network surrogates $F_{fr}^{NN}(\Delta v_s)$ and $F_{pr}^{NN}(\Delta z_s)$, illustrated in Figure 1. The goal is to find suitable replacements that minimize the mismatch between the simulated, disturbed

DOI 10.3384/ecp218445

outputs $\hat{a}_b$ and $\hat{a}_w$ of the nonlinear model and the observed data during the optimization. As discussed before, we write this objective as an integral over the entire time horizon, i.e.

$$\min \int_{t_0}^{t_f} \left(\frac{\hat{a}_b - a_b}{\sigma_{\hat{a}_b}} \right)^2 + \left(\frac{\hat{a}_w - a_w}{\sigma_{\hat{a}_w}} \right)^2 \mathrm{d}t, \tag{16}$$

where $\sigma_{\hat{a}_b}$ and $\sigma_{\hat{a}_w}$ are corresponding standard deviations of the data, ensuring that both accelerations contribute equally to the objective. Furthermore, it is known that both force elements have a zero crossing and therefore, the additional constraints

$$F_{fr}^{NN}(0) = 0 \quad \text{and} \quad F_{pr}^{NN}(0) = 0 \tag{17}$$

are simply added to the optimization problem.

We employ three different training strategies. At first, both feedforward neural networks are trained directly on the full trajectory using randomly initialized parameters, while the initial state guesses are obtained from a simulation of the linear model (I). Each network has the structure $1 \times 5 \rightarrow 5 \times 5 \rightarrow 5 \times 1$ and therefore, both nets contain just 92 parameters in total. We use the smooth squareplus activation function

$$\text{squareplus}(x) := \frac{x + \sqrt{1 + x^2}}{2} \tag{18}$$

to ensure a twice continuously differentiable NN as required for Ipopt. In the second strategy (II), described in Section 2.5.2, first an acceleration scheme is employed, where the same networks are trained on a short segment one eighth of the entire trajectory. After that, a simulation of the neural QVM with the obtained parameters is performed. The resulting states are used as initial guesses in the subsequent optimization with full data. To show that the surrogates need not be NNs and training can be performed efficiently with other parameter-dependent expressions, the third strategy (III) uses rational functions to model the unknown behavior. For instance, F_{fr} is replaced by

$$F_{fr}^{RC}(\Delta v_s) := \frac{\sum_{k=0}^{N} \omega_k T_k(\Delta v_s)}{\sum_{k=0}^{D} \theta_k T_k(\Delta v_s)}, \tag{19}$$

where T_k is the k-th Chebyshev polynomial, ω_k and θ_k are parameters to be optimized, and N and D are the numerator and denominator degrees. For both rational functions we choose $N = D = 7$, resulting in a total of merely 32 learnable parameters.

Since the QVM contains very fast dynamics due to high-frequency excitations, in all cases the time horizon is divided into a tightly spaced, equidistant grid of 2500 intervals and using a constant 5-step Radau IIA collocation scheme of order 9. This leads to a total of 12500 collocation nodes and more than 2.7×10^6 nonzeros in the Jacobian and roughly 4.73×10^6 nonzeros in the Hessian of the large-scale NLP.

4.1.3 Results

Table 1 presents the training times for all strategies. Each optimization was run for a maximum of 150 NLP iterations and was automatically terminated early if no further significant improvement in objective could be achieved. We want to stress that all trainings, performed on a laptop, are executed in under 7 minutes, compared to 4.5 hours for the fastest optimization in (Kamp, Ultsch, and Brembeck 2023) using ODE solver-based training. Clearly, this is also due to the fact that smaller neural components are used. Furthermore, Figure 2 depicts the objective value with respect to training time, where both the first and second optimizations of (II) are concatenated.

Strategy	Total	Ipopt	Callbacks
(I)	394.43	284.18	110.25
(II) - initial	26.75	15.08	11.67
(II) - final	173.06	124.65	48.41
(II)	199.81	139.73	60.07
(III)	34.96	27.10	7.85

Table 1. Training Times in Seconds

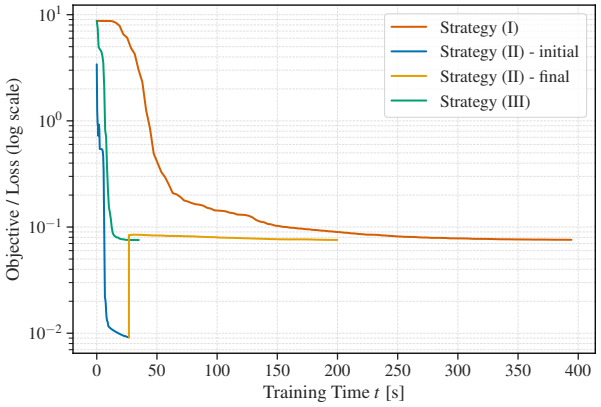


Figure 2. Objective History with Respect to Training Time

Even though a poor initialization has been used and thus the optimization required more time, the naive strategy (I) converged stably to a suitable optimum with a similar objective as strategies (II) and (III). For this example, the acceleration scheme (II) proves effective, since the initial optimization for a shorter data trajectory takes just 26.75 s in total, and the subsequent initial guesses for states and parameters become very good approximations of the real solution. This can be observed in Figure 3 and Figure 4, where the resulting neural surrogates are depicted. By performing the second optimization, (II) effectively halves the time required by strategy (I), i.e. less than 3.5 minutes, and furthermore results in indistinguishable neural components.

Moreover, as seen in Figure 3 and Figure 4, the resulting very small NNs match the reference in almost perfect

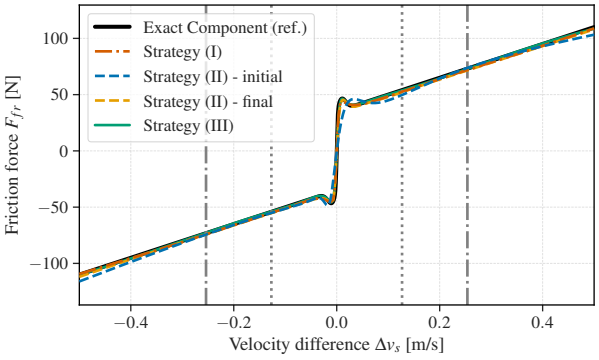


Figure 3. Neural and Reference Damper Characteristics

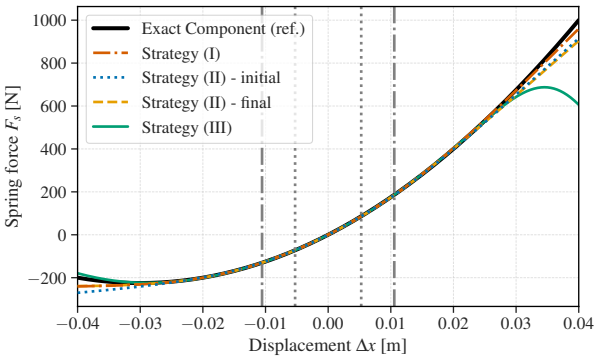


Figure 4. Neural and Reference Spring Characteristics

accordance, correctly representing highly nonlinear parts of the damper. These also generalize in a very natural way, as can be seen from the behavior outside the vertical dotted lines, which represent the first and second standard deviations of the inputs to the nets. Performing simulations on a new, unknown input road shows that the obtained PeNODE from (II) matches the nonlinear reference model perfectly, as illustrated in Figure 5. Note that, our characteristics of the damper and spring serve as even better surrogates than those reported in (Kamp, Ultsch, and Brembeck 2023), despite using significantly smaller networks and requiring considerably shorter training times. While (Kamp, Ultsch, and Brembeck 2023) relied on larger models with longer ODE solver-based training, our approach yields more accurate and better-generalizing results.

By having principal knowledge of the underlying characteristics shown in Figure 3 and Figure 4, it is possible to model observed behavior with minimal parameters. Since the number of Hessian nonzeros grows quadratically with the number of parameters, such knowledge of the procedure can greatly benefit both training time and surrogate quality. Therefore, consider simple rational functions as an educated guess for both missing components. This optimization, with 32 instead of 92 free parameters, is performed in under 35 seconds without any acceleration strategy. Moreover, the obtained surrogates are of mostly equal quality to the NN components from (II), although

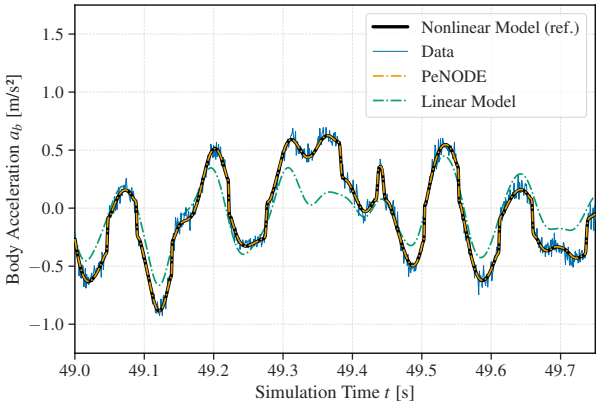


Figure 5. Body Accelerations a_b for Simulations of PeNODE and Standard Models on a Type C Road (Múčka, Peter 2018)

Figure 4 shows that the rational function does not generalize as well. Nevertheless, these results demonstrate that unknown behavior may be expressed with fewer parameters and yield equivalent quality.

4.1.4 Parallel Implementation

Finally, it is stressed that the parallel implementation, described in Section 3.2, leads to 5.44 times less time taken in the generated function callbacks. This yields a total training time that is more than halved and clearly shows that the implementation efficiently exploits the independence of collocation nodes.

Method	Total	Ipopt	Callbacks
GDOPT (default)	385.90	122.52	263.38
GDOPT (parallel)	173.06	124.65	48.41

Table 2. Comparison of Parallel and Sequential Optimization Times in Seconds of (II) - final

4.2 Van-der-Pol Oscillator

To illustrate the ability of learning a full NODE, we follow an example from (Roesch, Rackauckas, and Stumpf 2021), where a different kind of collocation method was proposed. This method approximates the RHS of the ODE with data, thus enabling faster, unconstrained training without the need for ODE solvers. Consider the Van-der-Pol (VdP) oscillator

$$\dot{x} = y \tag{20a}$$

$$\dot{y} = \mu y \left(1 - x^2\right) - x \tag{20b}$$

with $\mu = 1$ and initial conditions $x(t_0) = 2$ and $y(t_0) = 0$. Data generation is performed by simulating the dynamics with OpenModelica (Fritzson, Pop, Abdelhak, et al. 2020) on an equidistant grid with 200 intervals and artificially perturbing the observed states by additive Gaussian noise. To test sensitivities of the approach, we use 3 different

DOI
10.3384/ecp218445

levels of noise $\mathcal{N}(0,\sigma)$ to disturb the observable states, i.e. no noise ($\sigma = 0$), low noise ($\sigma = 0.1$) and high noise ($\sigma = 0.5$).

The continuous DOP has the form

$$\min_{\boldsymbol{p}_x,\boldsymbol{p}_y} \int_{t_0}^{t_f} (\hat{x}_\sigma - x)^2 + (\hat{y}_\sigma - y)^2 \,\mathrm{d}t + \lambda \|\boldsymbol{p}\|_2^2 \tag{21a}$$

s.t.

$$\dot{x} = NN^x_{\boldsymbol{p}_x}(x,y) \tag{21b}$$

$$\dot{y} = NN^y_{\boldsymbol{p}_y}(x,y) \tag{21c}$$

$$x(t_0) = 2,\ y(t_0) = 0, \tag{21d}$$

where $\hat{x}_\sigma, \hat{y}_\sigma$ is the disturbed state data, $NN^x_{\boldsymbol{p}_x}(x,y)$, $NN^y_{\boldsymbol{p}_y}(x,y)$ are neural networks of the architecture $2 \times 5 \to 5 \times 5 \to 5 \times 1$ with sigmoid activation function, $\boldsymbol{p} = [\boldsymbol{p}_x, \boldsymbol{p}_y]^T$ are 102 learnable parameters, while $\lambda > 0$ is a regularization factor to enhance stability.

As in (Roesch, Rackauckas, and Stumpf 2021), the training is performed on a 7 second time horizon, thus including a little over one period of the oscillator. Furthermore, 500 intervals and the 5-step Radau IIA method of order 9 are used. The initial guesses for the state variables are trivially chosen as the constant initial condition and the NN parameters are initialized randomly. No acceleration strategy or simulation is used for educated initial guesses. We set a maximum number of Ipopt iterations / epochs of 200 and an optimality tolerance of 10^{-7}.

4.2.1 Results

In almost all cases, the optimization terminates prematurely, since the optimality tolerance is fulfilled and thus, a local optimum was found. The corresponding training times are displayed in Table 3. Note that, because the high noise ($\sigma = 0.5$) leads to larger objective values, the regularization λ is increased in this case. Nevertheless, the optimization with a total of 2500 discrete nodes is very rapid and terminates in several seconds from an extremely poor initial guess, while in some instances runs settle in poor local optima. Clearly, more reasonable initial guesses and sophisticated initialization strategies will further enhance these times and stability.

σ	λ	Total	Ipopt	Callbacks	#Epochs
0	10^{-4}	8.17	5.95	2.22	80
0.1	10^{-4}	7.69	5.60	2.09	76
0.5	10^{-3}	13.49	9.79	3.70	134

Table 3. Training Times (in seconds) and Number of Ipopt Iterations / Epochs of the Van-der-Pol Oscillator NNs

Figure 6 presents the simulation results and training data for various levels of noise. It is evident that for no or low noise, the solution obtained is virtually indistinguishable from the reference, which is quite impressive. Even with high noise ($\sigma = 0.5$), the Neural ODE remains close to the true solution, showing significantly better performance compared to the results reported in (Roesch, Rackauckas, and Stumpf 2021), where NODE and reference do not align for $\sigma \geq 0.2$. While smaller neural networks are employed here, these compact models still demonstrate exemplary performance and fast training, highlighting the effectiveness of the approach under severe noise.

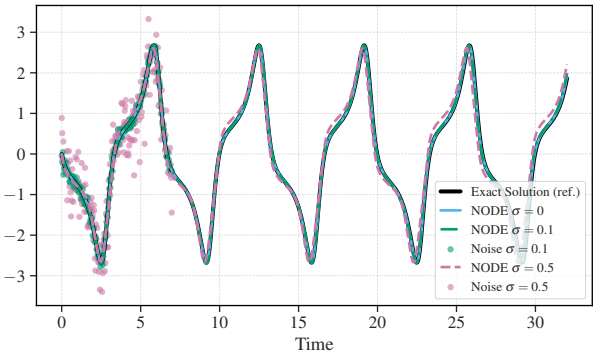


Figure 6. Simulation Results of the Neural and Reference Models as well as the Data for $y(t)$

To verify the robustness of the training procedure, a comprehensive sensitivity analysis, consisting of 100 training runs for each noise level, was conducted. The results, detailed in the Appendix and Figure 10, demonstrate that while all runs converge perfectly for the no-noise case ($\sigma = 0$) and approximately 95% show excellent agreement under low noise ($\sigma = 0.1$), as expected the robustness diminishes with high noise ($\sigma = 0.5$). In these cases, several runs converge to poor local optima or fail to converge, leading to solutions with noticeable period and amplitude mismatches or an outright collapse of the trajectory.

To further illustrate the obtained NODEs, we compare the learned and true vector fields of the ODE

$$\begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} NN^x_{\boldsymbol{p}_x}(x,y) \\ NN^y_{\boldsymbol{p}_y}(x,y) \end{bmatrix}, \quad \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} y \\ \mu y\left(1-x^2\right) - x \end{bmatrix}. \tag{22}$$

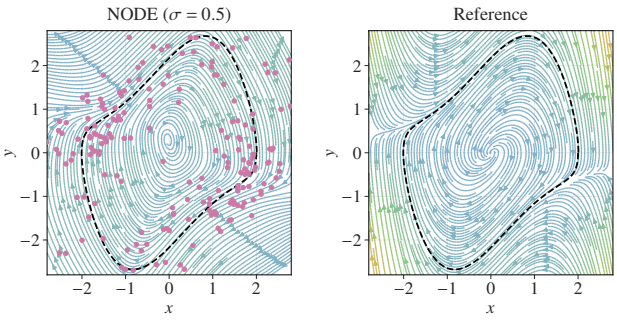


Figure 7. Neural ($\sigma = 0.5$) and Reference Vector Fields of the ODE and the Exact VdP Trajectory (dashed)

In Figure 7, we show the high-noise NODE, its training data, and the true vector field. Despite the heavily

scattered observable states, the NODE still manages to recover a vector field that aligns well with the true dynamics in the vicinity of the solution trajectory.

Further comparison is given in Figure 8, where the scalar fields visualize the 2-norm error between the neural and reference vector fields. The low-noise NODE ($\sigma = 0.1$) yields way smaller error values along the trajectory, but even the high-noise model produces a fairly accurate vector field in regions close to available training data. As expected, generalization outside this domain remains limited, but within the training region, the results demonstrate strong consistency.

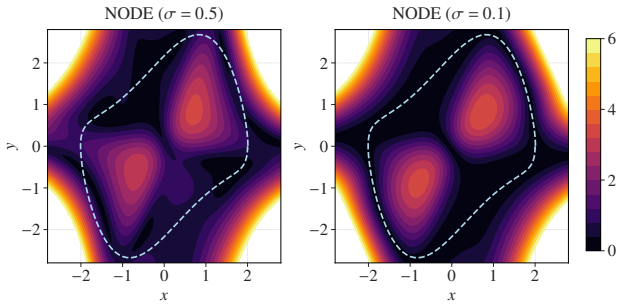


Figure 8. Scalar Fields Representing the 2-Norm Error Between the Neural ($\sigma = 0.5$, $\sigma = 0.1$) and Reference Vector Fields (Values > 6 are white)

In addition to the local collocation approach with 500 intervals, we also reproduce the results in (Shapovalova and Tsay 2025) using a global, *spectral* collocation method. We employ a single interval with 70 fLGR nodes, corresponding to Radau IIA of order 139. This high-order global formulation yields an approximation of equal quality to the figures above, even under high noise ($\sigma = 0.5$). Furthermore, the optimization terminates in outstandingly fast time, i.e. under 1.2 seconds. This demonstrates the remarkable efficiency and accuracy of spectral methods for such smooth problems.

5 Future Work

Although OpenModelica currently includes an optimization runtime implementing direct collocation (Ruge et al. 2014), it remains limited to basic features: it supports only 1- or 3-step Radau IIA collocation, does not allow parameter optimization, lacks analytic Hessians, and does not perform parallel callbacks. To overcome these limitations, work is underway to embed an extended version of *libgdopt* into OpenModelica. This integration combines recent developments from GDOPT with the existing strengths of OpenModelica, particularly its native ability to handle DAEs. The extended framework enables expressive Modelica-based modeling, native support of neural components via the NeuralNetwork Modelica library and incorporates recent advancements in mesh refinement for optimal control problems (Langenkamp 2024).

5.1 NeuralNetwork Modelica Library

The NeuralNetwork library, originally developed in (Codecà and Casella 2006), is an open-source Modelica library[5] modeling ML architectures with pure Modelica. Neural components can be constructed by connecting dense feedforward layers of arbitrary size with layers for PCA, standardizing, or scaling. These blocks contain all equations and parameters as pure Modelica code, which makes seamless integration of neural components into existing Modelica models straightforward. Together with this library, OpenModelica will enable modeling and training of PeN-ODEs within a single development environment.

5.2 Workflow

The workflow presented in Figure 9 illustrates the intended process for native Modelica-based modeling and training of PeN-ODEs. While some components of the workflow are operational, the full integration is still under active development. Users model physical systems and NN components with free parameters directly in Modelica and provide corresponding data. Both the model

[5] https://github.com/AMIT-HSBI/NeuralNetwork

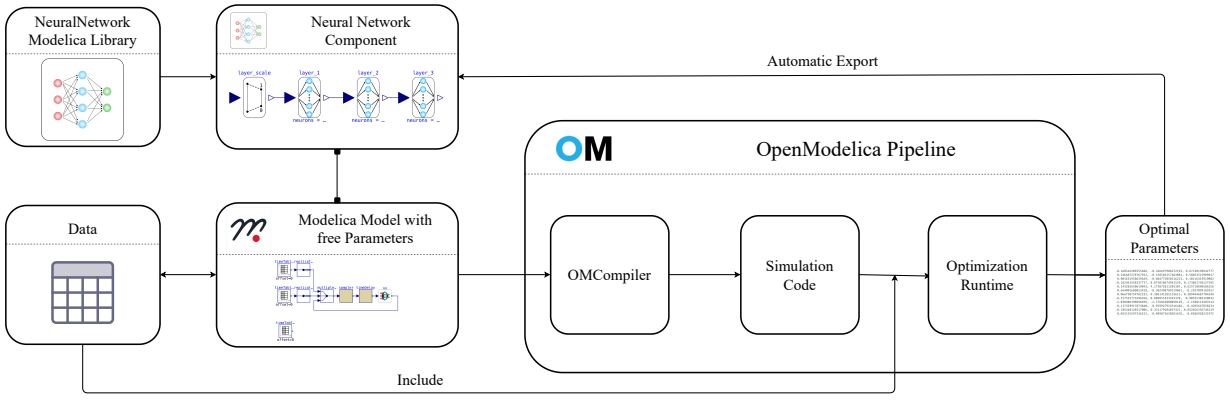


Figure 9. OpenModelica Workflow for PeN-ODE Training (under development)

DOI 10.3384/ecp218445

and data are processed by the standard OpenModelica pipeline, while the OpenModelica Compiler (OMC) generates fast C code. The new backend of OMC introduces efficient array-size-independent symbolic manipulation (Abdelhak, Casella, and Bachmann 2023). Ongoing work further targets resizability of components after compilation (Abdelhak and Bachmann 2025), offering significant benefits for array-based components such as NNs.

Furthermore, current work focuses on leveraging the generated simulation code to enable fast callback functions for a new optimization runtime. After training, the optimal parameters are directly inserted into a NN block, enabling immediate simulations by swapping connectors.

This workflow removes the need for export and import steps for neural components and models, e.g. in the form of a Functional Mock-up Unit (FMU). It also eliminates reliance on external training routines in Julia or Python and avoids external C functions in the model, since the neural components are pure Modelica blocks. This integrated workflow unifies modeling, training, and simulation within a single toolchain, enabling free and accessible optimizations.

5.3 Neural DAEs

One of the main advantages of the integration into OpenModelica is the native ability to handle DAEs. Modelica compilers such as OpenModelica systematically apply index reduction and block-lower triangular (BLT) transformations to restructure DAEs into semi-explicit ODE form with index 1 (Ruge et al. 2014; Åkesson et al. 2012). As the simulation code already resolves algebraic variables during evaluations, no additional handling, e.g. inclusion of algebraic variables in the NLP, is needed on the optimization side. This allows the new training workflow to seamlessly extend from ODEs to DAEs, far surpassing the current range of applications.

6 Conclusion

This paper proposes a formulation of PeN-ODE training as a collocation-based NLP, simultaneously optimizing states and NN parameters. The approach overcomes key limitations of ODE solver-based training in terms of order, stability, accuracy, and allowable step size. The NLP uses high order quadrature for the NN loss, potentially preserving the accuracy of the discretization. We demonstrate that known physical behavior can be trivially enforced.

We provide an open-source parallelized extension to GDOPT and on two example problems demonstrate exemplary accuracy, training times, and generalization with smaller NNs compared to other training techniques, even under significant noise. Furthermore, we show that the approach allows for efficient optimization of other parameter dependent surrogates.

Key limitations and challenges of the proposed method, including grid selection and general initialization strategies to increase stability, as well as training with larger datasets and networks are identified. Addressing and evaluating these issues in future work is essential to support broader applicability. To enable accessible training of Neural DAEs, without relying on external tools, work is underway to implement the method in OpenModelica.

Acknowledgements

This work was conducted as part of the OpenSCALING project (Grant No. 01IS23062E) at the University of Applied Sciences and Arts Bielefeld, in collaboration with Linköping University. The authors would like to express their sincere appreciation to both the OpenSCALING project and the Open Source Modelica Consortium (OSMC) for their support, collaboration, and shared commitment to advancing open-source modeling and simulation technologies.

References

Abdelhak, Karim and Bernhard Bachmann (2025). *Compiler Status and Development of the New Backend*. Presentation at the 17th OpenModelica Annual Workshop - February 3, 2025, accessed on 2025-04-25. URL: https://openmodelica.org/events/openmodelica-workshop/2025.

Abdelhak, Karim, Francesco Casella, and Bernhard Bachmann (2023-12). "Pseudo Array Causalization". In: pp. 177–188. DOI: 10.3384/ecp204177.

Åkesson, Johan et al. (2012). "Generation of Sparse Jacobians for the Function Mock-Up Interface 2.0". In: *Book of abstracts / 9th International Modelica Conferenc*. Vol. 76. Linköping Electronic Conference Proceedings. Linköping University Electronic Press, pp. 185–196. DOI: 10.3384/ecp12076185.

Amestoy, Patrick R. et al. (2001). "A Fully Asynchronous Multifrontal Solver Using Distributed Dynamic Scheduling". In: *SIAM Journal on Matrix Analysis and Applications* 23.1, pp. 15–41.

Andersson, Joel A E et al. (2019). "CasADi – A software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation* 11.1, pp. 1–36. DOI: 10.1007/s12532-018-0139-4.

Becerra, V. M. (2010). "Solving complex optimal control problems at no cost with PSOPT". In: *2010 IEEE International Symposium on Computer-Aided Control System Design*, pp. 1391–1396. DOI: 10.1109/CACSD.2010.5612676.

Biegler, Lorenz T. (2010-01). *Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes*. ISBN: 9780898717020. DOI: 10.1137/1.9780898719383.

Boyd, Stephen et al. (2011). "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers". In: *Foundations and Trends® in Machine Learning* 3.1, pp. 1–122. ISSN: 1935-8237. DOI: 10.1561/2200000016. URL: http://dx.doi.org/10.1561/2200000016.

Bruder, Frederic and Lars Mikelsons (2021-09). "Modia and Julia for Grey Box Modeling". In: pp. 87–95. DOI: 10.3384/ecp2118187.

Byrd, Richard H., Jorge Nocedal, and Richard A. Waltz (2006). "Knitro: An Integrated Package for Nonlinear Optimization". In: *Large-Scale Nonlinear Optimization*. Ed. by G. Di Pillo and M. Roma. Boston, MA: Springer US, pp. 35–59. ISBN: 978-0-387-30065-8. DOI: 10.1007/0-387-30065-1_4. URL: https://doi.org/10.1007/0-387-30065-1_4.

Chandra, Rohit et al. (2001). *Parallel programming in OpenMP*. Morgan Kaufmann.

Chen, Tian Qi et al. (2018). "Neural Ordinary Differential Equations". In: *CoRR* abs/1806.07366. arXiv: 1806.07366. URL: http://arxiv.org/abs/1806.07366.

Codecà, Fabio and Francesco Casella (2006-09). "Neural Network Library in Modelica". In: *Proceedings of the 5th International Modelica Conference*. Vol. 2. Modelica Association. Vienna, Austria, pp. 549–557.

Fritzson, Peter, Adrian Pop, Karim Abdelhak, et al. (2020-10). "The OpenModelica Integrated Environment for Modeling, Simulation, and Model-Based Development". In: *Modeling, Identification and Control: A Norwegian Research Bulletin* 41, pp. 241–295. DOI: 10.4173/mic.2020.4.1.

Gill, P. E. et al. (2007). *SNOPT 7.7 User's Manual*. Tech. rep. CCoM Technical Report 18-1. San Diego, CA: Center for Computational Mathematics, University of California, San Diego.

Gill, Philip E., Walter Murray, and Michael A. Saunders (2005-01). "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization". In: *SIAM Rev.* 47.1, pp. 99–131. ISSN: 0036-1445. DOI: 10.1137/S0036144504446096. URL: https://doi.org/10.1137/S0036144504446096.

HSL (2013). *HSL: A collection of Fortran codes for large-scale scientific computation*. Available at http://www.hsl.rl.ac.uk. Accessed: 15-04-2025.

Kamp, Tobias, Johannes Ultsch, and Jonathan Brembeck (2023-11). "Closing the Sim-to-Real Gap with Physics-Enhanced Neural ODEs". In: *20th International Conference on Informatics in Control, Automation and Robotics, ICINCO 2023*. Ed. by Guiseppina Gini, Henk Nijmeijer, and Dimitar Filev. Vol. 2. Proceedings of the 20th International Conference on Informatics in Control, Automation and Robotics. SCITEPRESS, pp. 77–84. URL: https://elib.dlr.de/200100/.

Langenkamp, Linus (2024-12). *Adaptively Refined Mesh for Collocation-Based Dynamic Optimization*. Master's thesis. DOI: 10.13140/RG.2.2.18499.72484.

Lehtimäki, Mikko, Lassi Paunonen, and Marja-Leena Linne (2024-01). "Accelerating Neural ODEs Using Model Order Reduction". In: *IEEE Transactions on Neural Networks and Learning Systems* 35.1, pp. 519–531. ISSN: 2162-2388. DOI: 10.1109/tnnls.2022.3175757. URL: http://dx.doi.org/10.1109/TNNLS.2022.3175757.

Liu, Fengjin, William Hager, and Anil Rao (2015-05). "Adaptive Mesh Refinement Method for Optimal Control Using Nonsmoothness Detection and Mesh Size Reduction". In: *Journal of the Franklin Institute* 47. DOI: 10.1016/j.jfranklin.2015.05.028.

Lueg, Laurens R. et al. (2025). *A Simultaneous Approach for Training Neural Differential-Algebraic Systems of Equations*. arXiv: 2504.04665 [cs.LG]. URL: https://arxiv.org/abs/2504.04665.

Magnusson, Fredrik and Johan Åkesson (2015). "Dynamic Optimization in JModelica.org". In: *Processes* 3.2, pp. 471–496. ISSN: 2227-9717. DOI: 10.3390/pr3020471. URL: https://www.mdpi.com/2227-9717/3/2/471.

Misener, Ruth and Lorenz Biegler (2023). "Formulating data-driven surrogate models for process optimization". In: *Computers & Chemical Engineering* 179, p. 108411. ISSN: 0098-1354. DOI: https://doi.org/10.1016/j.compchemeng.2023.108411. URL: https://www.sciencedirect.com/science/article/pii/S0098135423002818.

Múčka, Peter (2018-01). "Simulated Road Profiles According to ISO 8608 in Vibration Analysis". In: *Journal of Testing and Evaluation* 46, p. 20160265. DOI: 10.1520/JTE20160265.

Patterson, Michael A. and Anil V. Rao (2014-10). "GPOPS-II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using hp-Adaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming". In: *ACM Trans. Math. Softw.* 41.1. ISSN: 0098-3500. DOI: 10.1145/2558904. URL: https://doi.org/10.1145/2558904.

Rackauckas, Chris et al. (2020-01). *Universal Differential Equations for Scientific Machine Learning*. DOI: 10.21203/rs.3.rs-55125/v1.

Ramadhan, Ali et al. (2023). *Capturing missing physics in climate model parameterizations using neural differential equations*. arXiv: 2010.12559 [physics.ao-ph]. URL: https://arxiv.org/abs/2010.12559.

Roesch, Elisabeth, Chris Rackauckas, and Michael Stumpf (2021-07). "Collocation based training of neural ordinary differential equations". In: *Statistical Applications in Genetics and Molecular Biology* 20. DOI: 10.1515/sagmb-2020-0025.

Ruge, Vitalij et al. (2014). "Efficient Implementation of Collocation Methods for Optimization using OpenModelica and ADOL-C". In: *Proceedings of the 10th International Modelica Conference, March 10-12, 2014, Lund, Sweden*. Vol. 96. Linköping Electronic Conference Proceedings. Linköping University Electronic Press, pp. 1017–1025. DOI: 10.3384/ecp140961017.

Schneider, C. and W. Werner (1986). "Some New Aspects of Rational Interpolation". In: *Math. Comp.* 47.175, pp. 285–299. DOI: 10.1090/S0025-5718-1986-0842136-8. URL: https://www.ams.org/journals/mcom/1986-47-175/S0025-5718-1986-0842136-8/.

Shapovalova, Mariia and Calvin Tsay (2025). *Training Neural ODEs Using Fully Discretized Simultaneous Optimization*. arXiv: 2502.15642 [cs.LG]. URL: https://arxiv.org/abs/2502.15642.

Sorourifar, Farshud et al. (2023-09). "Physics-Enhanced Neural Ordinary Differential Equations: Application to Industrial Chemical Reaction Systems". In: *Industrial & Engineering Chemistry Research* 62. DOI: 10.1021/acs.iecr.3c01471.

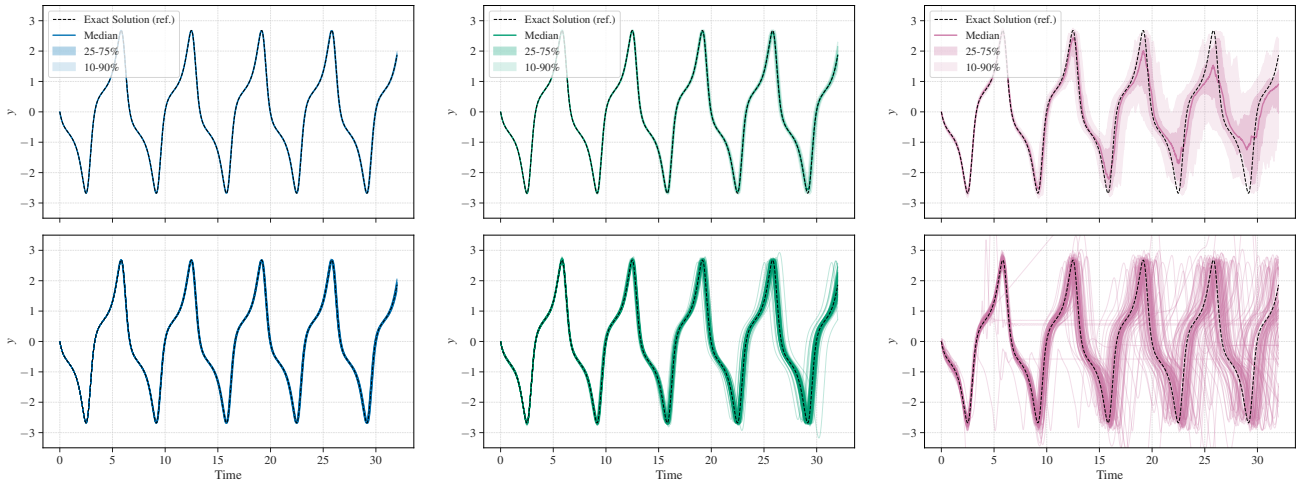
Thebelt, Alexander et al. (2022-02). "Maximizing information from chemical engineering data sets: Applications to machine learning". In: *Chemical Engineering Science* 252, p. 117469. DOI: 10.1016/j.ces.2022.117469.

Thummerer, Tobias, Johannes Stoljar, and Lars Mikelsons (2022). "NeuralFMU: Presenting a Workflow for Integrating Hybrid NeuralODEs into Real-World Applications". In: *Electronics* 11.19. ISSN: 2079-9292. DOI: 10.3390/electronics11193202. URL: https://www.mdpi.com/2079-9292/11/19/3202.

Wächter, Andreas and Lorenz T. Biegler (2006-03). "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". In: *Mathematical Programming* 106.1, pp. 25–57. ISSN: 1436-4646. DOI: 10.1007/s10107-004-0559-y. URL: https://doi.org/10.1007/s10107-004-0559-y.

Zhao, Jisong and Teng Shang (2018-09). "Dynamic Optimization Using Local Collocation Methods and Improved Multiresolution Technique". In: *Applied Sciences* 8, p. 1680. DOI: 10.3390/app8091680.

Appendix: Sensitivity Analysis and Robustness of VdP Neural ODEs



(a) No noise: $\sigma = 0$ **(b)** Low noise: $\sigma = 0.1$ **(c)** High noise: $\sigma = 0.5$

Figure 10. Sensitivity analysis of the learned Van-der-Pol (VdP) Neural ODEs to different levels of Gaussian noise. For each noise level, 100 training sessions were performed with random initial guesses for the neural network parameters and a random seed for the data perturbation. The plots show the simulation results for the observable state y(t) over an extended time horizon of 32 seconds. Each subplot consists of two panels: The top panel visualizes the reference solution (dashed black line), the median of the 100 learned solutions, and the 25-75% and 10-90% percentile bands. The bottom panel shows all 100 individual learned trajectories (faint lines). **(a) No noise** ($\sigma = 0$)**:** The method demonstrates full robustness and perfect convergence, with all 100 solutions converging to the exact reference solution. **(b) Low noise** ($\sigma = 0.1$)**:** The method remains highly robust. A vast majority of the solutions (approximately 95%) form a tight band around the reference, showing excellent agreement. **(c) High noise** ($\sigma = 0.5$)**:** As expected under severe noise, the robustness is reduced. While many solutions remain relatively close to the reference, some diverge significantly, indicating a failure to find a good local optimum during training. The solutions that do converge often show a period mismatch or diverge after a few periods, but still capture the general oscillatory behavior. The combined effects of reduced solution quality and an inaccurate period are reflected in the median and percentile bands, which exhibit a noticeable deviation in amplitude and phase after three periods compared to the true trajectory.

DOI
10.3384/ecp218445

Status of the SMArtInt Library: Simple Modelica Artificial Intelligence Interface

T. Hanke[1] J. Brunnemann[1] R. Flesch[1] J. Eiden[1]

[1]XRG Simulation GmbH, Hamburg, Germany, {hanke,brunnemann,flesch,eiden}@xrg-simulation.de

Abstract

This paper presents the current status of the open-source Modelica library SMArtInt (**S**imple **M**odelica **Art**ificial **I**ntelligence **Int**erface) (SMArtInt 2025), which offers users a straightforward and efficient approach of integrating artificial intelligence via neural networks directly into Modelica models. We provide a detailed overview of the library's features, area of application, and development process. The primary focus is on the diverse use cases of SMArtInt. The new version 0.5.1 brings an exciting feature with the support of the ONNX (ONNX 2025) format. ONNX (Open Neural Network Exchange) is an open, cross-platform format for the representation and exchange of deep learning models. This means that in addition to TensorFlow (TensorFlow 2025) models (TFLite), many other models, such as PyTorch (PyTorch 2025) can now also run in SMArtInt via the ONNX format.

Keywords: AI, SMArtInt library, Artificial Intelligence, Neural Network, TensorFlowLite, ONNX, BNODE

1 Introduction

1.1 Context of Paper

In recent years, machine learning techniques have successfully established themselves in various areas of science and technology, often summarized under the term "artificial intelligence" (AI). These methods have proven to be particularly effective in processing large amounts of data and enable an automatic recognition of correlations, patterns and classifications within the data (Lam et al. 2023; Kong et al. 2025).

Time series models, in which AI techniques such as neural networks can be used to predict and analyze temporal data, are more and more integrated into physical simulations. Such hybrid systems (see figure 1) combine data-driven insights with traditional physical models (Matei 2018; Kurz et al. 2022; Rudolph, Kurz, and Rakitsch 2024; Quarteroni, Gervasio, and Regazzoni 2025). This integration enables more accurate and flexible system modeling, especially for complex, non-linear or unknown processes , see for instance (Von Krannichfeldt, Orehounig, and Fink 2025), (Gijon et al. 2025) or (Markovic et al. 2018) as example with Modelica context. It also improves the ability to simulate and optimize dynamic systems, see e.g. (Singgraval, Suykens, and Geyer 2018), (Aka, Brunnemann, Freund, et al. 2023).

In the Modelica context of state space models, described by a system of ordinary differential equations (ODE), the hybrid model architecture (figure 1) leads to the concept of a Neural ODE (Chen et al. 2018): time derivatives of the states are computed from a combination of equations and neural networks.

There are different possibilities for integrating neural networks into physical system simulations. On the one hand, software manufacturers may develop tool-specific AI interfaces for integrating neural networks into their Modelica simulation environments. On the other hand, there are already different tool-independent, open-source approaches available in the Modelica community, such as the NeuralNetwork (NeuralNetwork 2025) library, which maps neural layers using Modelica blocks, whereby the associated weights and biases are obtained from a read-in matrix, or the SMArtInt library, which implements neural networks directly via external C-functions.

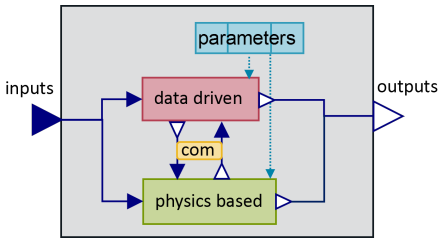


Figure 1. Concept of hybrid modeling: data driven and physics based model parts integrated into a single system model. Internal data exchange via interface variables (com).

The aim of SMArtInt is to create a library that allows integration of trained neural networks directly into Modelica very much in the fashion of input-output blocks. The library is explicitly intended to enable Modelica users to create hybrid models in a simple way while staying inside their preferred Modelica development tool, hence strengthening Modelicas position as multi physics engineering modeling language. Additionally, usage of external C-functions makes the Modelica equation system independent of the size of inferred neural networks, which may in practice contain several thousand parameters. Moreover this approach allows for exchanging the inferred neural network without re-translating the Modelica model, as

long as the input-output structure stays the same.

SMArtInt was developed within a German research collaboration (DIZPROVI 2021-2024)[1] (Brunnemann, Flesch, and Hanke 2024) under the name *SMArtIInt*. The first official release of version 0.1.0 dates back to March 2023.

1.2 Outline of Paper

This paper is structured as follows: Section 2 provides the scope and structure of the library as well as an insight into the functionality of SMArtInt. Section 3 discusses the innovations introduced in version 0.5.1, with a particular focus on the support of the ONNX format. Section 4 presents three different application cases for the library: the integration of a simple feed-forward neural network within a discretized pipe, replacing a heat transfer correlation; a hybrid superheater model trained on measurement data from a coal-fired power plant, developed as part of the DIZPROVI project; and a Balanced Neural ODE used to create a surrogate model of an entire steam cycle. Section 5 presents the SMArtInt funding concept and outlines the additional content of the SMArtInt+ commercial version. Finally, a summary and outlook is given in section 6.

2 Overview of the Library SMArtInt

2.1 Scope of Library

SMArtInt enables the integration of trained artificial intelligence (AI) models into Modelica tools. SMArtInt currently supports TensorFlow models (exported in TFLite format), as well as ONNX models. LiteRT (formerly TensorFlow Lite) (Shuangfeng 2020) was initially selected as the core inference runtime due to its optimizations for embedded systems and edge computing environments, offering a small binary size, high execution efficiency, and support for low-latency inference, while ONNX support was added in the latest version (see section 3) to improve SMArtInt's framework compatibility. For integration, users need to specify the path to the AI model, as well as the input and output dimensions of the neural network within the interface block. The network structure, including layers and neurons, does not need to be defined, allowing neural networks to be easily exchanged as long as the input/output structure remains unchanged. This flexibility enables repeated training with different datasets, hyperparameters, or internal architectures (e.g., layer configurations) without requiring modifications to the Modelica model in which the neural network is used. Inputs and outputs are accessible in Modelica as arrays via the interface blocks, ensuring a simple and flexible integration process.

In hybrid modeling, different levels of hybridization can be distinguished (see figure 2 for illustration). On the top level, entire models can be replaced by data driven surrogates. On the system level, individual submodels and replaceable models can be substituted to implement physical correlations directly from data. Additionally AI models can be used to model detailed physical effects that are usually not accessible at the level of system simulation: thanks to acausality of Modelica a simplified physics based system model may be inverted in order to 'subtract' its result from measurement data. For example the effect of heat exchanger fouling, represented in a simple manner by a model parameter in the original model, may be extracted in more detail as a time series by imposing the measured heat flow and extracting the effective time depending fouling as output of the inverted model. The generated time series may then be used in order to train a detailed data driven fouling model.

The approaches described above were investigated and implemented as part of the DIZPROVI research project.

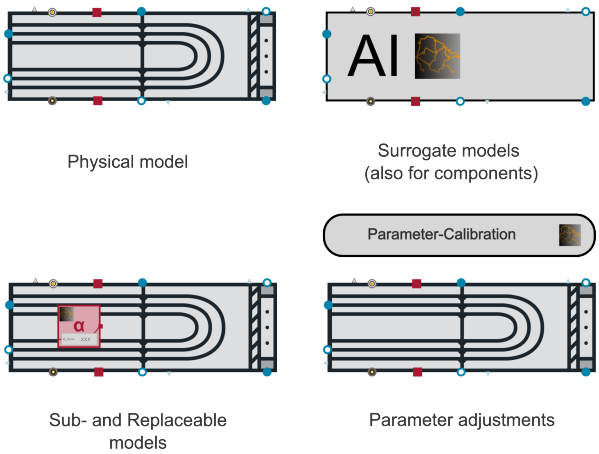


Figure 2. Different levels of hybridization

In the construction of hybrid models care has to be taken, when combining AI models based on measurements of the real system with an abstract physics based system model, subject to several simplifying model assumptions. In practice a thorough investigation is recommended in order to determine the 'best' level of hybridization as well as interface variables between physics and data based mode parts (figure 1). For example it may happen that certain spatially resolved sensor positions are not available in the 1D system model or vice versa certain system variables are not accessible for measurement. Moreover validity of data based model parts is limited to the input range of measurement data. It is also important to realize that AI models usually expect a causal input-output structure, while physics based Modelica models are equation based and acausal. A more detailed discussion of these issues can be found in the DIZPROVI project report (Brunnemann, Flesch, and Hanke 2024).

2.2 Structure of Library

The SMArtInt library is designed to be as simple and user-friendly as possible. Table 1 provides an overview of its top-level structure. The library consists of four packages and a *UsersGuide*, which offers essential information for getting started and effectively using the

[1] funded by the German Federal Ministry of Education and Research under reference number FKZ 03WIR0105E

DOI 10.3384/ecp218459

library.
The *Blocks* package contains interface blocks (as shown in Figure 3) necessary for integrating neural networks into Modelica models. The *Tester* package includes various examples demonstrating the correct usage of these interfaces and their integration into different models.

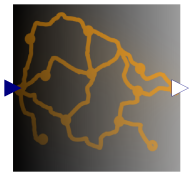


Figure 3. SMartInt's interface block for neural networks

For advanced users, the *BaseClasses* package provides direct access to the core functionalities of the library. Finally, the *Internal* package contains the underlying external C functions that build the base of SMArtInt.

Table 1. SMartInt library structure

SMArtInt	
UsersGuide	Basic information on usage and licenses
Tester	Examples to introduce new users to the library
BaseClasses	Basic models/templates for integrating different types of neural networks in Modelica
Blocks	Blocks to integrate different types of neural networks into Modelica models (based on the *BaseClasses*)
Internal	Internal functions for integrating/interacting with the neural networks

2.2.1 Technical realization

A large part of the functionality of SMArtInt is located outside of Modelica in external C functions, which are integrated into SMArtInt as dynamic libraries (.dll and .so). SMArtInt supports three different neural network types, each of which has its own interface block:

1. Feed-forward neural networks (**FFNNs**): Output depends only on the current input.
2. Recurrent neural networks (**RNNs**): Use previous inputs (time series) to compute the current output.
3. Stateful recurrent neural networks (**sRNNs**): Preserve their hidden state across inference steps to capture long-term dependencies.

SMArtInt operates as a real-time application, processing only the input variables provided by the solver at the current time step. This creates particular challenges regarding the input management of RNNs. This is addressed by the input management system, which supports RNNs as well as stateful RNNs in combination with solvers that utilize variable time step sizes, ensuring seamless integration within Modelica simulations.
Since RNNs must operate at the same temporal resolution as their training data, SMArtInt uses an external C-based container to store input values. If the time step of the Modelica solver exceeds the trained step size of the RNN, the required values are interpolated to maintain consistency. In addition, this input storage mechanism ensures robustness by allowing retrieval of the last accepted values in cases where solver steps are rejected.

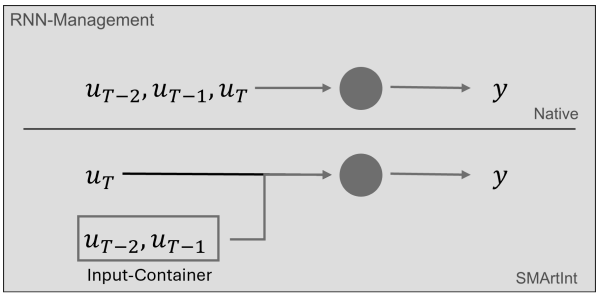


Figure 4. RNN input management

Stateful RNNs are not natively supported in TFLite and have limited functionality in ONNX (ONNX 2025), where network states can only be stored internally for batched inputs. Stateful RNNs must be explicitly restructured by representing layer states as outputs and defining corresponding state inputs. These states are managed externally by the SMArtInt input management system in a special state container, which enables seamless integration while maintaining time dependencies.

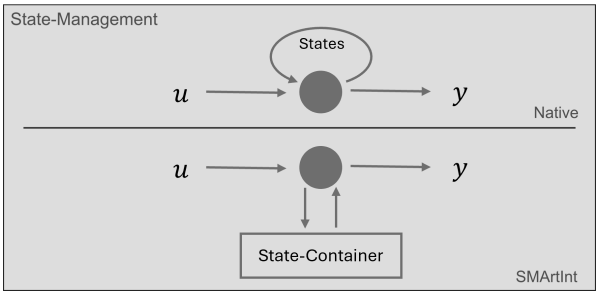


Figure 5. State input management

3 Latest Features of SMArtInt

Version 0.5.1 introduces support for the ONNX format. The Open Neural Network Exchange (ONNX 2025) is an open-source, cross-platform format designed for the exchange of deep learning models. It serves as an interface between various machine learning frameworks, en-

abling model conversion across different formats. The integration of ONNX extends SMArtInt's compatibility beyond TensorFlow (TensorFlow 2025) to include frameworks such as PyTorch (PyTorch 2025), NeoML (NeoML 2025), and many others. A comprehensive list of supported frameworks is available on the ONNX website[2]. SMArtInt automatically detects whether the specified model path corresponds to a TFLite or ONNX model and processes it accordingly. This ensures that Modelica users can work with a unified interface without requiring any additional configuration, simplifying the integration of neural networks into Modelica simulations.

4 Example use cases

In this chapter, three different examples are presented that illustrate different use cases of SMArtInt. Both feed-forward neural networks and recurrent neural networks (RNNs) are integrated into Modelica models. Finally, a complex architecture of neural networks (BNODE) (Aka, Brunnemann, Eiden, et al. 2024) is presented, which can also be implemented with SMArtInt in Modelica.

4.1 FFNN: Nusselt-Correlation

The first example model is a simple, trained feed-forward neural network integrated into Modelica as a TFLite model. It replaces the physical calculation of the Nusselt number with a neural network. The underlying correlation is given by the following relation:

$$\mathrm{Nu} = f\left(\mathrm{Re}, \mathrm{Pr}, \frac{\mathrm{L}}{\mathrm{d}}\right) \tag{1}$$

Figure 6 illustrates the model and the exchangeable heat transfer model used in the pipe. The pipe is discretized into ten segments, resulting in inference being performed ten times per time step on the same instance of the neural network.

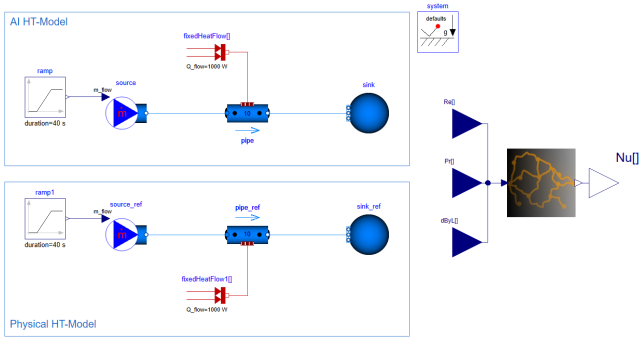


Figure 6. Left: Pipe model with AI Nusselt correlation and physical refernce model. Right: Diagram of the build in data based heat transfer correlation.

The model is primarily based on the SMArtInt example *Tester.PipeHeatTransferExample.TFLite.TestPipe_tflite* and has been extended with a reference pipe and a dynamic mass flow input. During the simulation, a constant heat flow is applied to the pipe, while the mass flow is defined by a ramp function. Figure 7 shows the variation of mass flow rates through the pipe. Initially, the mass flow is constant at 0.2 kg/s, and after 20 seconds, it is linearly increased to 0.9 kg/s over a period of 40 seconds. This dynamic change also impacts the heat transfer, as a higher mass flow with a constant heat flow results in a decrease in temperature within the pipe (as shown in Figure 8). When comparing the resulting temperatures of the pipe segments in Figure 8, no significant difference is observed between the physical and data-driven correlations.

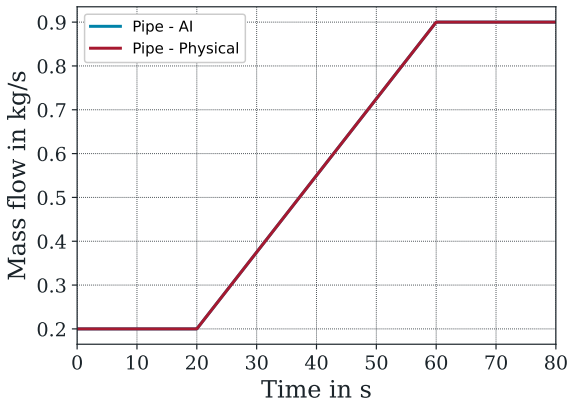


Figure 7. Mass flow in the pipe and the reference pipe

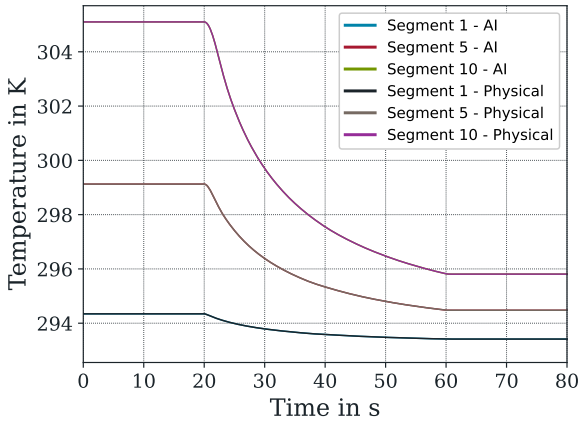


Figure 8. Temperature in the different pipe segments

4.2 sRNN: Superheater heat transfer model

The second example is based on a model (Brunnemann, Koltermann, et al. 2022) that comes from the DIZPROVI (Brunnemann, Flesch, and Hanke 2024) research project, in which SMArtInt also was developed. The project's primary focus was the replacement of physically modeled components with data-driven neural networks as part of a digital power plant twin (illustrated in Figure 9). In this example, the physics-based heat transfer model for

[2]ONNX, supported frameworks and converters: `https://onnx.ai/supported-tools.html#buildModel`

DOI 10.3384/ecp218459

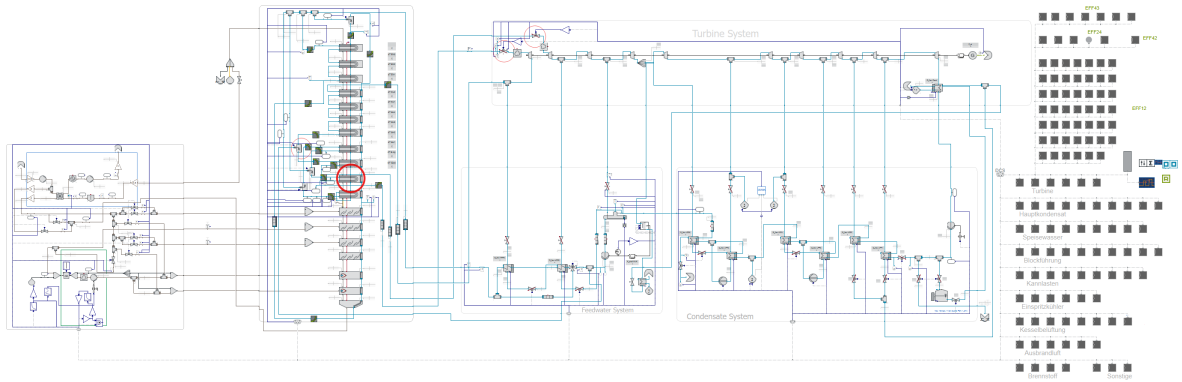


Figure 9. Digital twin of the investigated coal fire power plant, taken from (Brunnemann, Koltermann, et al. 2022)

one of the superheaters (SH4), which relies on energy balance, a Nusselt number correlation, and thermal radiation exchange, was replaced by a recurrent neural network (RNN). Using the inputs of flue gas and steam mass flow, as well as steam enthalpy and flue gas temperature at the superheater inlet, the neural network predicts the transferred heat flow as output. Figure 10 illustrates the predicted heat flow for both the physical and AI-based models within a validation scenario. The detailed control of the model presents a particular challenge for AI-based modeling, as the neural network must produce stable and reliable predictions even in regions outside the training distribution. Due to the mutual dependency between prediction and control, initial inference errors can cause the control logic to drive the system further into areas of low model confidence. This, in turn, degrades prediction quality and may ultimately lead to instability or simulation failure. This leads to two different integration approaches for the AI model:

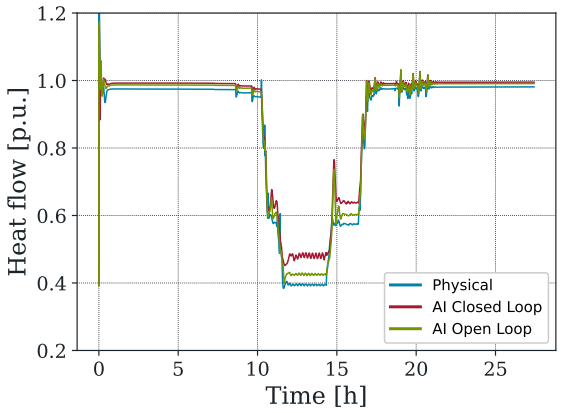


Figure 10. Heat flow SH4 (replaced heat transfer model)

1. Closed-loop integration, where the neural network is directly embedded into the superheater model, allowing interaction with the power plant's control system.

2. Open-loop integration, where the neural network operates independently of the overall model, serving primarily as a performance test for the trained neural network.

The AI-based heat transfer model was trained using data sets derived from measurement data. During data preparation, the spatial distribution of the sensors as well as the four-pass configuration of the evaporator area in the actual system were taken into account. In addition, the measurement data contains information on the degree of fouling and other relevant factors that influence the heat transfer surfaces. Therefore, in contrast to the physically based model, no further model calibration is required.
Figure 10 shows that both the open-loop and closed-loop models align well with the calibrated physical model in the steady-state full-load range. In the partial-load range, however, the heat flow predicted by both AI models exceeds the physically computed heat flow. This discrepancy does not necessarily indicate poor prediction accuracy of the AI model but may instead result from the calibration process of the physical model. For the physical model, the heat flow was adjusted via the fouling factor (CF) of the heat transfer surfaces under steady-state full-load conditions. Consequently, the actual heat flow in the partial-load range of SH4 may deviate from the physically computed and calibrated heat flow. Additionally, certain assumptions made during data processing may have led to lower-quality training data in the partial-load range, further affecting prediction accuracy. More details can be found in the DIZPROVI final report. The comparison of open-loop and closed-loop integration highlights the impact of control. The cooling regulation (CoolerSH3) stabilizes the steam outlet temperature of SH4, reducing thermal stress on the turbines. If heat transfer in SH4 increases, more cooling water is injected, lowering the outlet temperature and improving overall heat transfer. This effect is particularly evident in partial load operation. More detailed insight into the example and the research project can be found in the final report (Brunnemann, Flesch, and Hanke 2024).

4.3 BNODE: Steam-Cycle

The third example builds on the concept of a Balanced Neural ODE (BNODE), introduced by (Aka, Brunnemann, Eiden, et al. 2024). This approach employs a complex architecture composed of multiple interacting neural networks and can be used as a surrogate model for simulating the behavior of complex Modelica systems.
Balanced Neural ODEs combine Neural Ordinary Differential Equations (NODEs) (Chen et al. 2018) with Variational Autoencoders (VAEs) (Kingma and Welling 2019). Instead of approximating the original terms of the ODE directly by a neural network, the state space is mapped into a latent space and the neural network describes an ODE in the latent space. The latent states are then decoded back into the physical space to generate predictions that can be compared to the actual solutions of the ODE. By combining variational autoencoders with state space models, the generalization capability induced by the information bottleneck of VAEs can be used effectively. The information bottleneck created by the numerically defined latent space enforces a compressed, abstract representation of the data and promotes both data locality and the orthogonality of latent dimensions.
The term Balanced Neural ODE (BNODE) refers to the trade-off between state space compression and reconstruction quality, which is established during training and can be controlled via the β parameter. A detailed description of the BNODE concept is provided in the work of (Aka, Brunnemann, Eiden, et al. 2024).

Figure 11 illustrates the concept of the BNODE. The encoders (for state and control) map the input signals and initial system states into the latent space. The state encoder is only required for initialization, as the subsequent latent states are computed by the Neural ODE (NODE). The NODE is a neural network that estimates the state derivatives, from which the next latent states are determined using an ODE solver. These latent states can then be transformed back into the state space using the decoder.

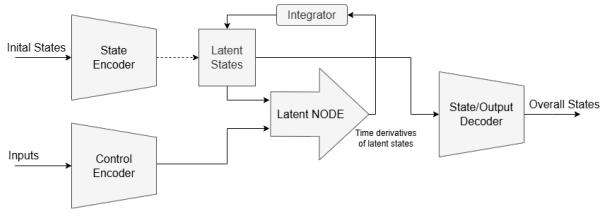


Figure 11. BNODE Concept

For this example, a BNODE was developed for a steam cycle of a coal-fired power plant, based on the concept proposed by (Aka, Brunnemann, Eiden, et al. 2024) and using a training process similar to that described in (Aka 2025). The model replaced by the BNODE originates from the Modelica library ClaRa (ClaRa 2025) (*Examples.SteamCycle_01*) and describes a steam cycle of a coal power plant, considering both mass and energy balances. The original physical model, consisting of 134 states, 29,775 variables, and 8,522 equations, is shown in Figure 12.

Out of the 134 states, the surrogate model approximates 102 states along with 7 additional outputs. Only states related to control elements (e.g., PI controllers), which do not carry essential information about the systems physical state, are omitted. The key parameters of the BNODE surrogate model are listed in the tables 2 and 3.

Table 2. Basic parameters of the BNODE model

Parameter	Value
States	102
Outputs	7
Control inputs	1
Latent states	128
β	0.1

Table 3. Hyperparameter of Encoders, Decoder and Latent ODE

Hyperparameter (for all NNs)	Value
Layers	4
Hidden dimension	64
Activation	ELU

After completion of the training process, the individual neural networks that constitute the BNODE architecture can be exported from the PyTorch format to the ONNX format, which can in turn be integrated into Modelica environments using SMArtInt. The Neural ODE solver employed during training is replaced by the native solver of the Modelica tool, by defining the output of the NODE network as the time derivative of the latent states (see code listing 1).

```
for i in 1:n_lat_states loop
        der(lat_states[i]) =
            LNODE_func.y[i];
end for;
```

Listing 1. Modelica loop of BNODE state derivatives

In addition to the trained networks, the initial states must be provided for model initialization and mapped into the latent space using the state encoder. It is essential that, along with the initial states, the corresponding control input is also specified. After initialization, the control input can be varied within the bounds defined during the training data generation. Figure 13 shows the BNODE brought back to Modelica. The initial states are stored in the *states_tab* table, while the control input is provided in the *P_Target* table. The neural networks that form the BNODE architecture are implemented as individual SMArtInt blocks within the model, and their inputs and outputs are processed at the equation level.

If the same power target (*P_Target*) time series is applied to both the physical model and the surrogate BNODE model, their simulation results can be directly compared. In this example, the control input is initially set to

DOI 10.3384/ecp218459

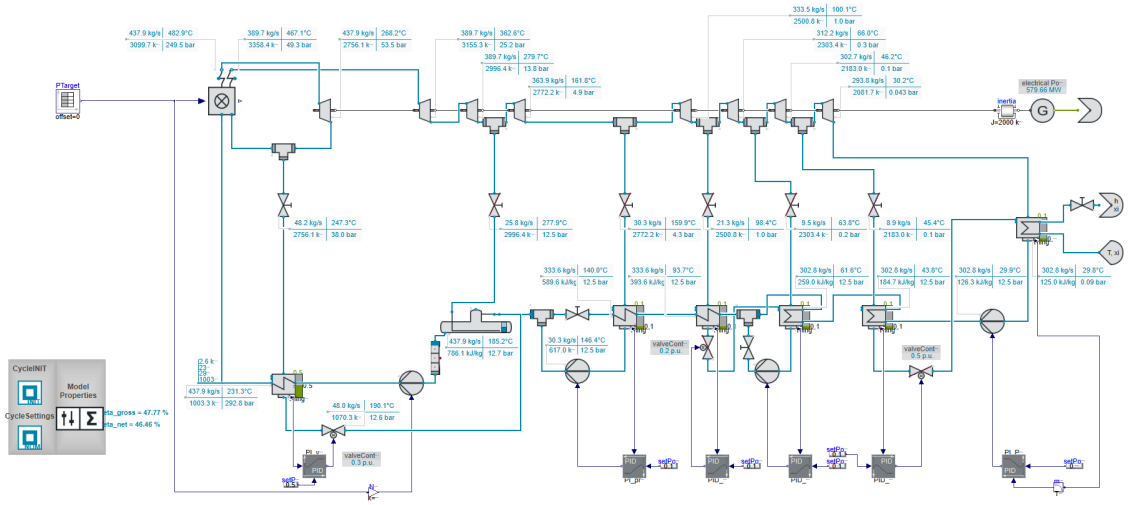


Figure 12. ClaRa: Steam cycle example

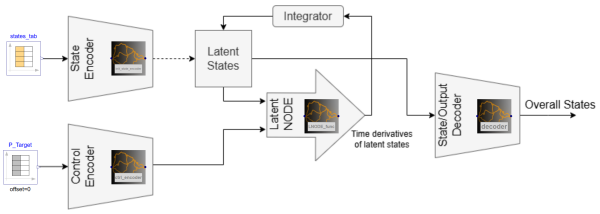


Figure 13. BNODE brought back to Modelica

the BNODE's starting value. After 1000 seconds of simulation time, it is linearly ramped up to 100% (full load) over the course of 100 seconds. Following an additional 900 seconds at full load, the power target of the steam generator is decreased to 70% (partial load), before being increased back to full load at 3000 seconds.
Therefore, the steam cycle displayed by both the BNODE and the physical model experiences a load change. Figure 14 shows the comparison results, including the power target and the seven additional outputs for both the physical model (in red) and the BNODE model (in blue).

The BNODE approximates the results of the physical model quite well, while performing state reduction at the same time. Only 116 of the 128 possible latent state dimensions are utilized. By adjusting the β value, the number of utilized latent state dimensions could be further reduced, although this might result in a decrease in the prediction accuracy of the BNODE. Figure 15 additionally presents a comparison of the computation times of both models. It becomes evident that, especially during dynamic operation, the BNODE requires significantly less computation time compared to the physical model. In this example scenario, the simulation computation time can be reduced by almost a factor of 10 using the BNODE concept. The speed-up factor can increase further in more complex scenarios involving additional load changes, as these have a significant impact on the computation time of the physical model, while leaving the BNODE largely unaffected.

5 The SMArtInt Funding Scheme: Sustaining Open-Source Development

SMArtInt is and will remain an open-source project, continuously evolving with regular updates and releases. To ensure the long-term financial sustainability of its development, it is complemented by a commercial companion version: SMArtInt+ (SMArtIntPlus 2025).
SMArtInt+ extends the functionality of the open-source SMArtInt library by further simplifying its use while also enhancing the quality of the results. The key features of the premium version are listed below:

- Extrapolation Warning
- Modelica Model Generator
- Network Generator (in development)
- Advanced Interfaces
- Tensorflow-Converter (Python)
- Tensorflow-Dedimensionalization-Layer (Python)

As previously mentioned, the individual features of SMArtInt+ are designed to address different aspects of model improvement. One such feature is the **Extrapolation Warning**, which helps ensure the reliability of neural network predictions. Neural networks are highly effective at interpolating within the range of their training data. However, their behavior can get unpredictable when making inferences on unseen data outside that range. In order to identify an extrapolation, SMArtInt+ includes an *Extrapolation Warning* based on three alternative methods: convex hull, kernel density estimation, and a simplified density estimation. These techniques monitor inputs in real time and detect whether the neural network is operating outside its training data.

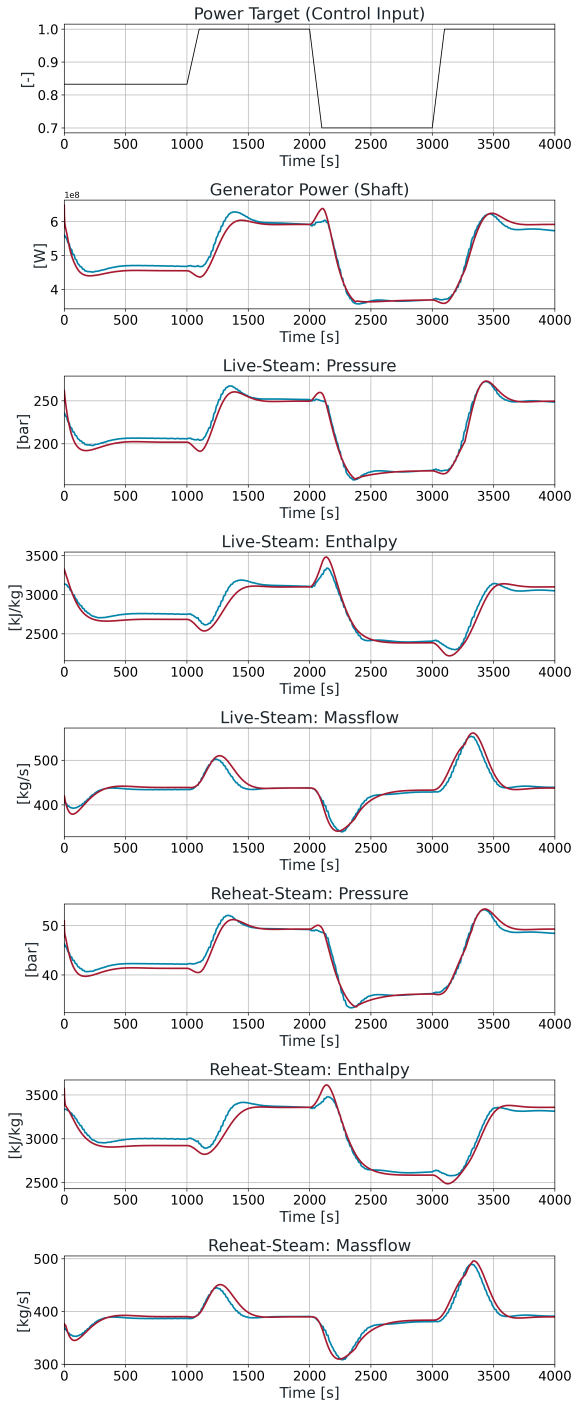


Figure 14. Comparison of different states between physical model (red) and the BNODE surrogate model (blue)

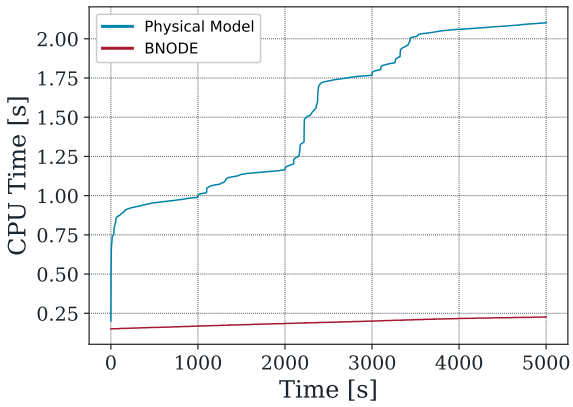


Figure 15. CPU times of the physical model (Dassl-Solver) and the BNODE surrogate model (Euler-Solver)

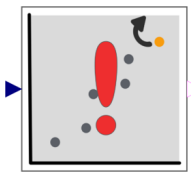


Figure 16. Icon of the ExtrapolationWarning Block

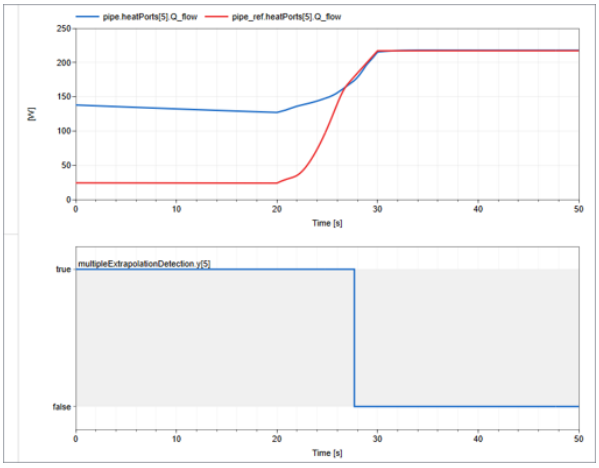


Figure 17. Extrapolation example of a neural network (network prediction and extrapolation curve)

Figure 16 illustrates the extrapolation warning icon triggered during an extrapolation event.
While Figure 17 presents a plot showing its status throughout the entire simulation.

The Modelica **Model Generator** automates the integration of pre-trained neural networks (in TFLite and ONNX format), enabling users to incorporate their models into Modelica with minimal effort. Especially when used in combination with the Network Generator (currently under development) which will allow users to define and train sequential feed-forward and recurrent neural networks with TensorFlow directly in Modelica, the Model Generator becomes a useful tool. SMArtInt+ also features enhanced **interface blocks** equipped with input/output smoothing, integrated scaling, and extrapolation monitoring.

The Modelica library also comes with a **Python-based converter** specifically designed for TensorFlow models with internal states (such as sRNNs), enabling their conversion to TFLite format for seamless integration into SMArtInt (see Section 2.2.1). In addition to the converter, SMArtInt+ includes **dedimensionalization layers (PI-Layer)** (Bakarji et al. 2022), which are custom TensorFlow layers that allow dimensionless quantities to be

DOI
10.3384/ecp218459

used as inputs to neural networks, either automatically or manually. This approach reduces the dimensionality of the underlying problem, enabling the use of smaller neural network architectures with fewer layers or units for the same application. Additionally, dedimensionalization improves the generalization capability of AI models. Figure 18 shows the structure of a neural network wrapped with a PI-layer.

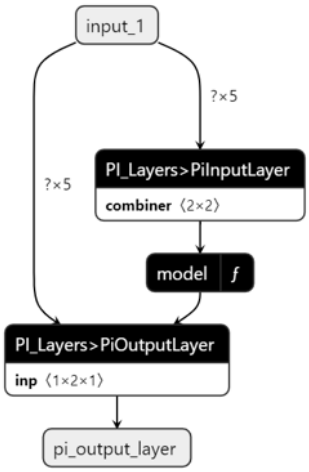


Figure 18. Structure of a neural network wrapped in a dedimensionalization layer

Further information on SMArtInt+ can be found in (SMArtIntPlus 2025).

6 Summary & Outlook

In this paper we have introduced the status of the open-source Modelica library SMArtInt, which enables Modelica users to combine physics based Modelica code with data driven neural networks in a Modelica-based workflow.

With its latest release, users of SMArtInt can now integrate ONNX models into the field of physical Modelica simulations. This enables users to utilize machine learning, deep learning, and other AI techniques from various ONNX-compatible frameworks, thus enhancing the modeling of complex physical systems. The capabilities of SMArtInt are illustrated by examples given in section 4.

SMArtInt is open source and intended to become a Modelica community project and we warmly invite the Modelica community to participate in the libraries improvement by providing user feedback, feature requests and discussing / coding new ideas in order to further enhance Modelicas hybrid modeling capabilities.

The SMArtInt development team will continue development in ongoing and planned research projects.

Acknowledgements

This work has been partially supported by the German Federal Ministry of Education and Research within the DIZPROVI research project under grant number 03WIR0105E.

SPONSORED BY THE
Federal Ministry of Education and Research

The authors thank the anonymous referees for valuable comments and suggestions.

References

Aka, Julius (2025). *Balanced Neural ODEs - Code*. fetched March, 4th 2025. URL: https://github.com/juliusaka/balanced-neural-odes.

Aka, Julius, Johannes Brunnemann, Jörg Eiden, et al. (2024). "Balanced Neural ODEs: nonlinear model order reduction and Koopman operator approximations". In: DOI: https://doi.org/10.48550/arXiv.2410.10174.

Aka, Julius, Johannes Brunnemann, Svenne Freund, et al. (2023-10). "Efficient Global Multi Parameter Calibration for Complex System Models Using Machine-Learning Surrogates". In: *Proceedings of the 15th International Modelica Conference*. Aachen, pp. 107–120. DOI: 10.3384/ecp204107.

Bakarji, Joseph et al. (2022). "Dimensionally Consistent Learning with Buckingham Pi". In: DOI: https://doi.org/10.48550/arXiv.2202.04643.

Brunnemann, Johannes, Robert Flesch, and Tim Hanke (2024). *DIZPROVI - Digitale Zwillinge für Prozessoptimierung und vorausschauende Instandhaltung : Schlussbericht im WIR! - KOI - Verbundprojekt*. Hamburg. DOI: 10.2314/KXP:1919534997.

Brunnemann, Johannes, Jan Koltermann, et al. (2022). "Maschinelles Lernen zur vereinfachten Erstellung und Wartung Hybrider Digitaler Zwillinge (Machine Learning for Simplification of Creation and Maintenance of Hybrid Digital Twin Models)". In: *Proceedings of the 24th Colloquium on Power Plant Technology, Dresden, Germany*. in German.

Chen, Ricky T. Q. et al. (2018). *Neural Ordinary Differential Equations*. DOI: 10.48550/ARXIV.1806.07366.

ClaRa (2025). Ed. by ClaRa Development Team. version 1.8.2, fetched March, 4th 2025. URL: https://github.com/xrg-simulation/ClaRa-official.

Gijon, Alfonso et al. (2025). *Integrating Physics and Data-Driven Approaches: An Explainable and Uncertainty-Aware Hybrid Model for Wind Turbine Power Prediction*. arXiv: 2502.07344 [cs.LG]. URL: https://arxiv.org/abs/2502.07344.

Kingma, Diederik P. and Max Welling (2019). "An Introduction to Variational Autoencoders". In: *Foundations and Trends in Machine Learning* 12.4, pp. 307–392. ISSN: 1935-8237. DOI: 10.1561/2200000056. URL: http://dx.doi.org/10.1561/2200000056.

Kong, Xiangjie et al. (2025-02). "Deep learning for time series forecasting: a survey". In: *International Journal of Machine Learning and Cybernetics*. ISSN: 1868-808X. DOI: 10.1007/s13042-025-02560-w. URL: http://dx.doi.org/10.1007/s13042-025-02560-w.

Kurz, Stefan et al. (2022). "Hybrid modeling: towards the next level of scientific computing in engineering". In: *Journal of Mathematics in Industry* 12.1, p. 8. DOI: 10.1186/

s13362-022-00123-0. URL: https://mathematicsinindustry.springeropen.com/articles/10.1186/s13362-022-00123-0.

Lam, Remi et al. (2023). *GraphCast: Learning skillful medium-range global weather forecasting*. arXiv: 2212.12794 [cs.LG]. URL: https://arxiv.org/abs/2212.12794.

Markovic, Romana et al. (2018). "Window Opening Model using Deep Learning Methods". In: https://doi.org/10.48550/arXiv.1807.03610. DOI: https://doi.org/10.48550/arXiv.1807.03610.

Matei, I. (2018). "Learning algorithms for physical systems: challenges and solutions". In: *Palo Alto Research Center*. https://www.naefrontiers.org/190462/Matei-presentation.

NeoML (2025). Ed. by ABBYY. fetched March, 4th 2025. URL: https://www.abbyy.com/neoml/.

NeuralNetwork (2025). Ed. by Fabio Codecï¿½, Politecnico Di Milano. fetched March, 4th 2025. URL: https://github.com/modelica-3rdparty/NeuralNetwork.

ONNX (2025). Ed. by onnx. fetched March, 4th 2025. URL: https://onnx.ai/.

PyTorch (2025). Ed. by pytorch. fetched March, 4th 2025. URL: https://pytorch.org/.

Quarteroni, Alfio, Paola Gervasio, and Francesco Regazzoni (2025). "Combining physics-based and data-driven models: advancing the frontiers of research with scientific machine learning". In: *Mathematical Models and Methods in Applied Sciences* 35.04. DOI: 10.1142/s0218202525500125.

Rudolph, Maja, Stefan Kurz, and Barbara Rakitsch (2024). "Hybrid modeling design patterns." In: *J.Math.Industry* 14. DOI: https://doi.org/10.1186/s13362-024-00141-0.

Shuangfeng, Li (2020). "TensorFlow Lite: On-Device Machine Learning Framework". In: *Journal of Computer Research and Development* 57.9, pp. 1839–1853. ISSN: 1000-1239. DOI: 10.7544/issn1000-1239.2020.20200291. URL: https://crad.ict.ac.cn/en/article/doi/10.7544/issn1000-1239.2020.20200291.

Singraval, Sundaravelpandian, Johan Suykens, and Philipp Geyer (2018-10). "Deep-learning neural-network architectures and methods: Using component-based models in building-design energy prediction". In: *Advanced Engineering Informatics* 38, pp. 81–90. ISSN: 1474-0346. DOI: https://doi.org/10.1016/j.aei.2018.06.004.

SMArtInt (2025). Ed. by XRG Simulation GmbH. version 0.5.1, fetched March, 4th 2025. URL: https://github.com/xrg-simulation/SMArtInt.

SMArtIntPlus (2025). Ed. by XRG Simulation GmbH. version 0.1.0. URL: https://xrg-simulation.de/en/seiten/smartint.

TensorFlow (2025). Ed. by tensorflow. fetched March, 4th 2025. URL: https://github.com/tensorflow.

Von Krannichfeldt, Leandro, Kristina Orehounig, and Olga Fink (2025-08). "Combining physics-based and data-driven modeling for building energy systems". In: *Applied Energy* 391, p. 125853. ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2025.125853. URL: http://dx.doi.org/10.1016/j.apenergy.2025.125853.

Zero-Shot Parameter Estimation of Modelica Models using Patch Transformer Networks

Ankush Chakrabarty[1] Marco Forgione[2] Dario Piga[2] Alberto Bemporad[3] Christopher Laughman[1]

[1]Mitsubishi Electric Research Labs (MERL), Cambridge, USA {chakrabarty, laughman}@merl.com
[2]SUPSI, IDSIA - Dalle Molle Institute for Artificial Intelligence, Lugano, Switzerland. {marco.forgione, dario.piga}@supsi.ch
[3]IMT School for Advanced Studies Lucca, Lucca, Italy. {alberto.bemporad}@imtlucca.it

Abstract

This paper introduces a transformer-based generative network for rapid parameter estimation of Modelica building models using simulation data from a Functional Mock-up Unit (FMU). Utilizing the `MixedAirCO2` model from the Modelica Buildings library, we simulate a single-zone mixed-air volume with detailed thermal and CO_2 dynamics. By varying eight physical parameters and randomizing occupancy profiles across 100 simulated systems, we generate a comprehensive dataset. The transformer encoder, informed by room temperature and CO_2 concentration outputs, predicts the underlying physical parameters with high accuracy and without re-tuning (hence, "zero-shot"). This approach eliminates the need for iterative optimization or can be used to warm-start such optimization-based approaches, enabling real-time control, monitoring, and fault detection in FMU-based workflows.

Keywords: transformer networks, dynamic simulation, system identification, model calibration, functional mockup interface, machine learning.

1 Introduction

Accurate modeling of building energy systems is critical for the design and operation of advanced control strategies such as model predictive control (MPC), fault detection, and digital twins (Zhan, Chakrabarty, et al. 2023). Physics-based modeling tools, such as those based on the Modelica language, offer a powerful framework for capturing the thermal, ventilation, and air quality dynamics of buildings through component-based system representations. However, the reliability of these models depends heavily on accurate parameter values, which can be challenging to determine in practice due to limited access to detailed building specifications, aging infrastructure, or changes in operational conditions over time (Chakrabarty et al. 2021; Wüllhorst et al. 2022).

Parameter estimation methods bridge this gap by calibrating model parameters to match observed output data. In the context of Modelica models exported as Functional Mock-up Units (FMUs) (Blochwitz et al. 2011), parameter estimation typically involves executing repeated simulations while adjusting model parameters to minimize the mismatch between simulation outputs and measured signals, such as temperature or CO_2 concentration. Classical approaches—including gradient-based optimization (Cañas et al. 2023), Bayesian calibration (Li et al. 2025), and black-box global optimization methods—have been successfully applied in this domain (Chakrabarty et al. 2021). Tools like RAPID (Vanfretti et al. 2016) provide modular ways to link external optimization routines to FMU simulations, supporting a range of problem formulations from gray-box modeling to model order reduction (Bres 2021). Despite their success, these methods can be computationally expensive, particularly when applied to large-scale simulation studies or real-time control tasks, due to their reliance on iterative optimization loops (Balali et al. 2023). Additionally, the authors of (Zhang and Mikelsons 2023) present a sensitivity-guided, iterative framework that combines BayesFlow and a physics-enhanced variational auto-encoder to improve parameter identification and data generation for model calibration. Their method enhances efficiency and accuracy by focusing on the most sensitive parameters during each iteration.

Herein, we study a fundamentally different paradigm: using transformer networks to learn the mapping from system outputs to model parameters directly from Modelica simulations. The idea is rooted in the principle that, for a well-defined family of parameterized models, there exists a functional relationship between the parameters and the corresponding output trajectories under typical excitation. By simulating a large number of such model instances offline, varying both physical parameters and disturbance conditions, we can train a transformer network to approximate the inverse mapping: from observed outputs back to the parameters that generated them. This multi-source parameter estimation has been investigated in the context of Modelica-based dynamical simulation only recently (Zhan, Wichern, et al. 2022; Li et al. 2025).

We focus on the use of transformer architectures, which have recently emerged as state-of-the-art tools for modeling sequential data in natural language processing, time-series forecasting, and symbolic regression (Vaswani et al. 2017). Transformer networks are a generative machine learning tool that excel at capturing long-range dependencies and temporal patterns, making them well-suited for analyzing

building simulation data, which often involves complex interactions between building materials and fluid flows, occupancy, and external weather conditions over extended time periods. Moreover, transformers have demonstrated impressive "zero-shot" capabilities: they often generalize well to entirely new forecasting or control tasks even without any additional fine-tuning. By leveraging pretraining on diverse datasets, they can transfer learned representations of temporal structure and physical dynamics to novel scenarios (such as predicting thermal loads in previously unseen building types or adapting to different climate conditions) simply by framing the problem in a compatible input format. This flexibility not only accelerates deployment in new environments but also reduces the need for costly labeled data when extending models to emerging use cases.

To evaluate our transformer-based approach to parameter estimation, we consider a simplified room-level building model implemented in Modelica and exported as an FMU. The model simulates the dynamics of air temperature and CO_2 concentration in a mixed-air volume, subject to external disturbances such as solar radiation and time-varying occupancy. Eight key physical parameters are varied across 2000 simulations: floor area, room height, room volume, external wall thermal resistance, window U-factor, window solar absorption fraction, exhaust air mass flow rate, and CO_2 emission rate per person. In each simulation, occupant presence schedules are also randomized to introduce realistic variability in internal gains. The output signals from each run—specifically, the room air temperature and CO_2 concentration—are collected at five-minute intervals for a full week, resulting in multichannel time series of high resolution. These serve as inputs to a patch-wise transformer encoder, followed by a multi-layer regression head that predicts the underlying parameter vector.

Compared to traditional parameter estimation methods, our approach offers two key advantages. First, it decouples the expensive optimization loop from the online estimation process. Once trained, the model can infer parameters from new output signals in a single forward pass, enabling rapid estimation that is suitable for time-sensitive applications such as adaptive control or live performance monitoring. Second, the neural network can leverage the full temporal structure of the input signals, including cross-channel correlations and delayed dependencies, which may be difficult to exploit effectively with handcrafted objective functions or scalar error metrics used in classical methods. While this approach does not replace traditional calibration frameworks, particularly in applications requiring safety-critical design, it offers a complementary tool for use cases where speed and scalability are paramount. Moreover, since the method is data-driven but trained entirely on simulated data, it mitigates the need for large amounts of real-world measurement data, making it practical in warm-starting optimization-based parameter estimation algorithms.

By integrating modern machine learning with established Modelica modeling workflows, this work aims to expand the toolkit available for building energy analysis, enabling faster, scalable, and data-informed modeling workflows for real-world systems. Specific contributions of this paper include: (i) proposing a transformer-based neural architecture for estimating physical parameters from building simulation output time series; (ii) demonstrating the effectiveness of zero-shot parameter estimation on a dataset generated using a Modelica FMU of a mixed-air room model, with randomized physical parameters and occupancy profiles, and (iii) empirical validation of the trained model, showing that it can accurately recover eight key physical building parameters that may vary as the building ages, or are difficult to measure, despite significant climate and occupant-induced variation.

The remainder of this paper is organized as follows. Section 2 formulates the parameter estimation problem and describes our FMU-based simulation model, including the `MixedAirCO2` benchmark, the procedure for exporting it as an FMU, and the generation of a large, randomized dataset for pretraining. Section 3 details the design of our patch-wise transformer encoder and regression head, including embedding, positional encoding, attention layers, and pooling strategy. Section 4 presents the experimental setup and results: we begin by assessing zero-shot estimation accuracy across a range of context lengths (:= sequences of measurement data collected for parameter estimation, to be used as contextual inputs for the generative network), then benchmark our transformer against non-transformer baselines to highlight the superiority of attention-based models in achieving robust zero-shot performance under varying data quantities. Section 5 summarizes our conclusions and outlines directions for future work, including extensions to probabilistic transformers for uncertainty quantification.

2 Motivation

2.1 Problem Statement

We consider a parameterized dynamical system represented by a Modelica-based Functional Mock-up Unit (FMU):

$$\mathbf{y}(t) = f(\mathbf{p}, \mathbf{d}(t), t), \tag{1}$$

where $\mathbf{p} \in \mathbb{R}^m$ is a vector of unknown physical parameters, $\mathbf{d}(t)$ denotes time-varying disturbances (e.g., occupancy-driven internal gains), and $\mathbf{y}(t) \in \mathbb{R}^n$ are the observable outputs over time such as indoor air temperature and CO_2 concentration. Our goal is to learn a parameter estimator $\hat{g}_\theta : \mathbb{R}^{T\times n} \to \mathbb{R}^m$ that maps output trajectories $\mathbf{y}_{1:T} = \{\mathbf{y}(t_1), \ldots, \mathbf{y}(t_T)\}$ to estimates of the underlying parameters. Formally, we write this as

$$\hat{\mathbf{p}} = \hat{g}_\theta(\mathbf{y}_{1:T}) \approx \mathbf{p}, \tag{2}$$

where θ denotes some parameterization of the map $\hat{g}$, e.g., weights of a deep neural network. We reiterate that unlike traditional optimization-based parameter estimation methods which solve an inverse problem for each new instance

DOI
10.3384/ecp218469

of contextual data $\mathbf{y}_{1:T}$, we propose to learn this mapping directly from synthetic data generated offline.

2.2 FMU-based Multi-Source Data Collection

We use an FMU extracted from the `Examples.MixedAirCO2` model from the Modelica `Buildings 11.0` library (Wetter et al. 2014) as the simulation model with default TMY3 file for meteorological data; see Figure 1 for the graphical layer representation in Dymola. This model represents a single thermal zone with a detailed energy and CO_2 balance, incorporating contributions from internal gains, air infiltration, ventilation, and weather-driven boundary conditions. The model includes thermal mass, surface convection and radiation, and moisture dynamics, making it: (i) representative of building simulation use cases, and (ii) easily extendable to multi-zone office buildings.

The zone is served by a mixing box that supplies outdoor air and recirculated return air based on a CO_2-based control strategy. To perform parameter estimation under realistic operating conditions, we utilize closed-loop data, where the PI controller remains active with gain constants carefully tuned to balance tracking performance and disturbance rejection. We also add small-gain sinusoids to the control action in order to provide some excitation for the inputs. We already know that the outputs are sufficiently sensitive to the persistently exciting inputs and will induce persistently excited outputs that we will use for learning. This configuration preserves feedback actuation, as evident from the Fig. 1, and allows the system's response to exogenous inputs (e.g., occupancy profiles, weather disturbances, and internal loads) to be interpreted in the context of both the building's physical properties and the control dynamics. This setup is not only realistic, but the identification problem is challenging because of varying occupancy profiles and weather conditions.

The physical parameters varied in each simulation are summarized in Table 1. Each parameter is sampled uniformly within the specified range. Each simulation spans 1 year of operation, sampled every 60 minutes (3600 seconds), resulting in $T = 8760$ time steps per simulation. The output signal at each time step is the room air temperature T and the CO_2 concentration: $\mathbf{y}(t) = \begin{bmatrix} T(t) & CO_2\,(t) \end{bmatrix}$.

Occupant-driven internal gains are also randomized by varying: (i) occupant count (persons), (ii) start time and duration of occupancy, (iii) daily schedules. However, occupant information is not provided to the model at inference time, reflecting practical limitations.

2.3 FMU Export Procedure

To enable parameter estimation using external (to Modelica) optimization tools, we export the `MixedAirCO2` model as a functional mock-up unit (FMU) using Dymola 2024 (Dassault Systèmes 2023). The following steps summarize the FMU export process:

1. CONTROL CONFIGURATION. We disable the internal PI CO_2 controller.

2. PARAMETER EXPOSURE. Key physical parameters targeted for estimation are re-declared as `input Real` variables within the Modelica model. This allows external modification during estimation without having to continuously recompile the FMU.

3. SOLVER AND TIME SETTINGS. Simulation is configured for a 1-year period with a storage step size of 1 hr to match the measurement data resolution. The CVODE solver (Cohen, Hindmarsh, Dubois, et al. 1996) is selected for robust handling of stiff system dynamics during FMU export and tolerances and solver step-sizes are set to default.

4. FMU GENERATION. The FMU is generated using Dymola's `Advanced` menu with the following settings:

   - FMI version: `2.0`
   - FMI type: `Model Exchange`
   - Solver: `CVODE`

5. VERIFICATION. The resulting FMU is validated using `fmpy`[1] in Python 3.11 to confirm correct loading, access to exposed parameters, and consistency of simulated outputs with the original Modelica model under default disturbance input profiles.

In summary, this export procedure ensures that the physics-oriented Modelica model is encapsulated in a reusable and differentiable FMU format, suitable for integration with external state-of-the-art deep learning tools for which widespread Modelica support at the time of writing remains limited.

2.4 Multi-Source Dataset and Dataloader

Figure 2 presents multiscale building simulation outputs: daily temperature cycles with subtle day-to-day shifts, sharp occupancy-driven internal-gain spikes, multi-day ambient-temperature fronts, and a nonlinear, saturating heating response. This highlights the complex disturbances and actuator dynamics that make parameter estimation challenging.

The collected dataset is split into training (80 systems with uniquely sampled parameters), validation (10 systems), and test sets (10 systems). Channel-wise normalization is applied based on the training data statistics. During evaluation, model accuracy is assessed by comparing predicted parameter vectors $\hat{\mathbf{p}}$ to ground truth vectors $\mathbf{p}$ over the test set. We found it easier to work with logarithmic transforms of the parameters than the parameters themselves due to their values showing varying orders of magnitude.

[1] https://github.com/CATIA-Systems/FMPy

Table 1. Physical parameters varied in simulation experiments. Each parameter is sampled uniformly within the specified range.

Parameter Name	Description	Unit	Range
`gaiCO2.k`	CO_2 emission rate per person	kg/s	$[6 \times 10^{-6},\ 1.2 \times 10^{-5}]$
`roo.AFlo`	Floor area	m^2	[60.0, 200.0]
`roo.hRoo`	Room height	m	[2.5, 4.0]
`roo.V`	Room volume	m^3	[40.0, 300.0]
`matExtWal.R`	External wall thermal resistance	$K \cdot m^2/W$	[0.5, 3.0]
`glaSys.UFra`	Window U-value	$W/m^2 \cdot K$	[1.0, 3.0]
`glaSys.absSolFra`	Window solar absorptance	–	[0.3, 0.7]
`mOut_flow`	Exhaust air mass flow rate	kg/s	[0.05, 0.5]

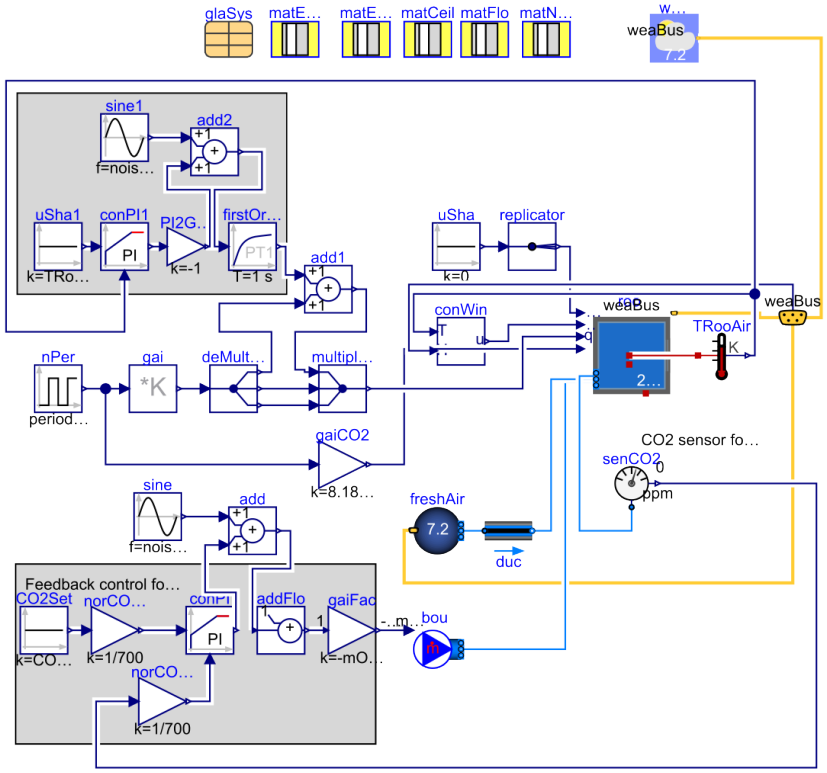


Figure 1. Graphical representation of adapted `MixedAirCO2.mo` model from Modelica `Buildings` Library, with additional PI control loop closed on room air temperature.

Data loading is implemented via `PyTorch 2.0`'s custom `Dataset` class that wraps the full collection of multivariate time-series records and their corresponding target vectors. Rather than returning entire sequences, the `Dataset`'s length is defined by a "virtual epoch" size, and each call to `getitem` randomly selects one series and then extracts a contiguous subsequence of fixed length from it. This subsequence is returned together with the associated target vector. By sampling both the series index and the time-window start position at each access, the `Dataset` generates a diverse set of training examples each epoch without explicit shuffling. We then wrap this `Dataset` in a standard `DataLoader` with a chosen batch size, yielding batches of subsequences and targets for efficient mini-batch training. Randomization inside the `Dataset` ensures variety in each batch.

More concretely, given contextualizing measurement inputs $\mathbf{Y} \in \mathbb{R}^{Q \times T \times n}$ and static outputs $\mathbf{P} \in \mathbb{R}^{Q \times m}$ across Q systems, each training sample is constructed as follows:

1. A system index $q \in \{1, \ldots, Q\}$ is drawn uniformly at random.
2. A start index $t_0 \in \{0, \ldots, T - L\}$ is sampled uniformly.
3. A temporal window $\mathbf{y}_{q,t_0:t_0+L-1} \in \mathbb{R}^{L \times n}$ is extracted.
4. The target is the corresponding output vector $\mathbf{p}_q \in \mathbb{R}^m$.

The dataset returns tuples $(\mathbf{Y}, \mathbf{P})$ with fixed window length L and configurable epoch size M.

DOI
10.3384/ecp218469

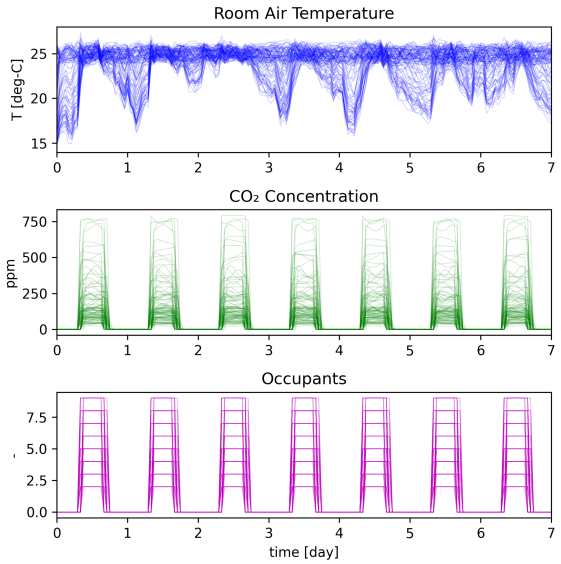


Figure 2. Example multi-system simulation outputs from FMU over one week; with loops closed on carbon dioxide and room air temperature. The outputs have been induced by time-varying disturbances, including fluctuating internal gains, multi-day ambient temperature fronts, and heating power responses.

3 Transformer-Based Parameter Estimator

We employ a patch-wise transformer encoder for parameter estimation from multivariate time series, followed by a deep multilayer perceptron (MLP) head for final prediction. The full architecture is referred to as the *patch transformer*, and is composed of five main components:

1. PATCH EMBEDDING. The input time series $\mathbf{Y} \in \mathbb{R}^{B\times T\times n}$ is separated into patches by a 1-D convolution: $\mathbf{z}_0 = \text{Conv1D}_{\text{stride}=\ell_p,\text{kernel}=\ell_p}(\mathbf{Y}^\top)$, where ℓ_p is the patch size and T is the sequence length. This transforms the input to $B \times (T/\ell_p) \times d$.

2. POSITIONAL ENCODING. A sinusoidal positional encoding $\mathbf{e} \in \mathbb{R}^{T/\ell_p \times d}$ is added to the patch embeddings, providing temporal awareness.

3. TRANSFORMER ENCODER BACKBONE. A stack of L standard transformer encoder layers as described in (Vaswani et al. 2017), each with H attention heads and feedforward width d_{ff}, processes the sequence $\mathbf{z}_L = \text{TransformerEnc}(\mathbf{z}_0 + \mathbf{e})$.

4. GLOBAL POOLING. A global average pooling layer aggregates the token outputs:

$$\mathbf{h} = \frac{1}{T/\ell_p} \sum_{i=1}^{T/\ell_p} \mathbf{z}_L^{(i)}.$$

5. REGRESSION HEAD. A MLP with LeakyReLU activations and LayerNorm at the input maps the pooled embedding to the target vector $\hat{\mathbf{p}} \in \mathbb{R}^{B\times m}$.

Our patch transformer is inspired by the Patcher model introduced in (Ou et al. 2022), which employs a patch-wise transformer encoder. Unlike the Patcher model, we replaced the mixture-of-experts decoder with a regression head, which is simply composed of fully connected layers.

The hyperparameters used in the patch transformer estimator are listed in Table 2. We did not perform extensive hyperparameter search, and expect that a smaller network will likely suffice if optimized by Bayesian neural architecture search (Nasrin et al. 2022) or an automated tool like Optuna (Akiba et al. 2019). All linear and convolutional layers are initialized using Xavier normal initialization (Glorot and Bengio 2010). Biases are zero-initialized.

Table 2. Patch Transformer Hyperparameters

Hyperparameter	**Value**
Input channels C	2
Patch size ℓ_p	12
Embedding dimension d	64
Number of Transformer layers L	4
Number of attention heads H	8
Feedforward dimension d_{ff}	256
Dropout	0.2
Output dimension d_{out}	8
Activation	LeakyReLU(α=0.2)
Positional Encoding	Sinusoidal (fixed)
Pooling	Global average pooling

The patch transformer parameter estimator combines localized feature extraction and global sequence modeling, which is critical for capturing the complex temporal dynamics inherent in multivariate time series from building systems. Patch embedding via 1D convolution reduces sequence length and emphasizes local temporal correlations, while the transformer encoder captures long-range dependencies using self-attention, avoiding the limitations of recurrence-based methods. The use of sinusoidal positional encoding preserves temporal order, and global average pooling enables efficient dimensionality reduction and regularization. A deep MLP regressor maps the aggregated latent representation to parameter estimates, allowing flexible modeling of non-linear, non-convex mappings typical in grey-box system identification.

Compared to alternatives, this architecture provides several advantages. Unlike RNN-based models (e.g., LSTM, GRU), it enables full parallelization during training and better handles long-range dependencies without gradient degradation. CNN-only approaches lack adaptability to temporal variation, and MLPs trained on raw sequences fail to encode temporal structure. The transformer's ability to dynamically attend across patches, combined with the efficiency of patch-wise tokenization, makes it more expressive and scalable. This inductive bias, along with its modular structure, is expected to offer improved accuracy and robustness over conventional baselines for zero-shot parameter estimation in building energy models.

4 Results and Discussion

All experiments were conducted on a MacBook Pro equipped with an Apple M3-Max system-on-a-chip featuring 16 cores (12 performance cores and 4 efficiency cores) and 64 GB of unified memory. Training on CPU for the patch transformer required between 1-2h. The Adamax optimizer is used for training with a learning rate of 0.001 for 2000 training iterations, the batch size is $B = 16$.

4.1 Test Setup

We evaluate the trained transformer regressor on held-out time–series snippets drawn from the same one-year simulation of 100 independent systems. Each snippet spans a fixed "context window" of length L (in hours), chosen from $\{24, 48, 96, 168, 360, 720\}$. For each L:

1. We randomly sample $N_{\text{win}} = 1000$ disjoint windows of length L from the full dataset, forming a *test* subset.

2. The network (whose inputs had been normalized and whose outputs were predicted in base-10 logarithmic scale) is applied in inference mode to each window, yielding $\widehat{\mathbf{p}}_{\log 10}$. We recover physical parameter estimates by
$$\widehat{\mathbf{p}} = 10^{\widehat{\mathbf{p}}_{\log 10}}.$$

3. The true parameter vector $\mathbf{p}$ for each system is likewise converted from $\log_{10}$ back to the original units.

4. We compute two metrics over the test set:

   - The coefficient of determination for each parameter
   $$\mathsf{R}^2 = 1 - \frac{\sum_{i=1}^{N_{\text{win}}} \left(p_i - \widehat{p}_i\right)^2}{\sum_{i=1}^{N_{\text{win}}} \left(p_i - \bar{p}\right)^2},$$
   where $\bar{p}$ is the sample mean of the true values.
   - The relative accuracy for each estimate,
   $$\mathsf{A}_i = 1 - \frac{\left|p_i - \widehat{p}_i\right|}{p_i}, \tag{3}$$
   which is 1 for a perfect match and is smaller as the absolute relative error grows.

4.2 Patch Transformer Results

In Fig. 3, we test the quality of the outputs generated by the FMU over 21 days. We first use 7 days as an input to the transformer model, as illustrated by the blue trace, and from this infer a set of predicted parameters for the model. Note that for this test we switch off persistently exciting noise on the controls to promote realism. By simulating with the predicted parameters for the 14 days after the 7 days of context input, we see that the error in the predictions is small, as is evident by the overlap between the orange dashed predicted trajectory and the blue solid true trajectory. Figure 4 presents scatter plots comparing predicted versus true values for eight key parameters of a thermal building model under varied contextual inputs. Each subplot corresponds to a distinct physical parameter. Red dots denote individual predictions, while the solid black line indicates the ideal identity mapping (i.e., perfect prediction). Each subplot reports the coefficient of determination (R^2) to quantify regression accuracy. The model demonstrates excellent predictive performance ($\mathsf{R}^2 \geq 0.95$) across most parameters, with only a modest drop in performance for the room height. The relatively lower score likely arises from its weaker influence on the observable outputs used for inference. Unlike parameters such as floor area or ventilation mass flow rate, which directly impact air volume exchange and thermal load profiles, room height has a more indirect effect, primarily through its contribution to overall room volume and stratification effects, which may not be strongly expressed in the measured outputs (e.g., temperature, CO_2 levels) under the simulation conditions. Additionally, identifiability may be confounded when both area and height jointly influence volume, making it difficult for the model to disentangle their individual contributions. This leads to a higher spread in predicted values for room height and hence a lower R^2 score. Overall, these results validate the model's high-fidelity parameter inference capability across diverse building configurations.

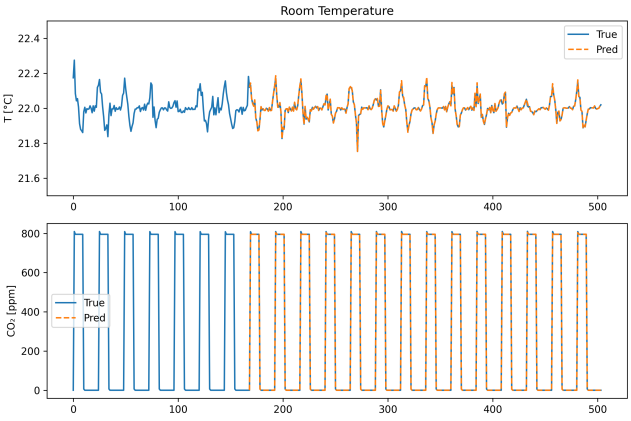


Figure 3. Predicted (orange) vs. true (blue) simulated outputs with a 7-day context set of measurements, and 14-day predictions.

Figure 5 presents boxplots of relative accuracy (3) for each parameter, evaluated over context windows of 1, 2, 4, 7, 15, and 30 days. Each box illustrates the interquartile range of relative accuracy across all test samples, with the median indicated by a horizontal line. As context length increases, inference accuracy improves consistently across all parameters, with median values approaching 1.0 and reduced variability. This trend is especially pronounced between 1 and 7 days, after which gains begin to saturate. Parameters such as exhaust mass flow rate (`mout.flow`) and CO_2 generation rate (`gaiCO2`) exhibit high identifiability even with short contexts, reflecting their strong observability through direct influence on air quality dynamics. In contrast, geometric and envelope-related parame-

DOI
10.3384/ecp218469

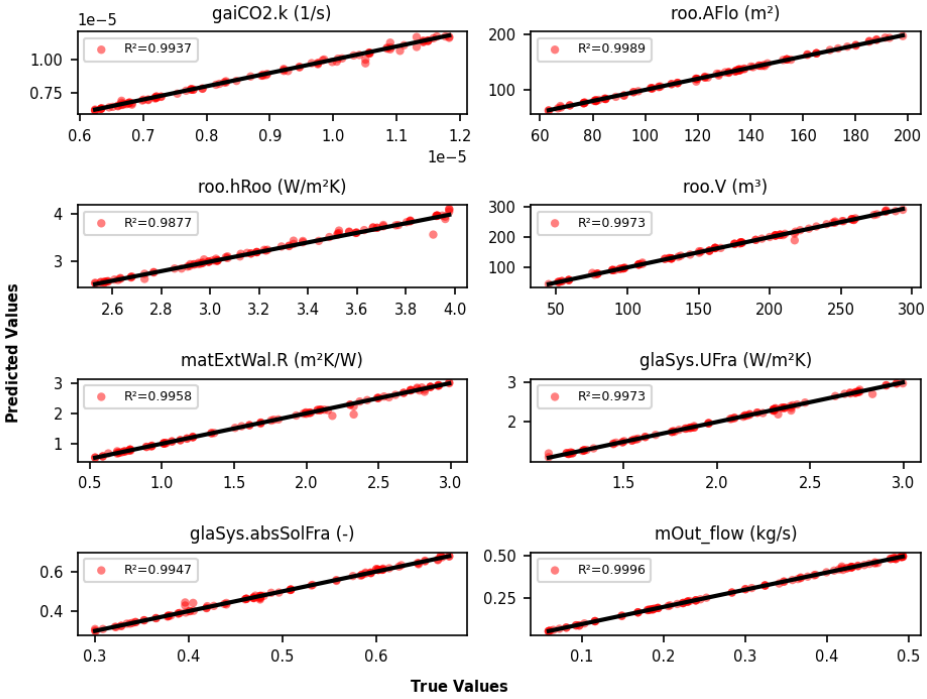


Figure 4. R^2 scatter plots (true vs. predicted) for each parameter, with 48 hr context windows.

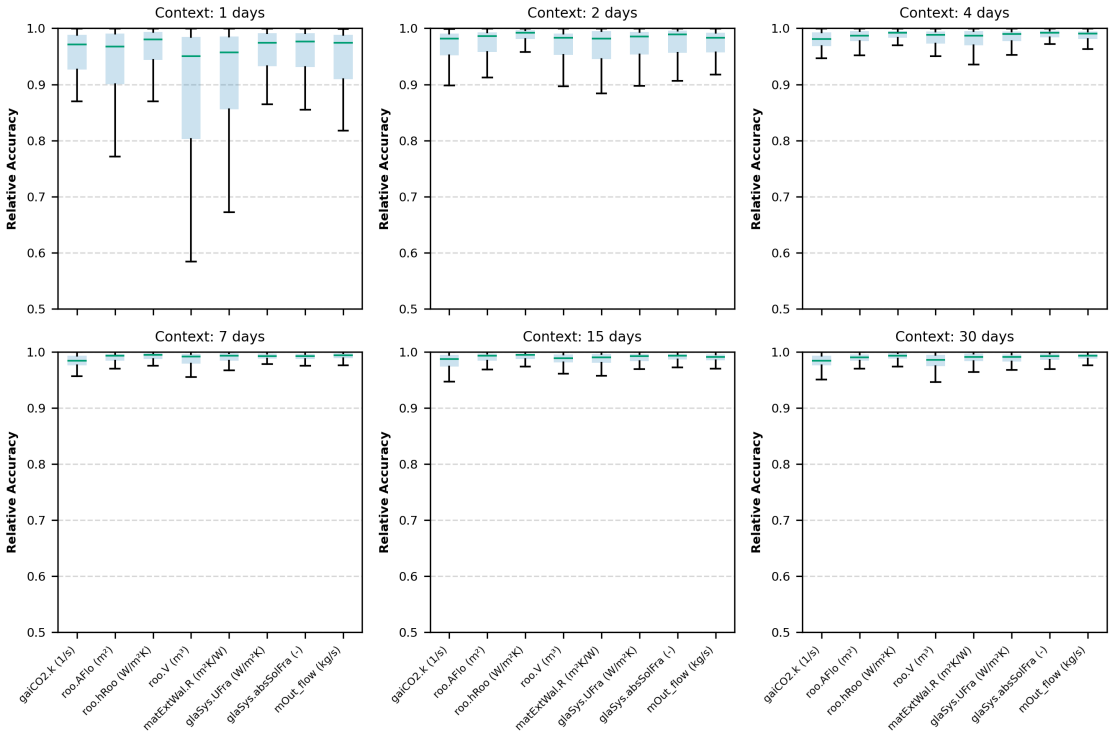


Figure 5. Boxplots of relative accuracy (3) versus context length.

ters (e.g., wall resistance `matExWal.R`, window U-factor `glsys.Ufrc`) benefit more from longer observation periods due to their slower thermal dynamics, indirect influence on measurable outputs, and entanglement with other physical properties. These factors lead to broader variance and reduced accuracy under short contexts. A 7-day context window offers a practical trade-off: it is short enough for deployment during commissioning while long enough to recover key physical parameters with high fidelity. This balance maximizes data efficiency and supports robust ini-

tialization of physics-based optimization routines in real buildings.

Table 3. Per-parameter R^2 on the test set for different transformer patch sizes for $N_{\text{win}} = 100$ parameter estimates.

Parameter	Patch 4	Patch 12	Patch 24
`gaiCO2.k`	0.9977	0.9947	0.9888
`roo.AFlo`	0.9989	0.9985	0.9975
`roo.hRoo`	0.9802	0.9979	0.9698
`roo.V`	0.9966	0.9982	0.9942
`matExtWal.R`	0.9850	0.9991	0.9913
`glaSys.UFra`	0.9950	0.9976	0.9931
`glaSys.absSolFra`	0.9976	0.9983	0.9949
`mOut_flow`	0.9997	0.9997	0.9997

In Table 3, we perform an ablation on the patch size of the proposed patch transformer; 1000 inputs are tested and the corresponding R^2 score is calculated. Overall, the network seems to be quite robust to patch-size selection. All three patch sizes deliver excellent predictive accuracy, with every parameter achieving $R^2 > 0.9$. The best performance in terms of average R^2 score across the paramaters is found with a patch size of 12, indicating it best balances capturing both fine-scale fluctuations and longer-range dependencies. Make the patch size smaller seems to improve estimates on more sensitive parameters such as `mOut_flow` and `gaiCO2.k` but shows poorer performance on the slower thermal dynamics (e.g. `matExtWal.R`). This trade-off suggests that very small patches lack sufficient context for slowly changing parameters, while very large patches dilute high-frequency signals. Note that a patch size of 12/24 corresponds to a 12h/24h window, which neatly aligns with the dominant diurnal cycles in building thermal and occupancy patterns. Smaller patches (4h) capture only very local fluctuations, leading to weaker performance. Consequently, a 12/24 patch size is recommended for uniformly high fidelity across all eight physical parameters.

4.3 Performance Comparison

We compare our patch transformer approach to two well-established time-series models: LSTM (Long Short-Term Memory) and TCN (Temporal Convolutional Network).

LSTM is a recurrent neural network variant designed to capture long-term dependencies in sequential data (Hochreiter and Schmidhuber 1997). It remains a common baseline for temporal modeling tasks due to its ability to handle variable-length inputs and learn from sequences with temporal correlations. We include LSTM as a representative of RNN-based architectures, which are widely adopted in forecasting and control settings.

In comparison, TCN is a convolutional alternative to RNNs that uses dilated causal convolutions to model temporal dependencies over long horizons (Lea et al. 2017). TCNs are attractive for their parallelizability, stable gradients, and competitive performance across time-series benchmarks. Their ability to capture hierarchical temporal structure without recurrence makes them suitable non-transformer baselines.

Table 4 summarizes the R^2 scores (mean and standard deviation) across models for varying context lengths. Overall, the Patch Transformer consistently outperforms both LSTM and TCN at short and medium horizons, validating its architectural design and inductive biases for learning from fine-grained temporal patterns.

Table 4. Performance comparison with TCN and LSTM.

Context	Model	Mean R^2	Stdev R^2
1 day	Our Transformer	0.8694	0.0818
	LSTM	-0.0284	0.4489
	TCN	0.0327	0.9633
3 days	Our Transformer	0.9679	0.0263
	LSTM	0.4266	0.2208
	TCN	0.4784	0.2382
7 days	Our Transformer	0.9669	0.0222
	LSTM	0.9664	0.0221
	TCN	0.6244	0.1410

At a short context length of 1 day, the transformer yields a mean R^2 of 0.8694, significantly outperforming LSTM and TCN, both of which under-perform with lower or even negative R^2 values. This highlights the transformer's superior capacity for modeling local dependencies via learned attention over patched segment: often, neither RNNs nor standard convolutions excel at this without larger receptive fields. At a medium context length (3 days), the performance gap improves, with our patch transformer achieving 0.9679, which is over double that of LSTM and TCN. This suggests that as the context window increases modestly, the transformer is uniquely capable of integrating more history without degradation, due to its global receptive field and ability to capture complex inter-dependencies. For a long context (7 days), both the transformer and LSTM perform similarly well, but the transformer's slightly higher mean and lower standard deviation suggest more stable generalization across samples. Notably, TCN remains behind, despite its design for long-range patterns, likely due to the difficulty of effectively tuning dilation rates and convolutional depths for diverse input dynamics.

Overall, our patch transformer-based regressor consistently outperforms or ties the strongest recurrent baseline across all tested window sizes. Its patch embeddings and self-attention mechanism enable more efficient contextual aggregation, particularly in low-data regimes (short windows), justifying its adoption for parameter regression tasks where context length may vary.

5 Conclusions

In general, the process of calibrating the parameters of physics-based models to experimental data often requires a large number of simulations to iteratively refine the parameters so that the model response matches the observations. Moreover, the number of simulations needed typically increases very quickly with the number of parameters being

DOI
10.3384/ecp218469

estimated, making this process quite time consuming. In comparison, this transformer-based approach enables the generation of parameter estimates directly from a set of experimental data in a zero-shot manner without the need for expensive iteration. This method uses offline simulation effectively in a massively parallel context during the training period to shift the burden of simulation from the time after data is collected to the time after the model is constructed. As engineers often desire information as soon as possible once experimental data is in hand, this is an attractive shift.

Furthermore, the proposed method shows strong promise as a warm-start initialization method for gradient-based parameter calibration algorithms when there is a significant simulation-to-real gap or further refinement of the values generated by the transformer-based method is desired. In many physical modeling settings, accurate initial guesses are critical for successful convergence of non-convex optimization routines. However, deriving such guesses analytically or heuristically is often infeasible. The transformer network helps fill this gap by rapidly providing good initial estimates that improve robustness and reduce computational overhead in subsequent calibration tasks.

We also demonstrate that this transformer-based architectures generalizes more effectively than conventional LSTM and TCN models when estimating system parameters from time-series data, particularly under conditions with limited temporal context. By leveraging the self-attention mechanism, the transformer can infer model parameters using significantly shorter time windows, thereby overcoming a longstanding limitation in traditional calibration tasks which typically require extensive input data for accurate estimation. This is especially beneficial for applications where measurement data is sparse or costly to collect.

As a promising direction for future work, we propose extending this approach to probabilistic transformers. This would allow the model to not only predict point estimates but also provide uncertainty quantification in the form of distributions over possible parameter values. Such capability could be instrumental in probabilistic calibration frameworks, for which prior distributions over parameters are required; e.g. Bayesian calibration.

References

Akiba, Takuya et al. (2019). “Optuna: A next-generation hyperparameter optimization framework”. In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631.

Balali, Yasaman et al. (2023). “Energy modelling and control of building heating and cooling systems with data-driven and hybrid models—A review”. In: *Renewable and Sustainable Energy Reviews* 183, p. 113496.

Blochwitz, Torsten et al. (2011). “The functional mockup interface for tool independent exchange of simulation models”. In: *Proceedings of the 8th international Modelica conference*. Linköping University Press, pp. 105–114.

Bres, Aurelien (2021). “Dimensionality reduction for calibration of dynamic building simulation models”. In: *Building Simulation 2021*. Vol. 17. IBPSA, pp. 1991–1998.

Cañas, Carlos Durán et al. (2023). “Parameter Estimation of Modelica Building Models Using CasADi”. In: *Modelica Conferences*, pp. 301–310.

Chakrabarty, Ankush et al. (2021). “Scalable Bayesian optimization for model calibration: Case study on coupled building and HVAC dynamics”. In: *Energy and Buildings* 253, p. 111460.

Cohen, Scott D, Alan C Hindmarsh, Paul F Dubois, et al. (1996). “CVODE, a stiff/nonstiff ODE solver in C”. In: *Computers in physics* 10.2, pp. 138–143.

Dassault Systèmes (2023). *Dymola*. Version 2024x. Dassault Systèmes. URL: https://www.3ds.com/products-services/catia/technologies/dymola/.

Glorot, Xavier and Yoshua Bengio (2010). “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, pp. 249–256.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). “Long short-term memory”. In: *Neural computation* 9.8, pp. 1735–1780.

Lea, Colin et al. (2017). “Temporal convolutional networks for action segmentation and detection”. In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 156–165.

Li, Yicheng et al. (2025). “A framework for calibrating and validating an HVAC system in Modelica”. In: *Journal of Building Performance Simulation*, pp. 1–21.

Nasrin, Shamma et al. (2022). “ENOS: Energy-aware network operator search in deep neural networks”. In: *IEEE Access* 10, pp. 81447–81457.

Ou, Yanglan et al. (2022). “Patcher: Patch transformers with mixture of experts for precise medical image segmentation”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, pp. 475–484.

Vanfretti, Luigi et al. (2016). “RaPId: A modular and extensible toolbox for parameter estimation of Modelica and FMI compliant models”. In: *SoftwareX* 5, pp. 144–149.

Vaswani, Ashish et al. (2017). “Attention is all you need”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS’17. Long Beach, California, USA: Curran Associates Inc., pp. 6000–6010. ISBN: 9781510860964.

Wetter, Michael et al. (2014). “Modelica Buildings library”. In: *Journal of Building Performance Simulation* 7.4, pp. 253–270.

Wüllhorst, Fabian et al. (2022). “AixCaliBuHA: Automated calibration of building and HVAC systems”. In: *Journal of Open Source Software* 7.72, p. 3861.

Zhan, Sicheng, Ankush Chakrabarty, et al. (2023). “A virtual testbed for robust and reproducible calibration of building energy simulation models”. In: *Building Simulation 2023*. Vol. 18. IBPSA, pp. 3491–3498.

Zhan, Sicheng, Gordon Wichern, et al. (2022). “Calibrating building simulation models using multi-source datasets and meta-learned Bayesian optimization”. In: *Energy and Buildings* 270, p. 112278.

Zhang, Yi and Lars Mikelsons (2023). “Sensitivity-guided iterative parameter identification and data generation with BayesFlow and PELS-VAE for model calibration”. In: *Advanced Modeling and Simulation in Engineering Sciences* 10.1, p. 9.

DOI
10.3384/ecp218469

Integration of Physical and AI Models Using Open and Interoperable Standards: A Model-Based Methodology for Autonomous Robot Development

Sebastian Rojas-Ordoñez[1, 2] Mikel Segura[1] Ekaitz Zulueta[2]

[1]IKERLAN, S. COOP, Spain, {jsrojas,msegura}@ikerlan.es
[2]EHU/UPV, Spain, jrojas015@ikasle.ehu.eus - ekaitz.zulueta@ehu.eus

Abstract

The development of Cyber-Physical Systems, particularly in the field of autonomous mobile robotics, often relies on proprietary environments, which limit flexibility and interoperability. This paper proposes a modular and open-source methodology that enables the modeling and simulation of such systems using non-proprietary tools. The methodology integrates Functional Mock-up Interface standard for model exchange, Open Neural Network Exchange standard for Convolutional Neuronal Networks algorithms integration, ROS2 as robotic middleware, and Gazebo as the simulation environment. To validate the approach, we applied it in the development of a mobile robot that navigates autonomously by following traffic signals. This implementation demonstrates that these open technologies can be effectively combined, overcoming common integration barriers among proprietary tools. The proposed workflow provides a practical alternative to proprietary solutions and demonstrates the feasibility of integrating open standards for the development of autonomous robotic systems. *Keywords: Autonomous robotic systems, FMI, ONNX, ROS2, Gazebo.*

1 Introduction

Model-Based Engineering (MBE) has transformed the way engineers and researchers design and develop complex systems (Holtmann, Liebel, and Steghöfer 2024). It is a strategy intended to reduce development time and to improve the quality of the system. To this end, the reuse of models and tests is promoted, thus avoiding iterative manual work and optimising the testing phase. Today, both industry and academia rely on MBE to predict system behavior, reduce the need of expensive physical prototypes, and drive advances in mobile robotics (Amorim 2019). At the same time, Convolutional Neural Networks (CNNs) are being increasingly applied in these systems, particularly for handling image processing tasks that support robot navigation and decision-making (Javaid et al. 2023).

Nowadays, many developments in robotics rely on commercial tools and proprietary platforms. While these tools offer powerful capabilities, their high cost and limited interoperability often restrict innovation (Haessler, Giones, and Brem 2023), increasing the demand for open-source alternatives that provide more flexible and accessible solutions.

In this context, the Modelica Association plays a key role in promoting and maintaining open standards for Cyber-Physical System (CPS) development, such as the Modelica Language for modeling dynamic systems and the Functional Mock-up Interface (FMI) (Modelica Association 2025) for ease the model exchange and co-simulation between different modeling and simulation (M&S) tools. These tools enable the development of complex systems, allowing for predictive modeling before physical prototyping, thus, reducing costs and minimizing development risks (Moshood et al. 2024). The integration of mathematical models with computer simulations has become a critical factor in ensuring the reliability and performance of modern systems (Idoko et al. 2024). However, applying these M&S capabilities to modern robotics frameworks is not exempt from challenges (Ray and Ramirez-Marquez 2020).

Regarding the evolution of robotics middleware, ROS2 emerged as a leading development framework for robotic applications. Particularly, ROS2 Humble (Open Robotics 2025b) offers enhanced features, including improved capabilities, robust security measures, and better support for multi-robot systems. When combined with advanced simulation environments like Gazebo (Open Robotics 2025a), it enables the validation of control algorithms and vision systems in realistic scenarios. However, the integration of Modelica-based models with these robotics frameworks remains challenging (Bardaro et al. 2017).

Additionally, the use of CNNs introduces a new set of challenges for systems based on traditional M&S approaches. As AI-based perception becomes increasingly relevant in CPSs, developers face the need to integrate machine learning and deep learning models into existing robotics and control workflows. The Open Neural Network Exchange (ONNX) (ONNX community 2025) standard addresses part of this challenge by offering a common, open format for representing and deploying trained neural networks across different platforms. However, the seamless integration of AI models in simulation tools,

robotic middleware, and control systems remains an area of active research. This highlights the need for flexible and modular development methodologies that can facilitate the integration of AI capabilities into CPSs (Parwej 2024).

In summary, our research addresses these integration challenges by proposing an open-source methodology that uses:

- FMI standard for exchanging validated physical models via FMUs.
- ONNX standard for exchanging AI-validated models for image processing.
- ROS2 Humble as robotics middleware.
- Gazebo for 3D simulation.

By focusing on the integration of these non-proprietary components, our methodology offers a cost-effective, modular, and interoperable solution for the development of CPS, focusing in autonomous mobile robotics. This approach not only ensures effective communication between control algorithms, vision processing, and physical simulation while laying the foundation for future improvements in adaptive and AI-driven robotic systems.

The paper is organized as follows: Section 2 reviews the background and related work; Section 3 details the methodology and its architecture; Section 4 describes the proof of concept used to implement and demonstrate the proposed methodology, specifically focusing on the integration of 3D simulation, control, and image processing components; Section 5 presents the results and their implications; and Section 6 concludes with final remarks and directions for future research.

2 Background and Related Work

The integration of simulation, control, and image processing has become essential in developing modern robotic systems (Nevliudov, Tsymbal, and Bronnikov 2021). Recent open-source tools and integration developments have enabled innovative approaches for designing and validating complex systems (Santos et al. 2024; Robles, Martín, and Díaz 2023). However, ensuring seamless integration between the different components is still difficult. This section examines key technologies and related work, highlighting current limitations and opportunities for improvement in simulation in MBE, open-source tools in robotics, model exchange using FMI, and the integration of AI models through ONNX in robotic applications.

2.1 Simulation in Model-Based Engineering

MBE utilizes models as central tools to understand, design, and manage complex systems. This methodology enables effective analysis, early detection of potential issues, and improved efficiency through clear, structured information exchange (Holtmann, Liebel, and Steghöfer 2024). MBE supports critical tasks such as capturing requirements, system design, simulation-based validation, code generation, and result verification, focusing on specific system components to optimize key areas and mitigate development risks (Khandoker et al. 2022). For instance, (Sekar and Baras 2022) describe a model-based framework for robotic grasping, demonstrating how MBE facilitates iterative improvements and early error detection, ultimately reducing prototyping costs and risks.

Recent implementations, such as the one presented by Matlab/Simulink (MathWorks 2024) that combine ROS2 with control systems, show the potential and limitations of current approaches. In this use case, the system employs Simulink models to control simulated robots and manage data exchange, including camera feed, odometry, and speed commands; the focus of this use case involves an Image Processing subsystem for sign detection and a Sign Tracking Logic subsystem, implemented with a Stateflow chart, that translates detected sign information into linear and angular velocity commands.

While these implementations demonstrate MBE's capability for robust, iterative design, it is necessary to highlight the need for more flexible, platform-independent solutions (Corallo et al. 2022).

2.2 Open-Source Tools in Robotics

Open-source platforms have transformed robotics development by providing alternatives to traditional proprietary solutions. Authors of (Patel, Liarokapis, and Dollar 2022) discuss this transformation through open robotic hardware, documenting significant advances and diverse implementation practices across a range of applications. Similarly, (Soori, Arezoo, and Karimi Ghaleh Jough 2024) demonstrates how frameworks such as ROS2 accelerate industrial integration, particularly in Industry 4.0.

These advances in hardware and systems framework have prepared the way for the evolution of complementary software tools that further streamline robotics development. Market analyses by (Cornille 2024) confirm that including visualization tools like Rviz and Rerun.io, alongside physics simulators such as Gazebo and MuJuCo, it is possible to reduce development cycles and improve system reliability.

2.3 Model Exchange Using FMI

The FMI is an open standard that enables model exchange and co-simulation across different M&S environments, facilitating the integration of components from several domains, such as mechanical, electrical, and control systems. This capability is crucial for the development of complex systems where multiple M&S tools may work together. It defines a format for packaging such models, called Functional Mock-up Units (FMU). A FMU represents the practical implementation of the FMI standard, thus, it is a software component that complies with the FMI standard, making it portable and reusable across compatible simulation environments.

DOI
10.3384/ecp218479

Multiple studies have demonstrated the effectiveness of the FMI standard in practical applications and system simulations. For instance, (Oudart et al. 2020) present a model-based toolchain that leverages the FMI standard for CPS co-simulation. Their approach effectively separates physical and logical models, achieving up to 40% reduction in development time. Additionally, (Shahsavari et al. 2021) introduce MCX, an open-source framework for digital twins, highlighting the capability of FMUs to encapsulate system dynamics in a reproducible manner.

Building on these results, integrating FMI with robotics frameworks has emerged as a natural next step to bridge the gap between simulation and control. One notable contribution in this direction is the `fmi_adapter` package (Lange et al. 2020), which provides a concrete implementation for connecting FMI with ROS and Gazebo. This tool enables two primary approaches for incorporating FMUs into ROS-based architectures:

1. **Node-Based Use:** As shown in Figure 1, this approach represents how to create a ROS node containing an FMU. The architecture shows subscribers (left side) connected to the FMI adapter (middle, blue), which interfaces with the FMI library to communicate with the Application FMU (right, green). This implementation is helpful to create the necessary ROS subscribers and publishers to handle all model variables.

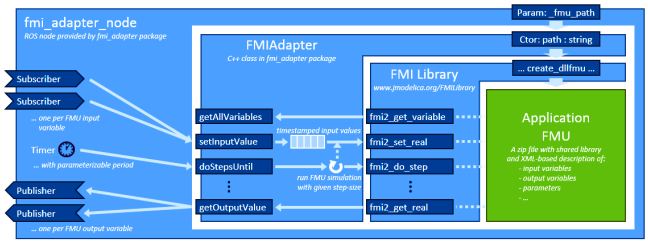


Figure 1. Node-Based Use
(Lange et al. 2020)

2. **Library-Based Use:** Figure 2 illustrates an alternative implementation using a C++ class (`fmi_adapter::FMIAdapter`). In this approach, the application node (left, green) directly interfaces with the FMI adapter components (blue). This method provides more direct control over the FMU lifecycle by eliminating the ROS node wrapper layer, allowing for more precise timing control and custom integration patterns.

Additionally, for Gazebo integration, (Lange et al. 2020) presented `gazebo-fmi` plugins, that enable the incorporation of FMU within the simulation environment. For co-simulation scenarios, `gazebo-fmi` components follow a master-slave architecture where one component, acting as the master, coordinates the execution of other components, which act as slaves. In this context, the main

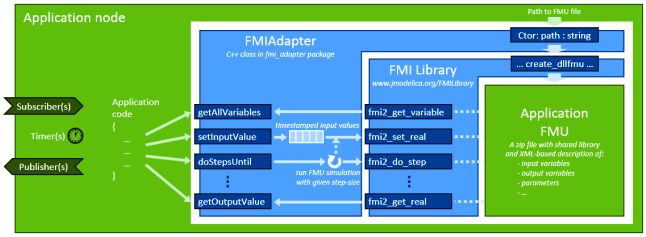


Figure 2. Library-Based Use
(Lange et al. 2020)

physics engine of Gazebo, acting as master, coordinates the overall simulation timing, while individual FMUs, acting as slaves, perform their specific calculations using their built-in solvers. The `gazebo-fmi` suite includes two key plugins:

1. **gazebo-fmi-actuator plugin:** For enabling the simulation of complex actuator dynamics such as reflected inertias, velocity/torque curves, and series elastic actuators.

2. **gazebo-fmi-single-body-fluid-dynamics plugin:** Designed to simulate fluid-body interactions, this plugin uses FMUs to compute lift, drag, and other fluid forces on a body.

2.4 AI Integration Using ONNX

Integrating artificial intelligence into robotic control systems presents significant opportunities in modern robotics, such as enabling adaptive decision-making, real-time optimization, and complex task automation. However, this integration also introduces technological challenges, including platform fragmentation, implementation complexity, and the need for reliable interoperability between AI models and robotic hardware (Patrício, Varela, and Silveira 2024). ONNX has emerged as a widely accepted standard for deploying AI models across various platforms. Its use enables standardized implementation of deep learning models, reducing dependence on customized and proprietary solutions. For instance, (El-Hussieny 2024) demonstrate a deep learning-based model predictive control strategy for a 3-DOF biped robot leg using ONNX, achieving real-time performance. Similarly, (Yue 2024) propose a real-time dynamic motion compensation design in vision-guided robots, that leverages ONNX to enhance task precision and system reliability, highlighting how standardized AI model deployment can improve task precision.

However, current approaches to integrating AI models with robotic control systems often rely on proprietary solutions or require complex custom implementations. This is particularly evident in applications that combine perception with dynamic control, where maintaining consistent performance across multiple subsystems can be difficult (Xu et al. 2024). Indeed, the robotics community has identified the need for more standardized, open approaches to AI integration that can reduce implementation

complexity while maintaining system reliability (Licardo, Domjan, and Orehovački 2024).

3 Proposed Methodology

To address the challenges outlined in Section 2, this paper proposes a development methodology for robotic systems based on open-source tools and following MBE principles. Our approach focuses on integrating physical and control models using FMUs, and AI algorithms via ONNX models, all within a Linux-based environment. ROS2 is proposed as the communication middleware, and Gazebo as the simulation platform. In this way, we aim to facilitate the development of such a complex systems in a cost-effective, modular, and interoperable manner.

It is important to note that our methodology assumes these models are already developed and available; therefore, the focus is on their integration into a robotic framework. This integration applies both to simulated environments and to deployment on physical hardware. The proposed methodology is guided by the following key principles:

Interoperability and Open-Source: The use of open-source tools and standards enables the development of a modular and scalable system architecture. This approach eliminates dependency on proprietary environments and promotes collaboration between robotic system developers.

Unified Integration of Physical and AI Models: This proposal unifies the integration of FMUs, containing physical models or control algorithms, with ONNX files, containing machine learning or deep learning algorithms. The integration is performed through ROS2, this way we obtain several advantages:

- As many robotic developments incorporate ROS2, integrating these components does not require modifying previously developed systems. Instead, adding new functionalities simply requires incorporating additional components.
- It reduces development time and costs, as predeveloped models can be incorporated into the system without the need to rewrite code. In other words, it acts as a code generation mechanism, with FMU and ONNX serving as the final code elements.
- It provides flexibility, allowing individual modules to be updated or replaced without affecting the entire system.

Modular Architecture: Our design adopts a modular appoach, where independent subsystems are wrapped within ROS2 nodes. ROS2 uses a communication model based on the publisher-subscriber pattern, where nodes can publish messages to topics or subscribe to them to receive information. This communication is built on top of Data Distribution Service (DDS), a middleware that enables automatic discovery of nodes and message exchange without requiring nodes to know each other directly. Gazebo provides the virtual environment for system simulation. This modular architecture reduces the complexity of integrating diverse technologies. As a result, our proposal is easily adaptable to a wide range of robotic applications.

4 Proof of Concept

To validate the effectiveness of the proposed methodology, we applied it to a scenario in which a robot navigates from point A to point B by following traffic signals that guide its path. We obtained this use case from a Simulink example originally designed to validate communication between Gazebo and Simulink. Our purpose is to reuse some of its elements to demonstrate how FMUs and ONNX models can be integrated for the development of CPS, showing that the same components and tools can be used both in virtual testing environments and in real hardware deployment.

As our approach does not involve the development of the models themselves, our initial intention was to reuse the odometry and signal-tracking logic by exporting them as FMUs, as well as exporting the image processing module as an ONNX model. However, we encountered some difficulties that required some adjustments. First, we were unable to generate FMUs directly from Simulink, since our development environment was Windows-based, while the final application is intended to run on Linux. FMUs generated on Windows cannot be seamlessly integrated into a Linux environment. To address this, we replicated these modules in OpenModelica under Linux and generated the FMUs from there.

Regarding image processing, Simulink relies on a proprietary toolbox that cannot be exported. To overcome this limitation, we recreated the functionality using a Python implementation, using PyTorch to enable ONNX export.

On the other hand, we were able to reuse the original Gazebo scenario. Therefore, our validation strategy is to test whether the robot behaves in the same way in both the original Simulink setup and our integration, now relying exclusively on non proprietary tools. This way, in this proof of concept, ROS 2 orchestrates the system, integrating FMUs for control and odometry tasks, and an ONNX model for image processing, with Gazebo serving as the simulation platform. The following subsections outline the processes involved in applying the proposed methodology.

4.1 Application Scenario

Figure 3 shows the simulated environment in Gazebo, where the map and layout of the simulation area can be appreciated. In this scenario, the mission of the mobile robot is to navigate from point A to point B while following visual signals. The yellow dashed line represents the expected path that the robot should follow based on the traffic signs positioned throughout the environment.

DOI
10.3384/ecp218479

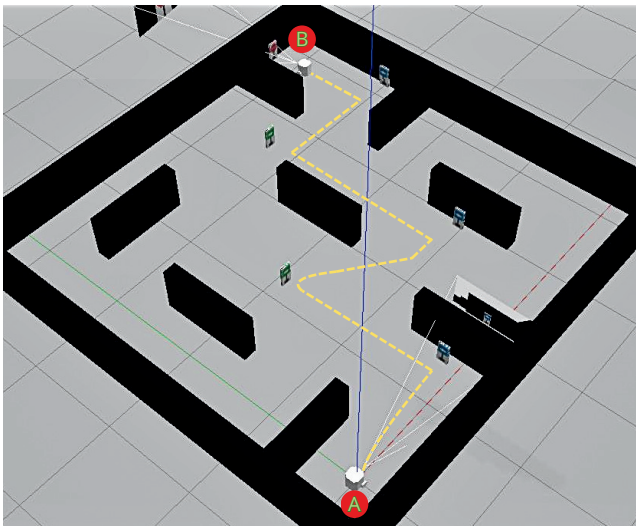


Figure 3. Simulated Gazebo environment showing the navigation path with traffic signs to guide the navigation of the robot.

The robot's navigation is guided by a set of traffic signals strategically placed throughout the environment, as shown in Figure 4. These signals provide the instructions for the robot's navigation behavior:

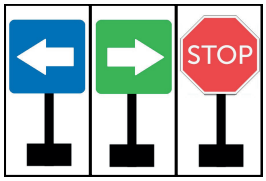


Figure 4. Traffic signals for robot navigation: Left arrow (blue) robot to turn left, right arrow (green) robot to turn right, and stop sign (red) robot to Stop.

4.2 Robot Pose Estimation (Odometry) FMU

The odometry management is handled via a two-stage process:

1. **Odometry Processing:** The system subscribes to the `/odom` topic in ROS2 to receive position and orientation data from the simulator. As shown in Figure 5, the FMU block `ReadOdometry` processes six input parameters: quaternion orientation (q_w, q_x, q_y, q_z) and cartesian position (x, y). The position coordinates (x, y) are directly mapped to the outputs ($posX$, $posY$), while the quaternion components are transformed into a 2D orientation angle ($theta$) using Euler angle conversion. This results in three output parameters ($posX$, $posY$, $theta$) that completely describe the robot's pose in 2D space.

2. **ROS2 Integration:** After understanding the functionality of the odometry FMU, we integrated it into a ROS2 node. This integration was performed by creating a node named (`fmu_odometry_node`) that subscribes to the robot's default `/odom` topic.

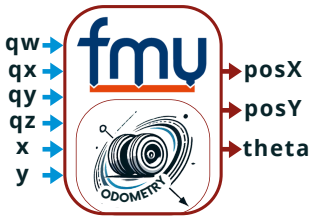


Figure 5. I/O port configuration of the odometry FMU block showing six inputs (four quaternion components and two position coordinates) and three outputs (2D position and orientation angle).

The node processes the odometry data and publishes the resulting pose information to a custom topic named `/fmu_odom_out`, as illustrated in Figure 6 the integration of the system is verified using rqt a ROS2's native introspection and visualization tool.



Figure 6. ROS2 node structure for the odometry FMU input subscription to `/odom` and output publication to `/fmu_odom_out`.

4.3 Traffic Sign Detection ONNX

The image processing subsystem employs a two-stage process:

1. **Vision System:** The ONNX model shown in Figure 7, processes RGB images with dimensions of 480×640×3 pixels received from the `/camera/image_raw` topic. The model takes this input tensor [1 480 640 3] and outputs two key parameters: `blobX` for the sign's horizontal position and `blobSize` for its dimensions.

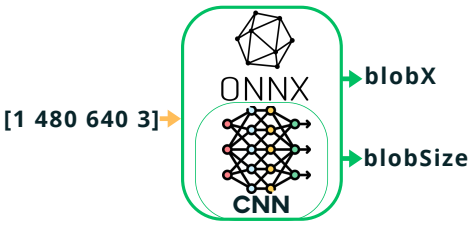


Figure 7. ONNX model structure showing the CNN architecture with its input tensor [1 480 640 3] (representing batch size, height, width, and RGB channels) and its two output parameters: `blobX` and `blobSize`.

2. **ROS2 Integration:** After understanding the functionality of the ONNX model, we integrated it into a ROS2 node. This integration was performed by creating a node that processes the camera input data and publishes the processed information through two custom topics: `/onnx_blobSize`, which provides

continuous updates of the traffic sign dimensions, and `/onnx_blobX`, which delivers the sign's horizontal position in the frame, as illustrated in Figure 8 the integration of the system is verified using rqt a ROS2's native introspection and visualization tool.



Figure 8. ROS2 node architecture for ONNX integration, illustrating input from the camera feed and output topics for sign detection parameters.

4.4 Autonomous Navigation Control FMU

The control system integrates the data from the FMU odometry model and the ONNX model subsystems through an FMU that manages the robot's control logic:

1. **Controller Design:** The SignTrackingLogic FMU incorporates decision-making logic to process visual inputs from the ONNX node and pose information from the odometry FMU. This FMU calculates the relative position of detected traffic signs and generates corresponding movement commands. Figure 9 shows the I/O configuration, where inputs include `blobSize`, `blobX`, and `pose` (i.e., *posX*, *posY*, *theta*), and outputs include linear velocity (`v`), angular velocity (`w`), and a stop command (`stopRobot`).

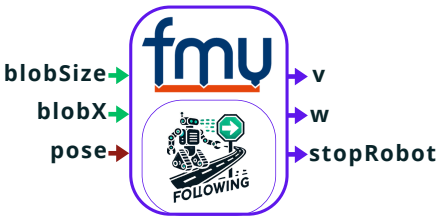


Figure 9. I/O port configuration of the SignTrackingLogic FMU block.

2. **ROS2 Integration:** After understanding the functionality of the SignTrackingLogic FMU, we integrated it into a ROS2 node. This integration was performed by creating a node named (`fmu_sign_tracking_node`) that subscribes to three topics: `/onnx_blobSize`, `/onnx_blobX`, and `/fmu_odom_out`. The node processes this information and publishes the resulting control commands to the `/cmd_vel` topic, as illustrated in Figure 10 the integration of the system is verified using rqt a ROS2's native introspection and visualization tool.

4.5 Integration of FMUs and ONNX

The final integration phase unifies the FMU-based control components and the ONNX-based image processing module within ROS2. This is accomplished through a layered architecture that ensures communication and synchronization between nodes while maintaining system modularity and scalability. The overall integration is validated using Gazebo as the simulation environment.



Figure 10. ROS2 node architecture for the SignTrackingLogic FMU, showing input subscriptions from image processing and odometry nodes, and output publication to robot control.

4.5.1 System Architecture in Linux

Figure 11 illustrates the complete system architecture, which comprises FMU Nodes for odometry processing and sign tracking logic, an ONNX Node for image processing and Gazebo Plugins for simulation interfacing through ROS2 as middleware.

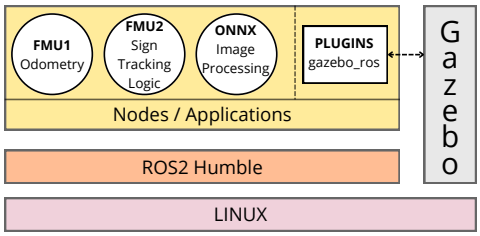


Figure 11. System architecture in a Linux environment.

4.5.2 Integration Implementation

The integrated system is deployed as follows:

1. **Component Deployment:** FMUs are executed within dedicated ROS2 nodes, the ONNX model is deployed within a computer vision node, and Gazebo interfaces with these components via the ROS2 bridge.

2. **Data Flow Management:** As shown in Figure 12, the system maintains synchronized communication through topic-based message passing between nodes, enabling coordinated execution of FMUs and ONNX models.

This integration setup shows the practical feasibility of combining ROS2, FMUs, ONNX models, and Gazebo within a Linux-based open-source environment. The resulting system supports modular development, clear separation of concerns, and synchronized communication across components, aligning with the principles outlined in our proposed methodology.

DOI
10.3384/ecp218479

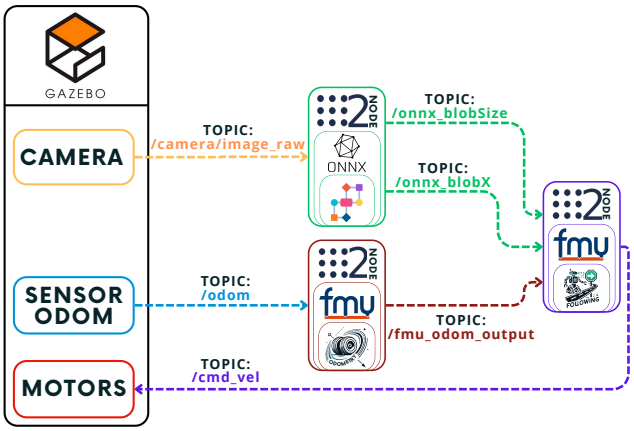


Figure 12. Data flow diagram inter-node communication.

5 Results and Discussion

This section presents and analyzes the results from our implementation, evaluating both the performance of individual components and the overall system integration. In addition to the standard performance metrics, we compare the original outputs of the FMU and ONNX models with those obtained after integration within the ROS2 nodes, thereby validating that our integration preserves the model characteristics and enhances system's interoperability.

5.1 Odometry FMU Performance Analysis

To validate the accuracy of our solution, we compared the roll angle output from the original model with that of our integrated FMU-ROS2 implementation. The roll angle was chosen because it requires quaternion-to-Euler conversion, while x and y coordinates pass directly through the system without transformation. This comparison is essential for verifying that the mathematical operations and coordinate transformations are preserved after integration. Figure 13 shows the comparison of roll angle signals.

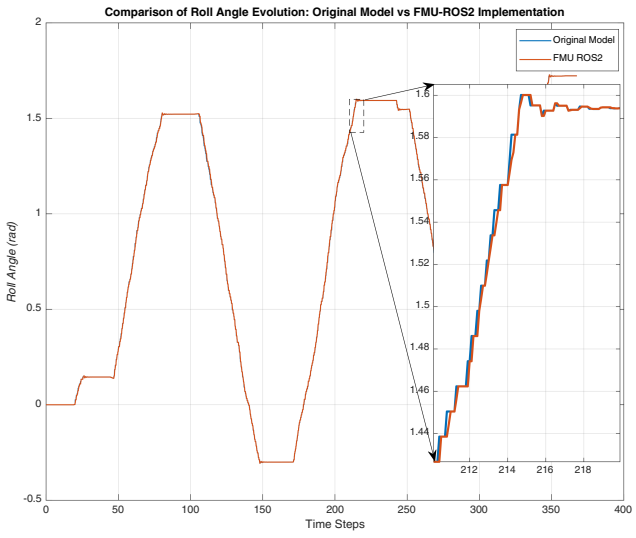


Figure 13. Comparison of roll angle evolution between the original model and FMU-ROS2 implementation.

From this comparison, we can conclude that the quaternion-to-Euler conversion accuracy is maintained. The signals exhibit slight variations, specifically on the order of 0.002 rad. These variations are acceptable, as the operational range goes from -0.4 rad to 1.8 rad. They are likely caused by differences in floating-point computations or by the overhead introduced during ROS 2 message passing.

5.2 ONNX Vision Processing Performance

To validate the accuracy of our ONNX implementation for sign detection, we compared the blobSize(3) output of the original ONNX model with our ONNX-ROS2 solution. This output was chosen as a representative measure since similar detection patterns are observed in other components of the blobSize and blobX vectors. Figure 14 shows the blob size signals.

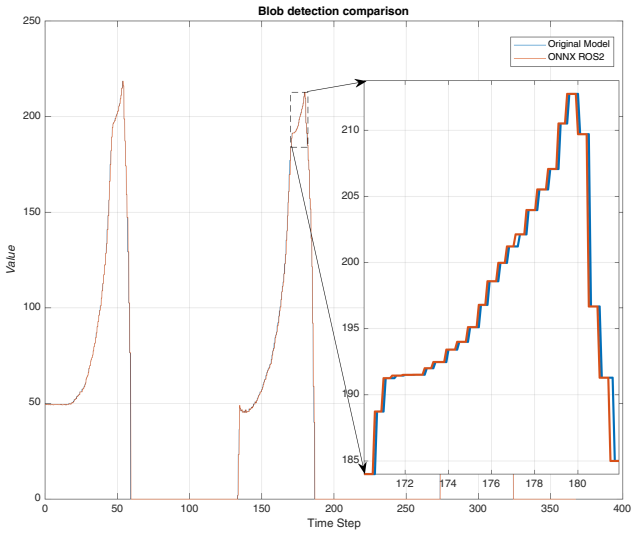


Figure 14. Comparison of blob detection between the original model and the ONNX-ROS2 implementation.

The comparison reveals that the detection accuracy is preserved. Similar to previous comparison, the signals also exhibit slight variations, specifically on the order of 1 pixel. These variations are acceptable, as the operational range goes from 0 to 215 pixels. They are likely caused by differences in floating-point computations or by the overhead introduced during ROS 2 message passing.

5.2.1 ROS2-Based System Integration

The overall system integration was verified using rqt, ROS2's native introspection and visualization tool. Figure 15 displays the runtime communication graph generated by rqt, validating the proposed system architecture as depicted in Figure 12.

The comparison between the proposed architecture and the actual implementation shows successful architecture validation, including the establishment of all intended communication pathways, correct node relationships, and proper topic naming and message routing. In terms of sys-

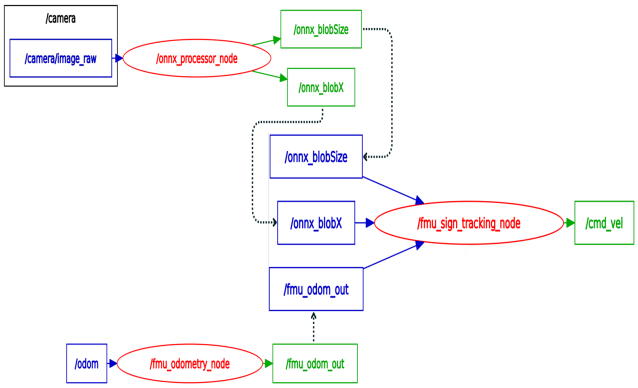


Figure 15. Runtime communication graph generated by rqt, showing node interactions and topic connections in the ROS2 environment.

tem integration, the data flow between the ONNX vision processing and FMU nodes is effective, and synchronization between odometry and control components is maintained, ensuring reliable message passing across the system.

5.2.2 Trajectory Analysis and Navigation Performance

To evaluate the effectiveness of the integrated system, we first analyzed the robot's trajectory through its X-Y position data. Figure 16 shows the complete path, where we can identify the navigation phases.

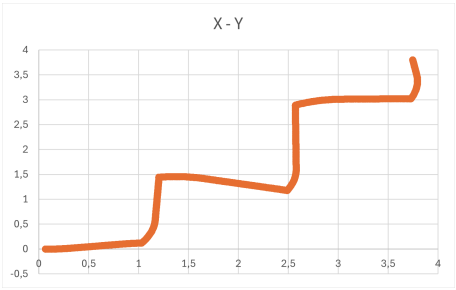


Figure 16. X-Y position plot showing the robot's trajectory.

When this recorded trajectory (orange line) is overlaid onto the simulation environment, as shown in Figure 17, we can verify the robot's successful navigation from point A to point B while following the traffic signs.

The comparison between the expected path (yellow dashed line in Figure 3) and the actual trajectory demonstrates that the robot successfully executed all required turns and maintained proper alignment during straight segments. The transitions observed in both representations validate the effectiveness of the SignTrackingLogic FMU in interpreting visual signals and generating appropriate control commands.

6 Conclusions and Future Work

In this work, we propose a cost-effective, modular, and interoperable methodology for developing CPSs using FMI

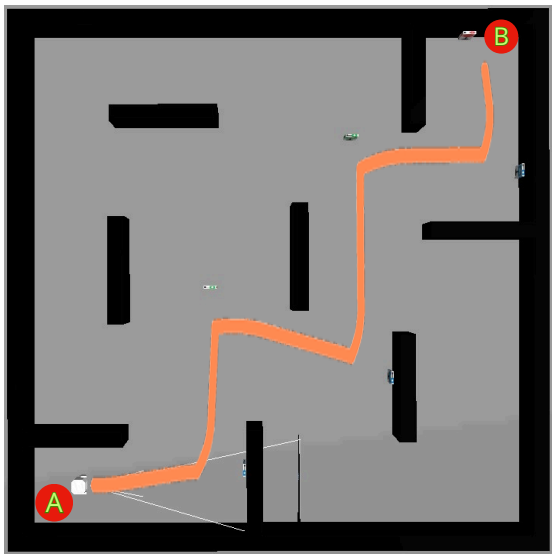


Figure 17. Actual trajectory (orange line) overlaid on the simulation environment, demonstrating precise path following and successful navigation through the course.

2.0 standard. We tested our solution with an autonomous robot navigation use case, integrating FMUs and ONNX models to establish the robot's behavior. The experimental results demonstrate that the system maintains precision while ensuring component independence. Specifically, our tests show accurate odometry tracking, reliable vision-based detection, and control behavior evidenced by consistent velocity patterns and effective trajectory adjustments.

The modular design, which wrappers FMUs and ONNX models as independent ROS2 nodes, offers significant benefits in terms of scalability, maintainability, and interoperability. The implementation demonstrates robust performance in both simulation and real-world environments, with the FMU-based control loop and ONNX inference running effectively within the computational constraints of the embedded platform. The ROS2 middleware proves efficient in handling the communication between components, maintaining system stability and responsiveness throughout operation.

The successful transition from simulation to physical implementation validates the practical applicability of our approach. As shown in Figure 18, while the real environment didn't exactly match the virtual scenario's measurements, the system demonstrated robust adaptability. This flexibility stems from the algorithm's fundamental reliance on traffic sign recognition and following, enabling effective operation in any environment where the standardized signs are present.

By relying on open and interoperable standards, our approach allows easy module replacement and simplifies system updates. This openness not only removes dependencies on proprietary tools but also fosters collaboration among robotic system developers. Moreover, our methodology significantly reduces development time and costs by

DOI
10.3384/ecp218479

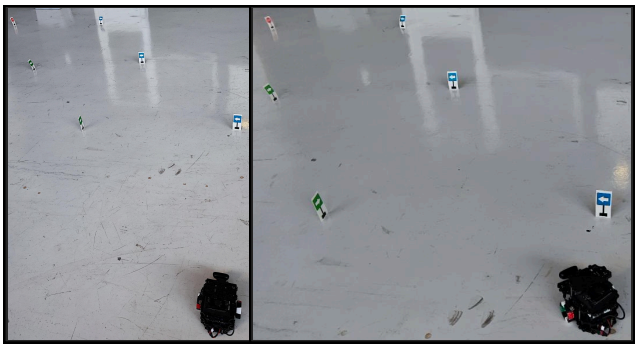


Figure 18. Implementation in physical robot demonstrating system adaptability in real-world conditions

allowing seamless code reuse between the simulation environment and the final hardware deployment. The same ROS2-based modules used for testing and validation in simulation can be directly integrated into the target robotic platform, eliminating the need for reimplementation.

Regarding future research, we identified the following aspects:

- **Advanced AI algorithms:** Development of efficient convolutional neural networks through dynamic neural architecture search and adaptive pruning techniques, focusing on minimizing computational overhead while preserving detection accuracy in vision-based navigation tasks.
- **Use of Zenoh as communication middleware:** Integration of the Zenoh protocol to enhance real-time data distribution in distributed CPS deployments. Zenoh's inherent support for time-sensitive networking and decentralized pub/sub patterns could reduce end-to-end latency while improving interoperability across heterogeneous hardware.

Overall, the current results suggest that our architecture is well-suited to accommodate such extensions while maintaining its core reliability and performance. This work opens new paths for integrating simulation, control, and AI-driven perception, contributing to the advancement of autonomous robotic systems.

References

Amorim, Tiago (2019-01). "Strategies and Best Practices for Model-Based Systems Engineering Adoption in Embedded Systems Industry". In: *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. URL: https://www.academia.edu/103032710/Strategies_and_Best_Practices_for_Model_Based_Systems_Engineering_Adoption_in_Embedded_Systems_Industry (visited on 2025-04-04).

Bardaro, Gianluca et al. (2017-07). "Using Modelica for advanced Multi-Body modelling in 3D graphical robotic simulators". en. In: pp. 887–894. DOI: 10.3384/ecp17132887. (Visited on 2025-04-13).

Corallo, Angelo et al. (2022). "Evaluating the Maturity of MBE Application". en. In: *Product Lifecycle Management. Green and Blue Technologies to Support Smart and Sustainable Organizations*. Ed. by Osiris Canciglieri Junior et al. Cham: Springer International Publishing, pp. 401–415. ISBN: 978-3-030-94335-6. DOI: 10.1007/978-3-030-94335-6_29.

Cornille, Tobias (2024). *Software Tools For Robotics Landscape (2024)*. Segments.ai. Accessed: Mar. 04, 2025. URL: https://segments.ai/blog/software-tools-for-robotics-landscape/.

Haessler, Philipp, Ferran Giones, and Alexander Brem (2023-03). "The who and how of commercializing emerging technologies: A technology-focused review". In: *Technovation* 121, p. 102637. ISSN: 0166-4972. DOI: 10.1016/j.technovation.2022.102637. (Visited on 2025-04-04).

Holtmann, Jörg, Grischa Liebel, and Jan-Philipp Steghöfer (2024-04). "Processes, methods, and tools in model-based engineering—A qualitative multiple-case study". In: *Journal of Systems and Software* 210, p. 111943. ISSN: 0164-1212. DOI: 10.1016/j.jss.2023.111943. (Visited on 2025-04-01).

El-Hussieny, H. (2024-07). "Real-time deep learning-based model predictive control of a 3-DOF biped robot leg". In: *Scientific Reports* 14.1, p. 16243. DOI: 10.1038/s41598-024-66104-y.

Idoko et al. (2024-11). "Mathematical modeling and simulations using software like MATLAB, COMSOL and Python". In: *Magna Scientia Advanced Research and Reviews* 12, pp. 62–095. DOI: 10.30574/msarr.2024.12.2.0181.

Javaid, Mohd et al. (2023). "An integrated outlook of Cyber–Physical Systems for Industry 4.0: Topical practices, architecture, and applications". In: *Green Technologies and Sustainability* 1.1, p. 100001. ISSN: 2949-7361. DOI: https://doi.org/10.1016/j.grets.2022.100001.

Khandoker, Azad et al. (2022). "Towards a logical framework for ideal MBSE tool selection based on discipline specific requirements". In: *Journal of Systems and Software* 189, p. 111306. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2022.111306.

Lange, Ralph et al. (2020-08). "Integrating the Functional Mock-Up Interface with ROS and Gazebo". In: pp. 187–231. ISBN: 978-3-030-45955-0. DOI: 10.1007/978-3-030-45956-7_7.

Licardo, Josip Tomo, Mihael Domjan, and Tihomir Orehovački (2024). "Intelligent Robotics—A Systematic Review of Emerging Technologies and Trends". In: *Electronics* 13.3. ISSN: 2079-9292. DOI: 10.3390/electronics13030542.

MathWorks (2024). *Sign-Following Robot with ROS 2 in Simulink*. URL: https://es.mathworks.com/help/ros/ug/sign-following-robot-using-ros2-simulink.html (visited on 2025-03-03).

Modelica Association (2025). *Functional Mock-up Interface*. Accessed: Mar. 07, 2025. URL: https://fmi-standard.org/.

Moshood, Taofeeq D. et al. (2024). "Infrastructure digital twin technology: A new paradigm for future construction industry". In: *Technology in Society* 77, p. 102519. ISSN: 0160-791X. DOI: https://doi.org/10.1016/j.techsoc.2024.102519.

Nevliudov, Igor, Oleksandr Tsymbal, and Artem Bronnikov (2021-12). "IMPROVEMENT OF ROBOTIC SYSTEMS BASED ON VISUAL CONTROL". In: *Innovative Technologies and Scientific Solutions for Industries*, pp. 65–74. DOI: 10.30837/ITSSI.2021.18.065.

ONNX community (2025). *Open Neural Network Exchange (ONNX)*. [Accessed 20-03-2025]. URL: https://onnx.ai/.

Open Robotics (2025a). *Gazebo*. Accessed: Mar. 07, 2025. URL: https://gazebosim.org/home.

Open Robotics (2025b). *ROS2 Humble*. URL: https://docs.ros.org/en/humble/index.html (visited on 2025-03-25).

Oudart, D. et al. (2020). "A Model based Toolchain for the Cosimulation of Cyber-physical Systems with FMI". In: *Proceedings of the 8th International Conference on Model-Driven Engineering and Software Development*. Valletta, Malta, pp. 15–25. DOI: 10.5220/0008875400150025.

Parwej, Firoj (2024-05). "The Implementation of NGCPS to Integrate AI in Order to Get the Accurate Decision Making". In: *2024 4th International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, pp. 1300–1305. DOI: 10.1109/ICACITE60783.2024.10616671. (Visited on 2025-04-13).

Patel, Vatsal, Minas Liarokapis, and Aaron Dollar (2022-01). "Open Robot Hardware: Progress, Benefits, Challenges, and Best Practices". In: *IEEE Robotics and Automation Magazine* PP, pp. 2–29. DOI: 10.1109/MRA.2022.3225725.

Patrício, Leonel, Leonilde Varela, and Zilda Silveira (2024). "Integration of Artificial Intelligence and Robotic Process Automation: Literature Review and Proposal for a Sustainable Model". In: *Applied Sciences* 14.21. ISSN: 2076-3417. DOI: 10.3390/app14219648.

Ray, Douglas and Jose Ramirez-Marquez (2020-01). "A framework for probabilistic model-based engineering and data synthesis". In: *Reliability Engineering & System Safety* 193, p. 106679. ISSN: 0951-8320. DOI: 10.1016/j.ress.2019.106679. (Visited on 2025-04-13).

Robles, Julia, Cristian Martín, and Manuel Díaz (2023). "OpenTwins: An open-source framework for the development of next-gen compositional digital twins". In: *Computers in Industry* 152, p. 104007. ISSN: 0166-3615. DOI: https://doi.org/10.1016/j.compind.2023.104007.

Santos, Adriano et al. (2024-11). "Integration of Artificial Vision and Image Processing into a Pick and Place Collaborative Robotic System". In: *Journal of Intelligent and Robotic Systems* 110. DOI: 10.1007/s10846-024-02195-z.

Sekar, P. K. M. and J. S. Baras (2022-07). "Model-Based Systems Engineering Simulation Framework for Robot Grasping". In: *INCOSE International Symposium*. Vol. 32. S2, pp. 82–89. DOI: 10.1002/iis2.12898.

Shahsavari, Sajad et al. (2021-06). *MCX An Open-Source Framework For Digital Twins*. DOI: 10.7148/2021-0119.

Soori, Mohsen, Behrooz Arezoo, and Fooad Karimi Ghaleh Jough (2024-01). "Intelligent Robotic Systems in Industry 4.0, A Review". In: *Journal of Advanced Manufacturing Science and Technology*, p. 2023. DOI: 10.51393/j.jamst.2024007.

Xu, Haowen et al. (2024-10). "Leveraging generative AI for urban digital twins: a scoping review on the autonomous generation of urban data, scenarios, designs, and 3D city models for smart city advancement". In: *Urban Informatics* 3.1, p. 29. ISSN: 2731-6963. DOI: 10.1007/s44212-024-00060-w.

Yue, L. (2024-08). *Design of Real-Time Dynamic Motion Compensation in Vision-Guided Robots for Biopsy Needle Alignment*. URL: https://essay.utwente.nl/103243/1/LI_MA_EEMCS.pdf.

Hybrid reinforcement learning enhanced trajectory planning for an ADR scenario for combined control of a satellite with a 7-axis robotic arm using Modelica/FMI

Matthias J. Reiner[1]

[1]DLR, Institute of Robotics and Mechatronics, Oberpfaffenhofen, Germany, Matthias.Reiner@dlr.de

Abstract

This work presents a novel hybrid trajectory planning algorithm that leverages reinforcement learning (RL) to address the challenges of active debris removal (ADR) in space, specifically for the combined control of a satellite and its 7-axis robotic arm. The proposed approach integrates a lightweight RL policy with correction algorithms and classical trajectory planning, enabling robust and collision-free maneuvering of the chaser satellite and precise placement of the robotic gripper near the target grasp point. This hybrid method is designed to handle uncertainties in target dynamics and sensor measurements, while remaining feasible for implementation on space-grade hardware. The effectiveness of the algorithm is demonstrated through comprehensive simulation studies using Modelica/FMI, validating its capability to generate safe and reliable trajectories in complex ADR scenarios.

Keywords: Reinforcement Learning, Trajectory Planning, Active Debris Removal, Combined Control, Robotics, Modelica, FMI

1 INTRODUCTION

Inactive satellites and debris, especially in the most used Low Earth Orbits (LEO), have been known to be a potential problem for a long time. The number of satellites increases every year, which increases the chance of a collision. Active Debris Removal (ADR) using satellites (chaser) equipped with a robotic arm is a possible technical solution to reduce this risk (Liou and Johnson 2006). ADR missions with robotic arms are very challenging, because they involve complex systems consisting of a satellite with a robotic arm with many degrees of freedom, while the information about a debris target is not always perfect (Weiss and Sarli 2014). Recent advances in the field of reinforcement learning (RL) could help solve problems in the field of ADR, especially when large uncertainties are involved. RL algorithms can adapt to changing environments, making them suitable for such tasks (Arulkumaran, Deisenroth, Miles Brundage, et al. 2017). However, the onboard hardware of satellites is usually very limited in its computational capabilities, and questions about the robustness of RL algorithms are still not completely solved (Kober, Bagnell, and Peters 2013; Han et al. 2023; Arulkumaran, Deisenroth, Michael Brundage, et al. 2017). In this paper, a hybrid trajectory planning algorithm is presented, which tries to combine conventional path planning methods and control strategies with an RL algorithm to address uncertainties in the ADR scenario.

Reinforcement learning (RL) is particularly attractive for ADR because it can learn adaptive strategies for collision avoidance and trajectory planning in the presence of nonlinear dynamics and uncertainty, where classical methods often face limitations.

The field of reinforcement learning (RL) for satellite on-orbit servicing (OOS) has witnessed notable advancements, particularly in the area of robotic arms on satellites. RL-based approaches have been developed to autonomously detect potential collisions, rendezvous with target satellites, and execute optimal collision avoidance maneuvers (CAMs) (Patnala and Abdin 2024). In addition, deep reinforcement learning (DRL) has been employed for autonomous guidance of redundant space manipulators. These approaches focus on solving path planning during the motion-synchronization phase with the mission target, using algorithms like Proximal Policy Optimization (PPO) to optimize the manipulator's guidance law (Ambrosio et al. 2024).

Furthermore, reinforcement learning algorithms with fast convergence have been proposed for routing in Low Earth Orbit (LEO) satellite networks. These algorithms address the dynamic topology changes and transmission requirements of LEO networks, enabling satellites to quickly adapt their routing strategies based on network link status updates (Ding et al. 2023).

Additionally, reinforcement learning techniques have been applied to path-planning for smart imaging of uncooperative space objects, enhancing the precision and efficiency of on-orbit servicing missions (Brandonisio, Lavagna, and Guzzetti 2021).

2 MODELLING AND SCENARIO

This paper focuses on a simulation study of an ADR scenario that is based on the inactive (target) satellite Envisat and a (chaser) satellite equipped with a robotic arm. The scenario starts with the chaser already in close proximity with variable initial conditions with respect to the starting distance and target spinning rate to Envisat. The angular

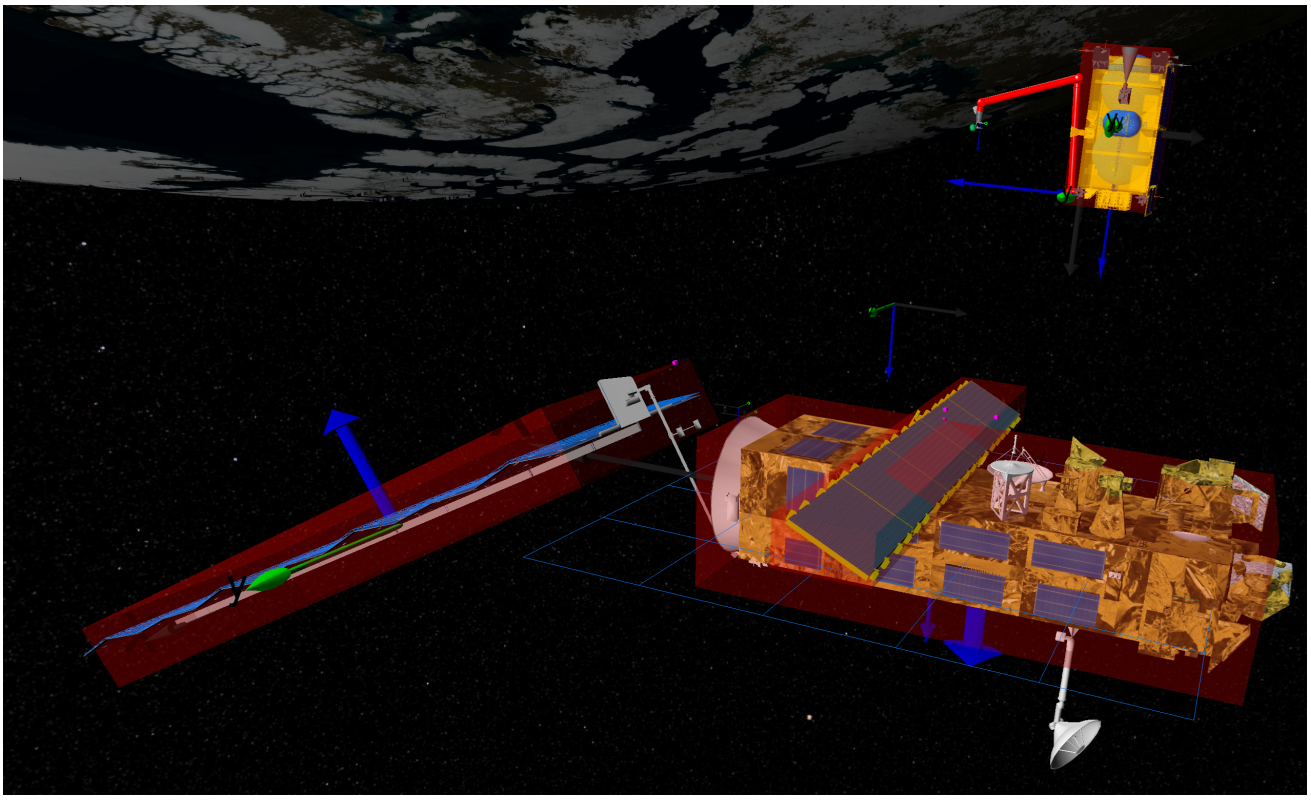


Figure 1. Visualization of the ADR scenario with Envisat and the chaser satellite with its 7-axis robotic arm and gripper. The four red bounding boxes around the target and chaser satellite are used for the collision avoidance of the RL algorithm. The computed critical closest points on the bounding boxes are shown as small pink spheres.

rates between target and chaser are assumed to be synchronized at the start of the simulation.

The scenario consists of multiple phases with different control modes. The first phase, which is the main focus of this paper, is the collision-free approach of the chaser satellite above Envisat's adapter ring and placing the gripper (also collision free) very close to the grasping point on the adapter ring.

This phase is handled by the hybrid RL algorithm which will be described in section 3. After this phase classical trajectory planning is used to stay synchronized with the spinning target using the combined control and grasping the adapter ring. After the gripper at the robot's tool center point (TCP) is closed, a detumbling mode is activated to reduce the angular rate of the target. After the detumbling is completed the deorbiting of the target would be started, but these phases are not considered in this work.

The models and scenario used are based on previous simulation studies, which were performed together with ESA (M. Reiner 2021; M. Reiner, Fernandez, and Ortega 2017; M. Reiner 2016), but were extended by additional uncertainty parameters for this scenario with respect to the tumbling (spinning) rate of the target and the starting distance for the trajectory planning.

The simulation models are created using the equation-based modeling language Modelica and dedicated Modelica-based libraries, the DLR Space Systems Library (M. Reiner and Bals 2014) and the DLR Environment Library (Briese, Klöckner, and M. Reiner 2017) and consider the following aspects (more details are given in the previous publications):

- Orbit dynamics for Envisat's sun synchronous orbit (LEO).
- First order approximation for the elasticity of Envisat's large solar array.
- Chaser satellite with 3D-pendulum models to approximate the fuel and oxidizer sloshing of the large tanks.
- Detailed control thruster array with 24 individual 22 N thrusters. The thruster firing pulse sequencing is implemented using a Pulse Width Pulse Frequency (PWPF) control in addition to a constrained least square (CLS) force and torque allocation.
- 7-Axis robotic arm mounted on top of the chaser with a fourth order approximation for each of the seven joints which approximate the flexibility of the gearboxes and friction as well as the internal high frequency torque control.
- Coupling and reaction forces between all elements are automatically considered by the multibody im-

DOI
10.3384/ecp218489

plementation supported by the Modelica Standard Library.

- The grasping of the robotic gripper of the adapter ring of Envisat is approximated by a switchable force and torque constraint with Baumgarte-like stabilization (Acquatella B. and M. J. Reiner 2014)
- Approximated IMU uncertainty using randomized simulated measuring noise and a simplified Exponentially Correlated Random Variable (ECRV) based camera performance model for the on-board camera and Light Detection and Ranging (LIDAR) system uncertainty.
- Robust H-infinity-based controller with outer Quaternion feedback loop for the combined control of the chaser and its 7-axis robotic arm.

To account for the uncertainty of the target's spinning rate, the initial angular velocity is allowed to vary within the following range:

$$\omega_{\mathrm{ta},0} = (\omega_{\mathrm{ta},0,x}, \omega_{\mathrm{ta},0,y}, \omega_{\mathrm{ta},0,z})^T \tag{1a}$$

$$\text{with: } -1^\circ/s <= \omega_{\mathrm{ta},0,x} <= 1^\circ/s, \tag{1b}$$

$$-1^\circ/s <= \omega_{\mathrm{ta},0,y} <= 1^\circ/s, \tag{1c}$$

$$-5^\circ/s <= \omega_{\mathrm{ta},0,z} <= 5^\circ/s \tag{1d}$$

Similarly, the (relative) starting distance of the chaser to the target (with respect to the target's CoM) is allowed to vary within the following range:

$$r_{\mathrm{ta},0} = (r_{\mathrm{ta},0,x}, r_{\mathrm{ta},0,y}, r_{\mathrm{ta},0,z})^T \tag{2a}$$

$$\text{with: } -2.0\,\mathrm{m} \leq r_{\mathrm{ta},0,x} \leq 2.0\,\mathrm{m} \tag{2b}$$

$$-2.0\,\mathrm{m} \leq r_{\mathrm{ta},0,y} \leq 2.0\,\mathrm{m}, -7.0\,\mathrm{m} \leq r_{\mathrm{ta},0,z} \leq -3.0\,\mathrm{m} \tag{2c}$$

For the RL algorithm, which requires many simulation steps, this model is too complex, and the training of the RL-algorithm would take an extremely long time on the available hardware (a single laptop with an Nvidia GeForce RTX 4090 GPU and Intel i9-13950HX CPU). Therefore, a simplified model is derived from the complex model for the training phase of the RL algorithm. However, the full model is later used for verification and validation (V&V). The simplified model only considers the kinematics for the chaser satellite and robotic gripper (at TCP). The spinning target is approximated as a rigid body with given initial conditions in a perfect zero gravity environment. The inputs to the model are direct kinematic translations for the chaser and robot TCP (without controller or actuator/sensor models). The outputs of the model are the observations and reward functions for the RL algorithm, which will be described in detail in sec. 3. This model can be computed very quickly, and is suitable for the time-consuming training process of the RL algorithm.

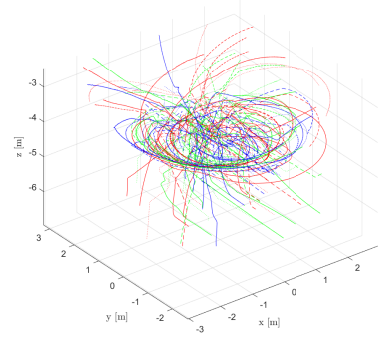


Figure 2. 3D-Visualization for the generated reference trajectories for the chaser satellite $r_{\mathrm{CH,ref}}$ (in relative target coordinate system) for the 100 random scenarios. From a random start location the trajectories convert to the desired position close to the grasping point on the target.

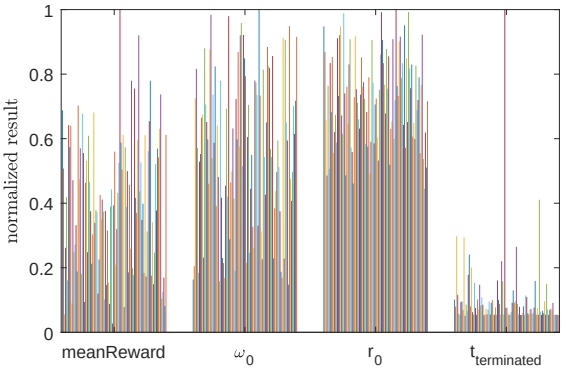


Figure 3. Visualization of the result of 100 random scenarios for different start values for the initial target angular velocity (vector length ω_0) and distance to target (vector length r_0).

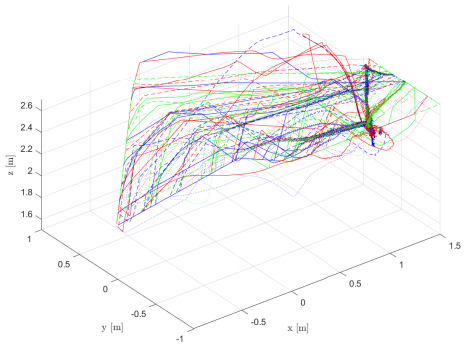
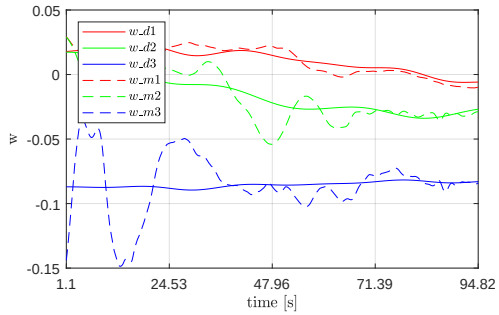
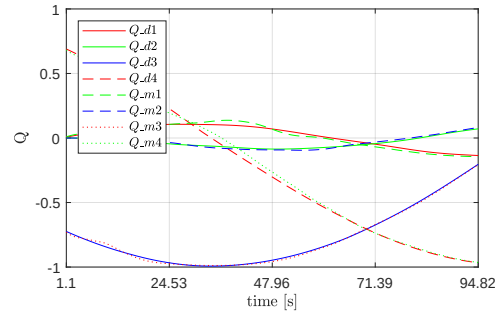


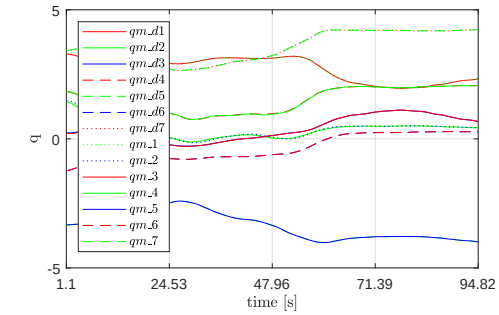
Figure 4. 3D-Visualization for the generated reference trajectory for the robot TCP $r_{\mathrm{TCP,ref}}$ (in local robot coordinate system) for the 100 random scenarios. The start location for all cases is $(-0.99, -0.01, 2.01)$m. The limitations for the TCP position given by $r_{\mathrm{TCP},min}$ and $r_{\mathrm{TCP},max}$ can be seen for some trajectories, that move along the given limits.



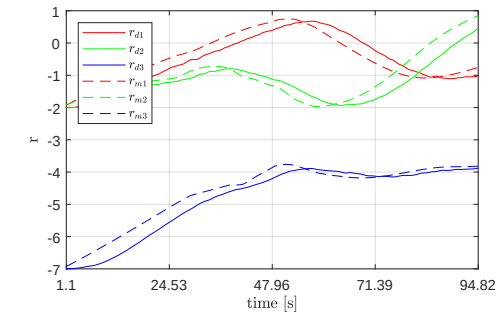
(a) Comparison of the commanded (ω_d) and measured (ω_m) angular velocity (including correction) to the combined controller in rad/s.



(b) Comparison of the commanded (Q_d) and measured (Q_m) Quaternion (including correction) to the combined controller.

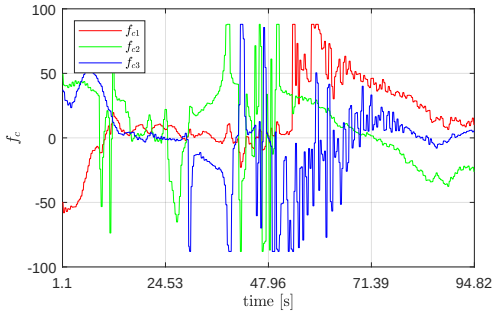


(c) Comparison of the commanded ($q_{m,d}$) and measured (q_m) robot motor position (including correction) to the combined controller in rad.

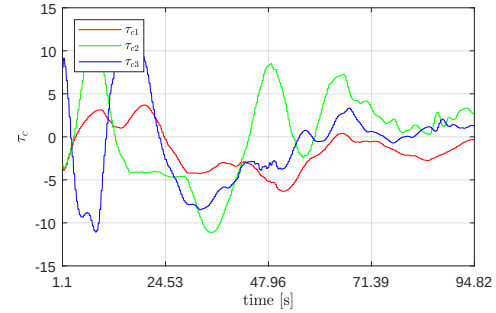


(d) Comparison of the commanded (r_d) and measured (r_m) chaser relative distance to the target in m.

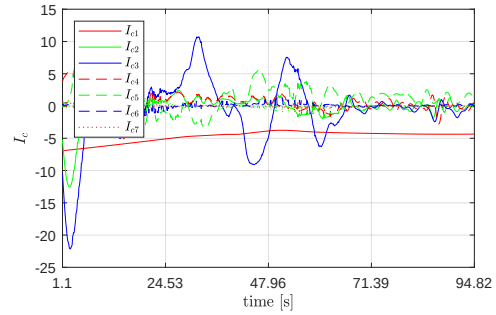
Figure 5. Simulation results for the generated command inputs and measured results for the combined controller for an example simulation case with high starting target angular velocities $\omega_{\text{ta},0} = (-1,-1,-5)^T\,^\circ/s$ and starting distance $r_{\text{ta},0} = (-2,-2,-7)^T\,m$.



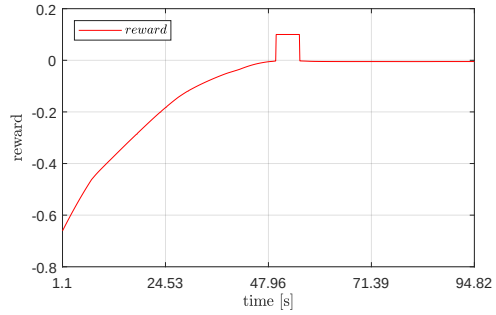
(a) Resulting thrust forces f_c commanded by the combined controller which are input for the force allocation to assign them to individual thrusters.



(b) Resulting thrust forces τ_c commanded by the combined controller which are input for the force allocation.



(c) Commanded motor torques for the 7-axis robot by the combined controller.



(d) PPO reward for the simulation case. The peak in the reward function (48 s) triggers the terminal condition C_{ter} and the classical trajectory planing takes over from the PPO algorithm.

Figure 6. Resulting thrust forces and torques for the chaser satellite and motor currents for the robot arm for the simulation case from Figure 5. Also plotted is the resulting reward function for the PPO algorithm.

DOI 10.3384/ecp218489

3 HYBRID REINFORCEMENT LEARNING ENHANCED TRAJECTORY PLANNING

The basic idea behind the hybrid reinforcement learning enhanced trajectory planning is to combine the advantages of the flexibility of an RL algorithm with classical trajectory planning for the chaser including an optimization-based inverse kinematics for the robot arm and correction terms for the simplified RL model.

Since the computational power of today's satellites is still limited, the algorithm is designed such that a simple RL part with relatively small neural network sizes is sufficient. It is then combined with correction terms and classical trajectory planning to achieve good accuracy and robustness with respect to the initial conditions and spinning behavior of the target satellite, sensor noise, as well as actuator and control limitations.

The RL base algorithm is a Proximal Policy Optimization (PPO) (Schulman et al. 2017) as implemented in DLR's Stable Baseline 3 (SB3) Python software package (Raffin, Hill, et al. 2021). PPO is widely used and has proven its capabilities in many applications, especially in the field of robotics (Arulkumaran, Deisenroth, Miles Brundage, et al. 2017; Kober, Bagnell, and Peters 2013; Han et al. 2023; Arulkumaran, Deisenroth, Michael Brundage, et al. 2017).

$$L^{\mathrm{CLIP}}(\theta) = \mathbb{E}_t\left[\min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)}\hat{A}_t,\right.\right.$$
$$\left.\left.\mathrm{clip}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)}, 1-\varepsilon, 1+\varepsilon\right)\hat{A}_t\right)\right] \tag{3}$$

The objective function for the Proximal Policy Optimization (PPO) algorithm can be expressed by Equation 3, were θ represents the parameters of the current policy, $\pi_\theta(a_t|s_t)$ is the probability of taking action a_t given state s_t under the current policy, $\pi_{\theta_{\mathrm{old}}}(a_t|s_t)$ is the probability of taking action a_t given state s_t under the old policy, $\hat{A}_t$ is the estimated advantage at time t and ε is a hyperparameter that controls the extent of clipping (which limits extreme changes). The policies are implemented as neural networks.

CATIA-Systems FMPy Python software package is used to simulate Functional Mockup Units (FMUs) in Python, which are generated by Dymola from Modelica Code for the models described in sec. 2. SB3 allows many models (environments) to be computed in parallel. This feature was extended to use FMPy and underlying Modelica models. For a fast and efficient implementation, Modelica/Dymola's feature of inline integration (Elmqvist, Otter, and Cellier 1995) is used, so that the numerical solver is directly part of the Modelica model inside the FMU. As a result, from the outside (SB3), the model can be used as a model with discrete step size and no additional integrator in SB3 is needed.

The basic idea for the RL part of the hybrid algorithm is to train a network to find a collision free trajectory for both the chaser satellite and the gripper, mounted at the TCP of the 7-axis robot arm, from a varying initial distance to the target, which can spin with different rates as defined in Equation 1 and Equation 2.

To simplify the collision avoidance problem, the target satellite is approximated by bounding boxes for the collision algorithm. For the target three different collision boxes are used. One for the main satellite bus and two for the solar arrays. Figure 1 shows the bounding boxes in transparent red. It is assumed, that the onboard cameras can accurately (with the modeled measurement noise) track these boxes for the target. The chaser itself is approximated by using an additional safety offset for the distance calculation (with respect to the chaser's CoM) according to its largest dimension, in addition, for the TCP a smaller offset is used to account for the size of the gripper.

For both the chaser and the robot's TCP the closest points on these collision boxes are computed (some are visible as small pink spheres in Figure 1) as well as the distance of these critical points to the TCP and chaser including the safety offsets for chaser and TCP.

Since the rotation of the chaser is controlled such that it tries to stay synchronized to the spinning target, only the translational part is of interest for the PPO algorithm (a correction described later will handle occurring deviations).

As observation inputs for the PPO algorithm, a series of normalized vectors are used. All are given as relative vectors with respect to the robot's mounting frame.

- The actual position of the TCP.
- The vector between the TCP and the target grasp point (with safety offset) shown in Figure 1 as small visual coordinate system (cosys) above the grasp point on the adapter ring.
- The vector between the chaser CoM and the chaser goal position close to the grasp point shown in Figure 1 as a local coordinate system above the central solar array.
- A set of $n_{vec} = 6$ vectors $d_{\mathrm{vec},i}$ describes the distance between the computed closest critical points for the three bounding boxes relative to the TCP and chaser center (with safety offsets).

The termination condition C_{ter} is achieved when both the TCP and the chaser CoM are within a given distance limit to the target positions as given in Equation 5, where $k_{\mathrm{safe}} < 1$ is just a safety factor to ensure the TCP is not very close to a bounding box and $k_{\mathrm{col,TCP}}$ is a scaling factor, to ensure the TCP is allowed to be closer to a bounding box than the chaser itself. $d_{TCP,Gr}$ is the distance between the TCP and the grasping end position (with safety offset), $d_{TCP,\max}$ is the maximum allowed deviation, $d_{\mathrm{CH},Gr}$ and $d_{\mathrm{CH,max}}$ are similar terms for the chaser. C_{col} is a criterion

for collision avoidance as defined in Equation 4. The reward r_t^{PPO} for the PPO is computed by Equation 6 if the termination is not reached yet. If the termination criterion is reached it is set to a constant high termination reward $r_t^{\text{PPO}} = R_{PPO,ter}$, which helps the PPO optimization to quickly find and keep the termination condition C_{ter}.

$$C_{\text{col}} = \sum_{i=1}^{n_{vec}} \frac{1}{\max(\varepsilon_d, \|d_{\text{vec},i}\|)} \tag{4}$$

$$\begin{aligned} C_{ter} \implies & d_{\text{TCP},Gr} < d_{\text{TCP,max}} \\ & \wedge\ d_{\text{CH},Gr} < d_{\text{CH,max}} \\ & \wedge\ C_{\text{col}}\, k_{\text{col,TCP}} < C_{\text{max,col}}\, k_{\text{safe}} \end{aligned} \tag{5}$$

$$\begin{aligned} r_t^{\text{PPO}} = \\ -\left(d_{\text{TCP,Gr}} + d_{\text{CH,Gr}} + \min\left(C_{\text{max,col}}, C_{\text{col}} k_{\text{col,TCP}}\right)\right) \end{aligned} \tag{6}$$

Since the optimization of the neural networks inside the PPO algorithm works better with scaled values, the reward and the observation vectors are normalized element-wise between zero and one using a scaling function $\text{f}_s()$ with predefined minimum and maximum values which scales an input vector using a combination of linear and sigmoid functions (near the limits), with adjustable bounds and smoothing.

The PPO neural networks consist of two networks (actor and critic value-function), each with two layers and 128 neurons per layer. The hyper-parameters for the PPO tuning were found using a hyper-parameter optimization implemented in Optuna (Akiba et al. 2019). In addition, generalized State Dependent Exploration (gSDE) (Raffin, Kober, and Stulp 2022) is used instead of action noise exploration.

The resulting action a_t of the PPO algorithm (Equation 7) is also a normalized vector with six elements (with each element between -1 and 1). The first three elements are the translation command for the gripper at the robot's TCP. The last three elements are the translation vector for the chaser satellite for the next time step. Both vectors are given with respect to the robot's mounting frame. A predefined scaling vector k_s is used to account for the faster dynamics of the robot arm compared to the chaser satellite (Equation 8). k_s is chosen such, that the robot and chaser should be able to handle the resulting translation movement in most cases. The actual agility depends on the current pose of the robot arm and the satellite, and is limited by the required thrust to achieve rotational synchronization, so k_s is an approximation. In addition, $a_{t,s,rob}$ is set to zero if the sum over all actions would move the relative TCP location outside the allowed bounds $r_{\text{TCP,min}} = (-1.5, -1.0, 1.0)\text{m}$ and $r_{\text{TCP,max}} = (1.5, 1.0, 2.7)\text{m}$. These bounds ensure that the robot arm is able to reach the location and that no self-collision with the chaser can occur.

$$a_t = \left(a_{t,\text{rob}}, a_{t,\text{ch}}\right)^T \tag{7}$$

$$a_{t,s} = a_t \circ k_s = \left(a_{t,s,\text{rob}}, a_{t,s,\text{ch}}\right)^T \tag{8}$$

For the PPO algorithm, the only measured sensor inputs are the target rotation matrix in each time step relative to the chaser, which is assumed to be measurable with the onboard sensors. Also available for the PPO algorithm is the relative distance vector between chaser and target at the start of the first phase. With this information and the geometric data for the bounding boxes, the PPO algorithm can compute a trajectory (and internally also the required observations and reward). The actual states of the chaser and robot arm are not fed back to the PPO algorithm. This has the advantage that the PPO algorithm works much more stably, since no instability due to feedback can occur; in some sense, it works as a feed-forward control for trajectory planning.

Since the actual states of the chaser and the robot arm can deviate from the internal state of the PPO (mainly because of actuation limitations and sensor noise and disturbance effects), a correction is needed. Thus, using the measured relative orientation $R_{\text{rob,base},m}$ and measured (relative) position of the chaser robot mounting frame $r_{\text{base},0,m}$, a corrected action $a_{t,\text{cor,rob}}$ is computed and transformed in the actual measured robot mounting frame Equation 9.

$$a_{t,\text{cor,rob}} = R_{\text{rob,base},m}\left(a_{t,s,\text{rob}} - r_{\text{base},0,m}\right) \tag{9}$$

For the position of the chaser, no such correction is made, since the TCP can be moved faster and errors in the rotation have only a small effect on the actual CoM position. The chaser directly follows the trajectory given by $a_{t,s,ch}$. Since $a_{t,cor,rob}$ is a Cartesian translation, an inverse kinematics algorithm is used to transform the trajectory in the robot joint space. As desired orientation for the TCP the orientation of the target grip point is used. A discrete implementation using a least squares algorithm, instead of the original constrained least squares (CLS) implementation from (Bellmann 2014), with additional limitations is used for performance reasons.

The PPO trajectory is followed until the condition C_{ter} is true. In this state the gripper is very close to the grasp point and the chaser close at a defined location near the grasp point.

From there classical trajectory panning algorithms are used for the grasping, detumbling and de-orbiting, as described in (M. Reiner 2021; M. Reiner, Fernandez, and Ortega 2017; M. Reiner 2016).

4 SIMULATION EXPERIMENT RESULTS

To test the proposed hybrid algorithm, a series of 100 simulations of the ADR scenario were performed, using the full complex model (instead of the simplified model used for training the PPO algorithm) as described in sec. 2.

Figure 2 shows the trajectories for the chaser satellite generated by $a_{t,s,\text{rob}}$. For the 100 simulations, the start values for the initial target angular velocity and distance to target were randomized within their respective bounds

DOI 10.3384/ecp218489

as defined in Equation 1 and Equation 2. For all 100 simulation the algorithm was successful and collision free.

Figure 3 shows the results for the mean achieved reward which is normalized with the minimal reward achieved (meanReward in the bar plot). The plot also shows the vector length of the initial target angular velocity and distance to target (normalized to the maximal values). The last bar shows the time it took for the termination, normalized to the longest time. The mean time for a termination was $40.99s$. The large visible outlier was a case were the algorithm was able to move both gripper and chaser very close to the target, but not close enough to achieve the terminal condition C_{ter}, only after $404.35s$ the condition C_{ter} was achieved. For such cases the algorithm could be improved in the future with a more adaptive terminal condition C_{ter} or an abort criterion if the phase takes too long.

Figure 4 shows the trajectory for the gripper at the TCP, which result from $a_{t,s,\text{rob}}$ for the 100 simulations.

Figure 5 and Figure 6 show some detailed results for one simulation case which is at the limit for the allowed initial target angular velocity. The trajectory, generated by the hybrid algorithm, is used as input for the combined controller which is used to control the chaser satellite and robot arm. The plots show the approach with the PPO algorithm and the transition to the grasping (using a classical approach). The de-tumbling and de-orbiting is not shown. The combined controller and classical trajectory planning used are described in more detail in the previous publications (M. Reiner 2021; M. Reiner, Fernandez, and Ortega 2017; M. Reiner 2016).

5 CONCLUSION AND OUTLOOK

This work describes a novel hybrid reinforcement learning enhanced trajectory planning for an ADR scenario for combined control of a satellite with a 7-axis robotic arm. A PPO algorithm is combined with a correction algorithm and classical trajectory planning to handle the collision free approach of a chaser satellite to a target and placing the gripper at the robots TCP near the grasping point for use with a combined controller, which commands the satellite and its robotic arm simultaneously.

The algorithm was tested in a complex simulation ADR scenario with 100 randomized initial conditions, and was able to generate collision free trajectories for all cases.

The PPO part of the algorithm is trained using a simplified model of the scenario, which allows fast training times and leads to small neural networks, which could be run on weaker space computing hardware. It acts as a feed-forward component to a correction algorithm, which helps to reduce instability caused by uncertain measurement feed-back and model mismatch.

In the future the algorithm could be extended to use a more complex model for the training if more computing power would be available and larger neural networks on board of the satellite.

The space industry is very restrictive in regards to use neural networks as part of the control but a hybrid algorithm, as presented here, which can be more easily supervised and extended with a fall-back solution could help the acceptance.

As potential directions for further work, systematic comparisons with other RL algorithms and classical control baselines could be explored, along with more detailed statistical analyses and ablation studies. It may also be valuable to investigate advanced domain adaptation and sim-to-real transfer techniques, as well as formal safety and stability guarantees, to further broaden the applicability of the proposed approach.

References

Acquatella B., Paul and Matthias J. Reiner (2014). "Modelica stage separation dynamics modeling for End-to-End launch vehicle trajectory simulations". In: *Proceedings of the 10th International Modelica Conference - Lund 2014*. Ed. by Hubertus Tummescheit and Karl-Erik Arzen. Vol. 96. Linköping Electronic Conference Proceedings. LiU Electronic Press, pp. 589–598. URL: https://elib.dlr.de/92218/.

Akiba, Takuya et al. (2019). "Optuna: A Next-generation Hyperparameter Optimization Framework". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Ambrosio, Matteo et al. (2024). "Redundant Space Manipulator Autonomous Guidance for In-Orbit Servicing via Deep Reinforcement Learning". In: *Aerospace* 11.5, p. 341. DOI: 10.3390/aerospace11050341.

Arulkumaran, Kai, Marc Peter Deisenroth, Michael Brundage, et al. (2017). "Deep reinforcement learning: A brief survey". In: *IEEE Signal Processing Magazine* 34.6, pp. 26–38. DOI: 10.1109/MSP.2017.2743240. URL: https://ieeexplore.ieee.org/document/8099596.

Arulkumaran, Kai, Marc Peter Deisenroth, Miles Brundage, et al. (2017). "A brief survey of deep reinforcement learning". In: *IEEE signal processing magazine* 34.6, pp. 26–38.

Bellmann, Tobias (2014). "Optimierungsbasierte Bahnplanung für interaktive robotische Bewegungssimulatoren". PhD thesis. Universität der Bundeswehr München. URL: https://elib.dlr.de/93031/.

Brandonisio, Andrea, Michele Lavagna, and Davide Guzzetti (2021). "Reinforcement Learning for Uncooperative Space Objects Smart Imaging Path-Planning". In: *The Journal of the Astronautical Sciences* 68, pp. 1145–1169. DOI: 10.1007/s40295-021-00288-7.

Briese, Lale Evrim, Andreas Klöckner, and Matthias Reiner (2017-05). "The DLR Environment Library for Multi-Disciplinary Aerospace Applications". In: *12th International Modelica Conference*. URL: https://elib.dlr.de/112971/.

Ding, Zhaolong et al. (2023). "Fast-Convergence Reinforcement Learning for Routing in LEO Satellite Networks". In: *Sensors* 23.11, p. 5180. DOI: 10.3390/s23115180.

Elmqvist, H., M. Otter, and F. Cellier (1995). "Inline Integration: A New Mixed Symbolic/ Numeric Approach for Solving Differential-Algebraic Equation Systems". In: *Proceedings of ESM, SCS European Simulation MultiConference*.

Han, Dong et al. (2023). "A Survey on Deep Reinforcement Learning Algorithms for Robotic Manipulation". In: *Sensors* 23.7, p. 3762. DOI: 10.3390/s23073762. URL: https://www.mdpi.com/1424-8220/23/7/3762.

Kober, Jens, J Andrew Bagnell, and Jan Peters (2013). "Reinforcement learning in robotics: A survey". In: *The International Journal of Robotics Research* 32.11, pp. 1238–1274.

Liou, J-C and Nicholas L Johnson (2006). "Understanding the full impact of space debris on the Earth environment". In: *Science* 311.5759, pp. 340–341.

Patnala, Susmitha and Adam Abdin (2024). "On-orbit Servicing for Spacecraft Collision Avoidance With Autonomous Decision Making". In: *Papers With Code*. URL: https://paperswithcode.com/paper/on-orbit-servicing-for-spacecraft-collision-avoidance-with-autonomous-decision-making.

Raffin, Antonin, Ashley Hill, et al. (2021). "Stable-Baselines3: Reliable Reinforcement Learning Implementations". In: *Journal of Machine Learning Research* 22.268, pp. 1–8. URL: http://jmlr.org/papers/v22/20-1364.html.

Raffin, Antonin, Jens Kober, and Freek Stulp (2022-08–11 Nov). "Smooth Exploration for Robotic Reinforcement Learning". In: *Proceedings of the 5th Conference on Robot Learning*. Ed. by Aleksandra Faust, David Hsu, and Gerhard Neumann. Vol. 164. Proceedings of Machine Learning Research. PMLR, pp. 1634–1644. URL: https://proceedings.mlr.press/v164/raffin22a.html.

Reiner, Matthias (2016). "GNC Simulation Tool for active Debris Removal with a Robot Arm". In: *Proceedings for the 6th International Conference on Astrodynamics Tools and Techniques (ICATT)*. URL: https://elib.dlr.de/104850/.

Reiner, Matthias (2021). "Real-time combined control for active debris removal for a satellite with a robot arm implemented on a SBC connected to a detailed multi-physics and VR simulation". In: *GNC 2021 ABSTRACT BOOK*. ESA. URL: https://elib.dlr.de/144621/.

Reiner, Matthias and Johann Bals (2014). "Nonlinear inverse models for the control of satellites with flexible structures". In: *Proceedings of the 10th International Modelica Conference - Lund, Sweden - Mar 10-12, 2014*. Ed. by Hubertus Tummescheit and Karl-Erik Arzen. Linköping Electronic Conference Proceedings. LiU Electronic Press, pp. 577–587. URL: https://elib.dlr.de/92164/.

Reiner, Matthias, Jesus G. Fernandez, and Guillermo Ortega (2017). "Combined control for active debris removal using a satellite equipped with a robot arm". In: *Proceedings for the GNC 2017, 10th International ESA Conferece on Guidance, Navigation and Control Systems*. URL: https://elib.dlr.de/115866/.

Schulman, John et al. (2017). "Proximal Policy Optimization Algorithms." In: *CoRR* abs/1707.06347. URL: http://dblp.uni-trier.de/db/journals/corr/corr1707.html#SchulmanWDRK17.

Weiss, Benjamin and Bruno M Sarli (2014). "Autonomous on-orbit servicing of satellites". In: *IEEE Access* 2, pp. 614–626.

 DOI 10.3384/ecp218489

Automatic Modelica Package and Model Generation from Templates and Data Files with Python, Exemplified with URDF

Antoine Pignède[1] Carsten Oldemeyer[1]

[1]German Aerospace Center (DLR), Institute of Robotics and Mechatronics, 82234 Weßling, Germany
{Antoine.Pignede,Carsten.Oldemeyer}@dlr.de

Abstract

Creating correct Modelica models and packages from templates and data files describing a multi-physics system is useful in numerous situations. For example, in laying out power plants, designing airplane air conditioning or chemical reactions (Barth and Fay 2013; Santillan Martinez et al. 2018; Ramonat et al. 2025). This publication shows two possibilities of doing this with an example from robotics design and simulation. The `URDFModelica` library contains templates for links, joints and whole robots that can be mobile or stationary. The library also has a Python script that takes a valid URDF file as input. With minimal manual processing of this input file, a complete robot simulation package is created automatically. Alternatively, the input file, translated to a Modelica record, can be used to set the parameters and connections of a generic robot model. The `URDFModelica` library has already been successfully used for quick generation of first Modelica simulations of existing robots or robots yet to be fully developed. The general structure and approach can be adapted to other application domains without much effort. `URDFModelica` has been released as open source Modelica library on GitHub.

Keywords: model and package generation, Python Modelica processing, URDF, robotics

1 Introduction

There are numerous examples through all domains where Modelica is one among several tools for design, development, modeling, simulation, analysis etc. Keeping a consistent set of parameters through all tools is crucial but can be challenging and the same also holds for model structures and elements interaction. Ideally, there is one data file or set of files that compiles all this information in usable form and with consistent units. Examples of widely accepted data formats are **J**ava**S**cript **O**bject **N**otation *JSON*[1], E**x**tensible **M**arkup **L**anguage *XML*[2] and **Y**AML **A**in't **M**arkup **L**anguage™*YAML*[3].

But even if the data file fulfills these requirements, there needs to be some function in the tools to read and possibly write the data file. The `ExternData`[4] Modelica library (Beutlich and Winkler 2021) for example, does that. It reads files at initialization to set parameters, initial values and lookup tables. But the Modelica user still needs to create the models.

Taking robotics as example, there are situations when one data file can contain more information than just parameter and initial values. The **U**nified **R**obot **D**escription **F**ormat *URDF*[5] [6], introduced by the **R**obot **O**perating **S**ystem *ROS*[7] (Macenski et al. 2022), describes a robot as a set of links with appropriate attributes and joints that connect links together, more details in subsection 2.1. As such it can not only be used to set parameters but to create whole packages automatically. This works well because the model structure is fixed before initialization. URDF is supported by many CAD software packages. Simulation tools, e. g. CoppeliaSim[8] and Isaac[9], can build simulation models starting from a URDF file. The similar capability for Modelica is newly introduced in this work. The new `URDFModelica` library developed at the German Aerospace Center (DLR) parses a valid URDF file and automatically creates a complete robot simulation model as a new Modelica package or as a Modelica record that configures one model using variable-length arrays. For the simulation model as Modelica package, it creates separate models and parameter records for each link and joint as well as the robot model, that collects all elements and connects them to chains. It is thus possible to adapt all elements individually whenever needed without affecting all elements of the same kind. The separation between parameters and models also permits an update function that only rewrites parameter values. For the simulation model as single Modelica record, a generic robot model is contained in the library. This has two arrays of variable length for links and joints. Connections between these and for the flanges of active joints are managed by two matrices generated from the URDF source by the Python script. This way to automatically create a robot simulation needs

[1] `https://www.json.org/`
[2] `https://www.w3.org/XML/`
[3] `https://yaml.org/`
All URLs in footnotes are accessed July 22, 2025.

[4] `https://github.com/modelica-3rdparty/ExternData`
[5] `http://docs.ros.org/en/jazzy/Tutorials/Intermediate/URDF/URDF-Main.html`
[6] `https://wiki.ros.org/urdf`
[7] `https://www.ros.org/`
[8] `https://coppeliarobotics.com/`
[9] `https://developer.nvidia.com/isaac`

considerably less code but one cannot change individual elements in contrast to the simulation as new package.

Translation in the other direction, i. e. from Modelica to URDF, is currently not implemented and not planned for the near future because no application has been identified.

`URDFModelica` is different from `ExternData`, but also different from `Modelica-Builder`[10]. It does not manipulate arbitrary Modelica files, but only translates from one format to Modelica syntax for the special application type of robotics simulation. It also is different from the supported automated design (Matei et al. 2024) whose aim is to automatically create Modelica models, too. But, `URDFModelica` only translates an existing design from one description language to another while the supported automated design uses optimization from basic elements to create new models fulfilling specific tasks.

All three categories, read parameters defined externally, manipulate individual Modelica files externally, create whole packages for special applications automatically, have their area of applications and are useful in many fields. The approach and structure of the new library explained here can be adapted to other application domains without much effort. Therefore, the interest of this publication may go beyond the Modelica robotics community.

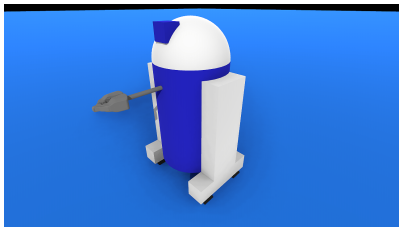


Figure 1. DLR Visualization 2 of the example mobile robot described in URDF and simulated in Dymola.

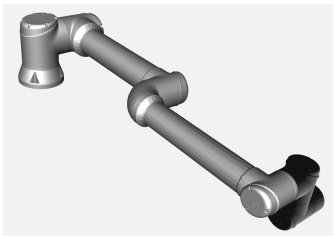


Figure 2. Standard Library Animation of the example stationary robot described in URDF and simulated in OpenModelica.

As appetizer, Figure 1 and Figure 2 show a mobile and a stationary robot created and simulated with `URDFModelica`. These are the R2D2-inspired example robot of the official ROS 2 documentation and the UR10e manipulator from Universal Robots[11]. They are contained as examples in the library, that has been released as open source Modelica library in Summer 2025: **`https://github.com/DLR-RM/urdfmodelica`**.

[10] `https://github.com/urbanopt/modelica-builder`

[11] `https://www.universal-robots.com/products/ur10e/`

The purpose of this publication is twofold. The first purpose is to generally explain how package and model generation can be done. The methodology is adaptable to other domains and other similar research questions.

The second purpose is to introduce the new `URDFModelica` library. This is split into two sections because two approaches for simulating robots based on URDF are implemented. The *package* approach and the *model structure defining record* approach. Independent of this, there are two variants for visualization, either with multibody animation from the Standard Library or the DLR Visualization 2 (Kümper et al. 2021).

Therefore, the text is structured along the two approaches, first the package then the record approach. Each time, a concept is explained theoretically first and then exemplified. Before this, the data source file and a general overview of `URDFModelica` are explained because they are the basis for both approaches.

2 Data Source File

Automatic package or model generation at first needs a good data source. There are numerous formats for this that cannot be covered entirely here, but some general guidelines with Modelica in mind can be given.

Ideally, one should adapt a few things before the Modelica package is created. For example, the export may result in wrong data types like real-valued color triplets instead of the integer-valued triplets required by Modelica. Most probably, file paths to resources such as lookup tables or CAD files for visualization must be changed in order for Modelica to find them. The recommendation is to put all these files in the resources directory of the Modelica package and refer to them relatively following the pattern `filename="modelica://Package/Resources/..."`. These steps are easier if the format is human-readable, but that is not a hard requirement.

Unfortunately, users tend to stretch, go beyond or abuse the capabilities or intentions of the data formats even if they are standardized. For example, JSON does not include comments, although they are occasionally seen in practice. YAML, as a second example, is not standardized. This makes handling, especially of large files, difficult and checking correctness is nearly impossible.

Based on experience of the authors, it often is beneficial not to include absolutely everything in the data file but rather restrict to the minimum that everybody agrees on. The automatically created package will almost always need post processing anyway.

2.1 Example: URDF and Modelica

> «*URDF* (**U**nified **R**obot **D**escription **F**ormat) is a file format for specifying the geometry and organization of robots in ROS.»[12]

[12] `http://docs.ros.org/en/jazzy/Tutorials/Intermediate/URDF/URDF-Main.html`

DOI 10.3384/ecp218497

URDF is in XML style and thus human-readable. A robot is a set of interconnected links and joints. Links have inertial properties, and optionally visual or collision properties. Joints have a type (fixed, rotational, translational, etc.), an axis, a parent link, a child link and a few other optional properties. URDF reserves keywords for more elements such as sensors and special Gazebo[13] elements, but these are not adopted by every software. For example, the `urdf_parser_py`[14] Python module, on which the conversion from URDF to Modelica is based on, only parses links and joints.

URDF has become one of the most popular formats in the robotics community (Tola and Corke 2023). Tola and Corke (2024) lists and discusses a comprehensive compilation of URDF robot models through the world. A large number of known CAD software are able to export their models as URDF, the ROS 2 documentation export page has a good overview with links to respective manuals.[15]

Thanks to the already existing Python parser for URDF, minimal preprocessing of the data source file is needed. Most important is that file paths point to resources that Modelica can find. The parser does not check for consistent values, e. g. whether an axis attribute is a unit vector, and recognizes only the minimal set of elements and attributes that everybody agrees on. For example, the ROS 2 documentation defines contact coefficients for the link collision model that are not recognized by `urdf_parser_py`.[16]

3 `URDFModelica` Library

The `URDFModelica` library contains records, models and resources to easily set up a simulation from an existing URDF file. See the library tree Figure 3. The two approaches described here, are contained in the library.

3.1 `URDFModelica` Links

The link model can be any rigid robot body or represent some other important robot part like a camera position or a measurement point. Associated with a link model is a link parameter record that has three records: inertial, visual and collision. These three parameter records themselves contain further records down the hierarchy until primitive data types such as mass and inertia real values, integer color vectors or strings for paths to CAD files.

A link always has an inertial component, that is a `Body` from the multibody Standard Library with parameters from the corresponding record, which can be zero.

Link visualization is optional. It is always included, but only active if a Boolean parameter is true. In the "main" variant, it is simulated using the visualizers from the Modelica multibody visualizers. The `geometry` type that is part of the link parameter record specifies which visualizer to conditionally include. The user may also choose to replace the Standard Library visualizers with the corresponding elements from the DLR Visualization 2 library with a simple script option.

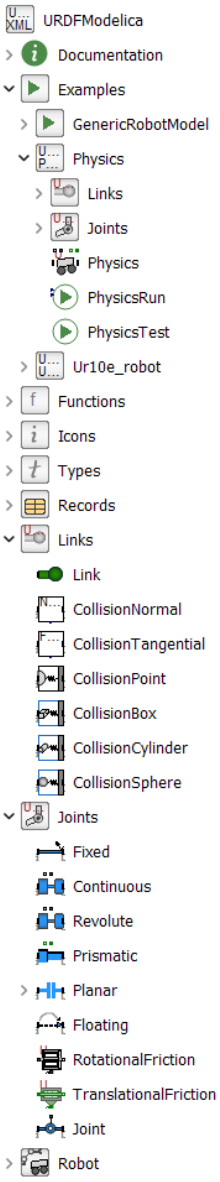


Figure 3. URDFModelica library tree.

Collision of links is optional, too. Again, it is always included in the template and active only if a Boolean parameter is true. Because the focus of `URDFModelica` is on the package generation and translation from URDF but not on contact detection and dynamics, collision is not simulated but only visualized. The same visualizers from Standard Library or DLR Visualization 2 as for link visualization are included equally in each link that has collision information. Color is pink for easier distinction. Collision can be globally turned off with a parameter that is propagated through the whole robot.

The link model stands as is and does not need to be processed by the automatic package generation to work. One can add a link to any multibody model of Modelica because all parameters have default values and optional components are conditional. The advantage of the URDF link compared to a Standard Library body is the attached parameter record with which it is easy to have different rigid bodies with optional visual and collision models. Additionally, the resulting Modelica model uses the same parameters as defined by URDF, which makes the backwards translation (currently not automatized) simple.

3.2 `URDFModelica` Joints

Joints connect links together. Different from links, there is not one joint with conditional components, but different models for each type. Still, all joint models have an instance of the same joint parameter record. This record has an enumerated type and nine records, e. g. for the parent and child link names, the joint axis, limits for effort and velocity, friction and calibration information. Note that not all information contained in the URDF joint description (and `URDFModelica` joint parameter record) are used in Modelica simulations.

[13] `https://gazebosim.org/home`

[14] `https://github.com/ros/urdf_parser_py`

[15] `http://docs.ros.org/en/jazzy/Tutorials/Intermediate/URDF/Exporting-an-URDF-File.html`

[16] `http://docs.ros.org/en/jazzy/Tutorials/Intermediate/URDF/Adding-Physical-and-Collision-Properties-to-a-URDF-Model.html#contact-coefficients`

There are six types. All of them first apply a Standard Library `FixedRotation` to set their origin.

- `Fixed` joints rigidly connect two links together without any degree of freedom.
- `Continuous` joints connect two links around a `Revolute` from the multibody Standard Library whose axis is set by the `Axis` parameter record. The joint flanges are also propagated to the outside such that the angle can be controlled. An integer parameter can optionally be set to mark the flange number for easier reference in a robot diagram.
- `Revolute` is the same as `Continuous` with the difference that limits for angle (upper and lower limit), angular velocity absolute value and effort (torque) absolute value are checked against at. A warning is output if a limit is violated.
- `Prismatic` is the same as `Revolute` for translational connection between links using `Prismatic` from the multibody Standard Library. All prismatic joints have limits for position (upper and lower limit), velocity absolute value and effort (force) absolute value. These are zero by default as defined by URDF. A warning is output if a limit is violated.
- `Planar` joints connect two links together with three degrees of freedom around a `Planar` from the multibody Standard Library. The `Axis` parameter record from URDF is perpendicular to the unconstrained degrees of freedom. A function calculates the simplest possible required n_x parameter by exchanging two vector elements of which one is not zero such that it can be negated.
- `Floating` joints connect two links together with all six degrees of freedom around a `FreeMotion` from the multibody Standard Library.

`Revolute` and `Prismatic` joints may have friction. URDF reserves two parameters for this: a friction and a damping coefficient. Therefore, simple friction models are also contained in `Joints`. The friction force f_r depending on velocity v is

$$f_r = R_v v + \frac{2R_h}{1+\exp(-sv)} - R_h$$

that is continuously differentiable. Parameters are the viscous friction coefficient R_v (URDF damping), stiction coefficient R_h (URDF friction) and steepness s of stiction (default is 100). Rotational friction is the same with torque and angle replacing force and position.

The joint models stand as is and do not need to be processed by the automatic package generation to work. One can add any joint to any multibody model of Modelica because all parameters have default values and optional components are conditionally included.

3.3 `URDFModelica` Background Information

`URDFModelica` only relies on the Modelica Standard Library (`4.0.0`) and needs Python to translate from URDF to Modelica. It has been developed and tested on OpenModelica `v1.24.0`[17] (Fritzson et al. 2020) and Dymola `2024x`[18]. The *model structure defining record* approach however does not yet work with OpenModelica. Instead of the default Standard Library visualizers, one may also choose to do object visualization with the DLR Visualization 2 library (Kümper et al. 2021). Both variants are provided in the library and an option on translation lets the user decide which variant to take. The "visualization 2" version has been developed and tested only with Dymola because DLR Visualization 2 currently does not support OpenModelica. The restricted free community edition[19] suffices to run the examples.

The Python code has been developed and tested with Python `3.12` from the official sources[20]. The module `urdf_parser_py` has been installed with `pip`.

4 Automatic Package Generation in Modelica

Creating a Modelica package automatically is not a trivial undertaking, even with well-defined data source and templates. A general approach that is adaptable to many domains and situations is explained in this section. This is the first approach of this publication subsequently referred to as *package* approach.

In overview terms, there is a Modelica library that contains template models, parameter records definitions and resources. Among the resources is a Python translation script and space for necessary files for the automatically created package to work correctly. These can be lookup tables, CAD files, JSON initialization files, etc. The separation of parameter records and Modelica models is an important part of the proposed automatic package generation. Counterintuitively, created records and files do not extend form the templates but do full copies of their content. The advantage of individual model alteration is judged higher than the drawback that one cannot simply change all similar models in one step. This and more aspects will be elaborated in the following subsections.

4.1 Modelica Records

A central part of the suggested package structure is to separate parameters from models, a general pattern that is widely applied in modeling and simulation. Thus, the first task is the definition of Modelica records. Here, one must replicate the elements contained in the data source file that

[17] `https://openmodelica.org/`

[18] `https://www.3ds.com/products/catia/dymola`

[19] `https://www.sr-scil.de/simulationsbibliotheken/kommerziell-verfuegbar/visualization/`

[20] `https://www.python.org/`

DOI 10.3384/ecp218497

are potentially needed for the simulation model in Modelica syntax. A hierarchical approach and providing default values is recommended.

When the automatic package generation is run, instances of the parameter records are created and given the values of the data source file. These copies are given the same name as the model they refer to, with "Parameters" as suffix and saved as single files in the new package.

4.1.1 `URDFModelica` Records

A URDF-described robot is a set of links and joints. Thus, there is not one but two records on the top of the hierarchy.

4.2 Modelica Template Models

Having template Modelica models helps to automatically create useful Modelica packages. One could create partial templates and then extend the concrete instances. However, a different approach is put forward here. Having copies of the templates allows to change individual models after generation. The drawback that changes cannot be applied to all models at once, is deemed less important. If needed, packages can be recreated at any moment, but this comes at the cost of losing individual modifications.

A design choice concerns the number of template models and enumerated types. One can decide to have one template model that fits all purposes and selects its conditional components based on the variant. Otherwise, one can choose to define one, generally simpler, template model for each variant. The decision which approach to use, depends on preference, data source, and mainly on the interface to other parts of the model. Less important for the decision are the icons and number of components in each template. Both approaches are implemented in the example of this publication. There is one template for all links, regardless whether they are boxes or cylinders, and six templates for joints, one for each joint type defined by the data source standard. All joint types have their own template because some of them have flanges in addition to the two multibody frames needed by all joints. All links use the same template because they all only have one multibody frame to attach them to joints or other links. Conditional components inside a link do not introduce much complexity. On the contrary, adaptation to other link types, e. g. converting a CAD defined wheel to a cylinder, is made easy.

A model always needs the corresponding parameter record. The template itself has its parameter record as replaceable component. This permits instances of the template model to stand for themselves and be added manually to any Modelica model. Then, one can select existing records, possibly overwrite the defaults and instantiate new models. When the automatic package generation is run, the replaceable parameter record is not copied. Instead, the correct record, that is the one with the same name plus "Parameters" suffix, is added without the "replaceable" keyword because one should not need to replace the parameter record with another one in post processing. The instance name is always "parameters" for easy reference.

Other contents of templates can be divided in required and optional components. Required components, e. g. interface, are simply copied when automatically created. Optional components are declared conditional. This is a second requirement so that instances of a template can stand for themselves. When the automatic package generation is run, one could copy all optional components relying on Modelica conditional components. However, the Python script only copies the needed optional components to make the new model smaller.

4.2.1 `URDFModelica` Templates

Because a URDF-described robot is a set of links and joints, there are two packages with templates. Thus, the automatic package generation will create two subpackages for links and their parameters as well as for joints and their parameters.

Link Template

The link model described in subsection 3.1 serves as template for links described by URDF.

When a new package is created, copies of the link template and record are put in a new `Links` subpackage for each link. Thus, the link models can be adapted independently of the parameter definitions. Similarly, parameter values can be updated without modifying the link models.

The correct visualization and collision components (box, cylinder, sphere, CAD file), if any, are selected from all components contained in the template based on the `geometry` type that is part of the link parameter record.

Joint Templates

The joint models described in subsection 3.2 serve as template for joints described by URDF.

When a new robot package is created, copies of joints of appropriate type are put in the new `Joints` subpackage for each joint. A joint parameter record copy is added for each joint, too. Thus, the joint models can be adapted independently of the parameter definitions, parameters which can be updated at any time without modifying the joint models.

Different from links, joints have no conditional components because there is not one general joint template, but six more specialized joint templates.

4.3 Aggregate Modelica Template Model

The last required template is the one that combines all elements together to the whole system. This template is mostly empty apart from the interface. When the automatic package generation is run, the previously created models of the subsystems are added to a new system model and connected together based on the data source file. This is done in the Python script with string manipulation functions that create correct Modelica syntax before and after the "equation" keyword.

4.3.1 `URDFModelica` Robot Template

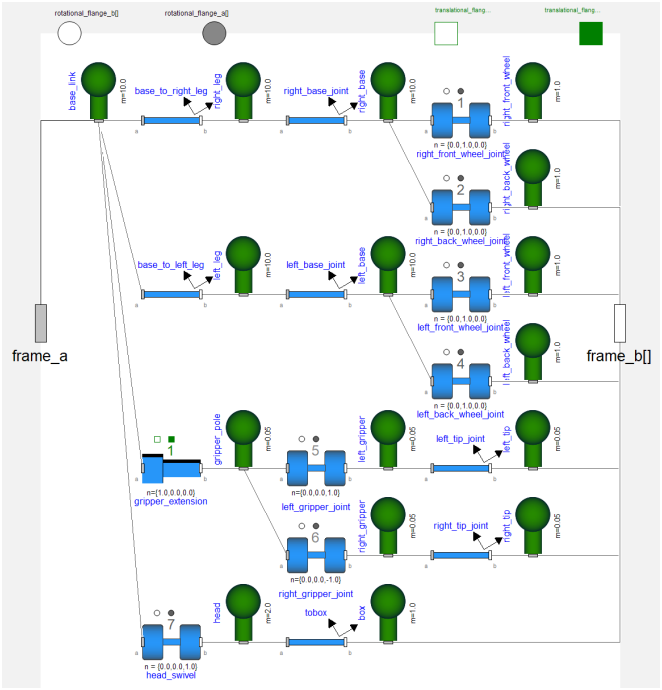


Figure 4. Graphics view of the example mobile robot.

The `Robotname` model puts all links and joints together and connects them according to the parent and child link string contained in each joint record. The graphics view of the R2D2-inspired example (Figure 1) is shown in Figure 4. The first link is connected to a `frame_a` on the left. All links that are leaves of chains are connected to a vector of `frame_b` on the right of the model to allow for interaction with external components. Additionally, rotational and translational flanges are propagated to the outside for every continuous, revolute and prismatic joint. The correct array sizes are calculated automatically. For easier reference in the joint icons, the flange number is also given as parameter to the concerned joints. The robot model diagram size is automatically adjusted to show all elements. This is the last model relying on the URDF source description, the further two Modelica models are suggestions for usage.

4.4 Modelica Usage Models

Further templates for how to create example models or FMUs may also be convenient for users.

4.4.1 `URDFModelica` Usage Models

For example, `RobotnameRun` builds an environment around `Robotname`. There is a multibody `World`, and outside inputs for position and angle of joints. `RobotnameRun` is one possible model to compile as FMU[21] (Blochwitz et al. 2012).

`RobotnameTest` is a model to quickly check that the package has been created correctly by providing some simple inputs to all needed signals.

[21] `https://fmi-standard.org/`

4.5 Python Script

The actual generation of the Modelica package is done with a Python script. Modelica also has text and file manipulation functions, but lacks the versatility and ease of Python. For example, there are parser modules for XML or JSON files that make the translation from these formats to Modelica syntax a lot easier.

Inputs to the Python script are the data source file and the location where to save the new package. The latter must be divided into two inputs, the path to the directory where the Modelica package structure begins, and the Modelica package inside this package structure. This is needed because each Modelica file begins with the keyword “within” and the Modelica package structure.

4.5.1 Package Generation

The Python script in general performs the following tasks.

1. Parse the input file. This is one of the main reasons to use Python because there already are modules for many data formats such as JSON, XML or YAML.
2. For each element
   (a) Create a new parameter record by translating the parsed information to Modelica syntax. Save the parameter record in the given package structure, possibly in further subpackages, with element name plus “Parameters” suffix.
   (b) Create a new model by copying only the needed contents of the template. Add the parameter record as “parameters” instance. Save the model in the given package structure, possibly in further subpackages, with element name.
3. Create the model that combines all elements to the system. Add all elements and connect them based on the actual source. Save this model in the given package structure with system name.
4. Optionally create and save a model that shows how to use the system.
5. Create the package order file of the new package.
6. Create the package definition file of the new package.
7. Update the package order one hierarchy level higher.

Note also that it is recommended to convert all names to title case, i. e. first letter upper case and the rest lower case, and that they should not contain "-" (minus) signs or be reserved keywords.

4.5.2 Parameter Update

One main reason for the division between parameters and models is to be able to update parameters without rewriting the whole package. This is possible if only numerical parameter values change in the data source file. In that

DOI 10.3384/ecp218497

case, the same Python script as above only recreates the parameter record files and skips all the rest.

Because the models of elements and the overall system model remain unchanged, it is not possible to change element types or connections in update mode. If the data source has such changes, one must completely rewrite the whole package or create a new one.

4.5.3 `URDFModelica` Python Script for Package Approach

The Python script `urdf_to_package.py` in the resources does the translation from URDF to Modelica syntax and creates or updates the Modelica package. It relies on the existing Python URDF parser but does not need further information or other non-standard modules otherwise.

Given a URDF file it first translates all information into one Modelica record exactly replicating the URDF contents. Then it writes, or rewrites, the parameter records of links and joints in the specified package. If the user chooses the update mode, the script terminates. Otherwise the entire package structure, with joints models, link models, robot model and example files, is recreated. Details about specific steps in this process are given in the respective paragraphs about links, joints, etc. further above.

5 Simulation Based on Modelica Record

Instead of generating new packages for every source file, an alternative way is a model that adapts to the structure during translation. While Modelica is not capable of handling systems with variable structure during simulation, it supports component arrays with varying sizes as long as those are known during translation. This is the second approach of this publication subsequently referred to as *model structure defining record* approach.

While retrieving the connection information from the data source file with a Modelica function is theoretically possible, it does not work with the Dymola version used by the authors. The tool must be able to expand the subscripts of connected variables into literals. Hence, a `connectionMatrix` containing that information is written into the single, complete record by the Python script as a workaround and used as shown in Listing 1. Each connection corresponds to a matrix row containing the indices of connected elements in its columns. Note that Listing 1 is adapted to the URDF and Modelica example and thus has two arrays of different elements (links and joints) and one connection matrix, because URDF defines parent links and child links to each joint.

Similarly, matrices for outside connections must be retrieved from the data source file and added to the model. In the robotics example of this text, it is the flanges of rotational or translational joints.

Finally, knowing the base and end elements of chains is usually necessary for interfacing with the environment. Roots and leaves can be determined relatively easily from the `connectionMatrix`.

Compared to the package generation approach of section 4, the simulation based on a single record reduces the amount of generated code and therefore maintenance effort considerably. In fact, the same few lines of Modelica code are enough to simulate all translated data source files. This implementation lends itself to use-cases where the system is frequently changed and used without modifications, e. g. when choosing the right system for a given task based on a simulation study.

A drawback is the lack of accessible presentation of the mapping from component name to index in the component vector that is more user-friendly with the package generation approach, see Figure 4. Changes in individual components are not possible either. As of writing model translation fails with OpenModelica with preliminary analysis indicating an error in component array handling.

5.1 `URDFModelica`: Simulation Based on Modelica Record

The `URDFModelica` library also does simulation from URDF source in a generic form following the *model structure defining record* approach. The source file rigidly defines the number of joints and links as well as their connections. A literal URDF robot implementation consists of two variable size arrays of links and joints whose size is the same as their respective records in the `Robot` record. The DLR Robots library (Bellmann et al. 2020) frequently uses this alternative way to create robots from URDF.

Listing 1. Simplified Excerpt of Robot Model.

```
model GenericRobot
 (...)
 Records.Robot data;
 Integer nJ = size(data.joint, 1);
 Integer nL = size(data.link , 1);
 Integer cM[nJ, 2] = data.connectionMatrix;
 Joint joint[nJ](jointParams = data.joint);
 Link  link [nL](linkParams  = data.link);
equation
 for iJ in 1:nJ loop
  connect(link[cM[iJ, 1]].frame_a,
   joint[iJ].frame_a);
  connect(joint[iJ].frame_b,
   link[cM[iJ, 2]].frame_a);
 end for;
 (...)
end GenericRobot;
```

The Python script of subsubsection 4.5.3 also creates a parameter record `Robot`, that contains the complete translation in one file. The translated links and joints that are used individually in the *package* approach, first are compiled to two arrays. Then, two additional matrices are generated from the joint information.

- `connectionMatrix[numJoints,2]`, theoretically explained in section 5, in practice becomes an array where each joint corresponds to a matrix row containing parent and child link indices in its columns. Usage is shown in Listing 1.

- `moveableJoints[numAxes,2]` has the joint number and movable axis number for every `Continuous`, `Revolute` and `Prismatic` joint. This sets connection of joint flanges to outside components at model translation.

Another part of the interface is a base frame that is used to place the robot in its environment. Which link to connect to the base frame is determined based on the `connectionMatrix` with a Modelica function that searches for the single root of the connection tree.

Higher level components need to provide the URDF robot record to the robot model via modifier.

6 Conclusion and Further Work

Automatic package generation or parameterization of generic models can be very useful, especially to keep consistent parameters and structures of models. It simplifies the initial setup and encourages good modeling practices, such as separating parameters from models.

This text explains two approaches, *package* and *model structure defining record*, that are adaptable to many domains and applications without much effort. The new `URDFModelica` library implements both approaches. It exemplifies the approaches and shows their practicality. The library has been released open source under BSD license on GitHub in Summer 2025. First users report quick translation, even of large robots (> 25 joints), and in general stable execution.

Further work can go in different directions. While there is not much to add to the general and more theoretical terms, `URDFModelica` can be expanded for example for proper collision models (Buse et al. 2023) or include more advanced drives or expand the range of robot elements. Although the Python parser does not recognize sensors, the URDF documentation reserves keywords for IMUs, cameras etc. An in-house extension to the library that also considers this, is already in use at DLR. Once `URDFModelica` has been released as open source Modelica library, efforts for extending `urdf_parser_py` will be undertaken. More examples of how to use the created Modelica models is another possibility for further work. For example, calculation of Jacobian or Coriolis matrices are important in robotics research.

Finally, the claim that the approach can be adapted to other domains, remains to be proven.

Acknowledgements

The authors would like to thank Robert Reiser for being the first person to test the library and Bernhard Thiele for proof reading the manuscript.

This work was supported by the Helmholtz Association project iFOODis (contract no. KA-HSC-06 iFOODis).

References

Barth, M. and A. Fay (2013). "Automated generation of simulation models for control code tests". In: *Control Engineering Practice* 21.2, pp. 218–230. DOI: 10.1016/j.conengprac.2012.09.022.

Bellmann, T. et al. (2020). "The DLR Robots library – Using replaceable packages to simulate various serial robots". In: *Asian Modelica Conference 2020, 8-9 October 2020, Tokyo, Japan*. Linköping University Electronic Press, pp. 153–161. DOI: 10.3384/ecp2020174153.

Beutlich, T. and D. Winkler (2021). "Efficient Parameterization of Modelica Models". In: *14th International Modelica Conference, 20-24 September 2021*. Linköping University Electronic Press, pp. 141–146. DOI: 10.3384/ecp21181141.

Blochwitz, T. et al. (2012). "Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models". In: *9th International Modelica Conference, 3-5 September 2012, Munich, Germany*. Linköping University Electronic Press, pp. 173–184. DOI: 10.3384/ecp12076173.

Buse, F. et al. (2023). "A Modelica Library to Add Contact Dynamics and Terramechanics to Multi-Body Mechanics". In: *15th International Modelica Conference, 9-11 October 2023, Aachen, Germany*. Linköping University Electronic Press, pp. 433–442. DOI: 10.3384/ecp204433.

Fritzson, P. et al. (2020). "The OpenModelica Integrated Environment for Modeling, Simulation, and Model-Based Development". In: *Modeling, Identification and Control* 41.4, pp. 241–285. DOI: 10.4173/mic.2020.4.1.

Kümper, S. et al. (2021). "DLR Visualization 2 Library - Graphical Environments for Virtual Commissioning". In: *14th International Modelica Conference, 20-24 September 2021*. Linköping University Electronic Press, pp. 197–204. DOI: 10.3384/ecp21181197.

Macenski, S. et al. (2022). "Robot Operating System 2: Design, architecture, and uses in the wild". In: *Science robotics* 7.66, eabm6074. DOI: 10.1126/scirobotics.abm6074.

Matei, I. et al. (2024). "Modelica supported automated design". In: *American Modelica Conference 2024, 14-16 October 2025, Storrs, CT, USA*. Linköping University Electronic Press, pp. 43–52. DOI: 10.3384/ECP20743.

Ramonat, M. et al. (2025). "Automated Generation of Simulation Models Based on Plant Engineering Data". In: *Workshop der ASIM Fachgruppen GMMS – Grundlagen und Methoden in Modellbildung und Simulation und STS – Simulation Technischer Systeme, 10-11 April 2025, Oberpfaffenhofen, Germany*. ARGESIM Reports. ASIM and ARGESIM, pp. 79–86. ISBN: 978-3-903347-66-3.

Santillan Martinez, G. et al. (2018). "Automatic Generation of a High-Fidelity Dynamic Thermal-Hydraulic Process Simulation Model From a 3D Plant Model". In: *IEEE Access* 6, pp. 45217–45232. DOI: 10.1109/ACCESS.2018.2865206.

Tola, D. and P. Corke (2023). "Understanding URDF: A Survey Based on User Experience". In: *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE), 26-30 August 2023, Auckland, New Zealand*. IEEE. DOI: 10.1109/CASE56687.2023.10260660.

Tola, D. and P. Corke (2024). "Understanding URDF: A Dataset and Analysis". In: *IEEE Robotics and Automation Letters* 9.5, pp. 4479–4486. DOI: 10.1109/LRA.2024.3381482.

Modelica Meets ASHRAE: Towards A Digital Standard for Building Control

Michael Wetter[1] Yan Chen[2] Karthik Devaprasad[2] Paul Ehrlich[3] Antoine Gautier[4] Jianjun Hu[1] Anand Prakash[1] Marco Pritoni[1]

[1]Lawrence Berkeley National Laboratory, Berkeley, CA, USA
[2]Pacific Northwest National Laboratory, Richland, WA, USA
[3]Building Intelligence Group, Afton, MN, USA
[4]Solamen, Laval, France

Abstract

Today's process for designing, specifying, installing and testing building HVAC control is not digitalized, leading to expensive manual workflows and missed operational performance. To digitalize the design-build-operate process for building HVAC control, the authors developed a process, associated tools and initiated the voluntary ASHRAE Standard 231P. This paper describes this process and tools, which are both based on Modelica. Standardization through a proposed voluntary ASHRAE Standard and the existing Modelica Language Specification provides a robust technology foundation for industry investment. It is based on declarative specification of the control logic, and allows reuse of existing technologies for open-loop and closed-loop control testing through coupling with an HVAC system or a whole building energy model. It supports control testing using MIL, SIL and HIL, and export of digital twins for operational support. The process and ASHRAE Standard 231P have been designed to accommodate existing Building Automation Systems product lines, while also enabling direct code generation such as by using FMI or eFMI. Control deployment can be digital or manual and conformance to the digital specification can be tested formally and programmatically at each step of the control delivery.

Keywords: Controls, Digitalization, Modelica Buildings Library, ASHRAE Standard 231P

1 Introduction

Design and implementation of control sequences for building and district energy systems has shown to be problematic. Barwig et al. (2002) showed that about one third of the errors in control systems for built-up HVAC systems, those that typically serve medium and large buildings that cover more than half of the floor area of the building stock, are attributed to programming errors. The problem of programming errors remains also today, as more recent literature shows (Crowe et al. 2020; Torabi et al. 2022). HVAC designers usually lack funding and skills needed to develop control logic. Rather than providing the control logic, they write the control intent, typically in the form of a 20 to 50 page untested Word document that has ambiguous formulations. The control provider then interprets this English language text and implements the control logic in software, often by reusing code from previous projects. This process has proven problematic even for the older HVAC systems, which have generally much simpler sequences than today's high performance control sequences. Modern controls are tasked with integrating heat pumps, chillers, and storage to not only meet temperature setpoints but also improve efficiency and shift energy use to periods with lower energy rates. The current control design and delivery process has not kept pace with digitalization. It remains highly manual, based on ambiguous specification, with minimal testing beyond spot checks during commissioning. While control providers have product lines that support communication standards, such as ASHRAE Standard 135 (BACnet, ASHRAE (2024)) or KNX, there is no standard for expressing the control logic.

To address these gaps, the authors have been developing a process and underlying tools for the digitalization of the controls design, delivery, implementation and commissioning. In support of this effort, the authors created a language for expressing control logic, called Control Description Language CDL (Wetter, Grahovac, and Hu 2018). Together with Modelica template models for HVAC systems and tools for configuring and translating CDL, CDL forms the foundation for control digitalization. CDL is defined as a small subset of Modelica, chosen in a way that allows translation to many legacy control languages, yet expressive enough to build configurable control sequences that allow for example specifying the type of sensors (e.g., a room CO_2 sensor), the type of equipment (e.g., water-based cooling coil, direct expansion cooling coil, presence of a heat recovery chiller etc.) and then automatically configure a control logic for the specified system.

Some of the authors also initiated an ASHRAE Standards Committee, and further developed CDL into the proposed ASHRAE Standard 231P[1].

This paper provides an overview of the CDL language,

[1]See https://www.ashrae.org/technical-resources/standards-and-guidelines

the control libraries that are being developed based on CDL, and the tools to configure controls and to translate CDL to target environments, which can be new or legacy Building Automation Systems (BAS) or Energy Information Management Systems (EMIS). As recent EMIS can also provide control capabilities (Pritoni et al. 2022), we collectively call them BAS below. The paper also describes verification of control logic implemented in a BAS to test conformance to the control specification that is expressed in CDL. Additionally, it highlights companies that have been incorporating CDL into their products and processes. Further information on earlier work related to this project can be found at `https://obc.lbl.gov`.

Assuming a widespread adoption of the digital control design-build-operate process presented here, the annual US electricity cost savings would be $8.2*B* and the natural gas savings $1.07*B* (CEC 2021).

2 Methodology

The development of the digital workflow started with the identification of key problems in today's building control design-build-commission-operate process, followed by stakeholder interviews and the development of use case and requirements. For industry adaption, the new workflow needs to reduce time and cost, address the lack of skills and budget among mechanical designers tasked with specifying controls, and ensure that the software solution is easy to support with existing control product lines. Additionally, it must be future-proof to enable more modern software solutions than those currently available in many automation product lines. Further, it has to enable rigorous sequence verification in a formal, programmatic way, which the authors believe is a core requirement that few control providers recognize today.

For this digital workflow, we developed the Control Description Language (CDL) (Wetter, Grahovac, and Hu 2018), which is a subset of Modelica. CDL prescribes basic blocks that are easy to support by legacy control systems, and it prescribes rules to compose such blocks hierarchically to implement control logic. The use of Modelica allows to perform simulations for design and verification of control and mechanical system operation coupled to a whole building or district energy model. It also allows code generation for future control product lines such as via the FMI or eFMI standard (Blochwitz et al. 2011; Lenord et al. 2021).

We validated CDL by creating HVAC control libraries, and using these libraries for closed-loop performance assessment with whole building energy simulation, using either Modelica or Spawn of EnergyPlus to simulate the building envelope model (Wetter, Ehrlich, et al. 2022; Zhang et al. 2022; Wetter, Benne, et al. 2023). Additionally, we developed a new workflow for digital control delivery, and we prototyped each step of the process, from design to code generation for WebCTRL, a commercial control product line from ALC (Automated Logic, a Carrier company) and verification of as-installed control logic in a real building (Wetter, Gautier, et al. 2019). See Wetter, Ehrlich, et al. (2022) for an overview.

To standardize the underlying language for control representation, in 2020, a formal application was submitted to ASHRAE to establish a standards committee, resulting in the proposed ASHRAE Standard 231P "CDL - A Control Description Language for Building Environmental Control Sequences." This standard aims to define a declarative graphical programming language for building environmental control sequences that is both human- and machine-readable, designed for specification, implementation via machine-to-machine translation, documentation, and simulation. The ASHRAE Standard 231P committee held its kick-off meeting in September 2020, and the proposed standard is expected to undergo a second, and potentially final, public review in 2025.

In the meantime, various control libraries and support tools have been implemented, and they are described in the following sections.

3 Workflows

Figure 1 shows a simplified flow chart of the digital workflow enabled by CDL. The figure is based on the more comprehensive workflow shown in Figure 2 in Wetter, Ehrlich, et al. (2022). Depending on the HVAC characteristics, energy code and project requirements, a mechanical engineer uses ctrl-flow (see Section 3.5), a web-based tool, to select and configure the HVAC system and control sequence through menu options. Then, the ctrl-flow software outputs the sequence specification. This consists of the English language sequence description, the digital control sequence conforming to ASHRAE Standard 231P, (essentially a hierarchical block-diagram in Modelica), and the semantic model conforming to ASHRAE Standard 223P. These steps are achieved using the modelica-json software described in Section 3.4.[2] A control provider then implements this control logic in a BAS. This can either be done manually using today's workflow, or preferably through software translation, which we prototyped and explained in Section 4.2 in Wetter, Ehrlich, et al. (2022). The control provider then instantiates the control logic in a controller and links inputs and outputs to physical devices or virtual control points. Regardless of whether the control logic was implemented through software translation or through manual programming, a commissioning agent then trends BAS inputs and outputs, and compares them against outputs computed by the CDL representation for the same control input and parameters. This verification test can be done using the funnel software [3], and prototype code for such verification is described in Section 4.3 in Wetter, Ehrlich, et al. (2022).

[2] modelica-json generates these outputs based on a configured Modelica model. The ability to export the configured Modelica model, as an additional output, is a planned feature for 2025.

[3] See `https://github.com/lbl-srg/funnel`.

DOI 10.3384/ecp218505

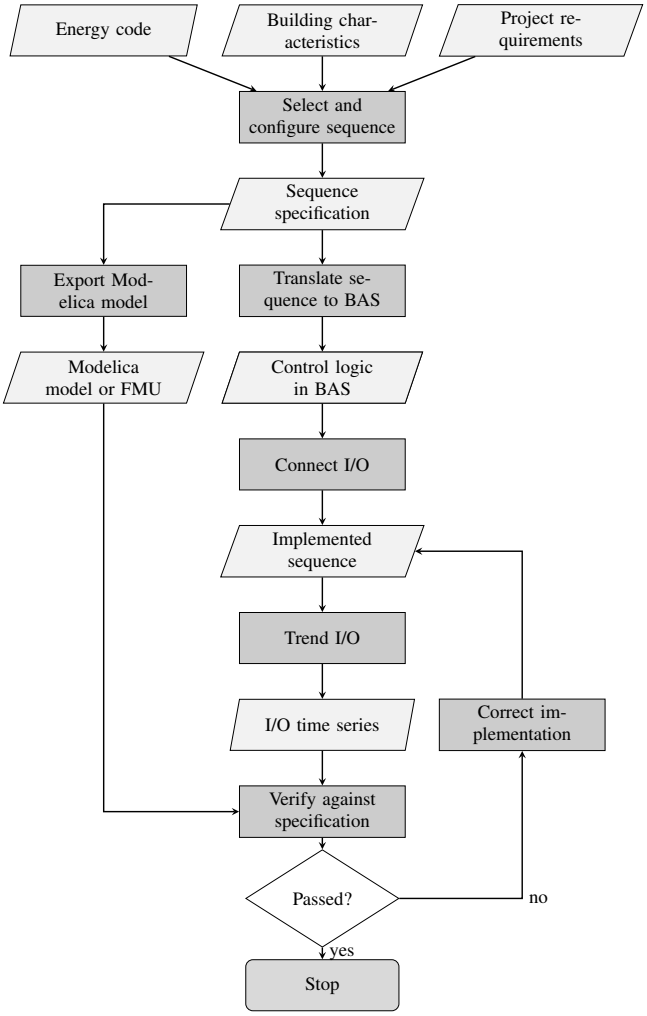


Figure 1. Workflow from control sequence configuration to export of a specification, implementation on a BAS and verification during commissioning. (Simplified representation based on Wetter, Ehrlich, et al. (2022).)

This final step of the workflow provides formal verification ensuring that the as-implemented control logic reproduces the control signal in accordance with the specification that was exported by the mechanical designer. Thus, we give the control vendor the option to use software translation, but don't require to do so, and regardless, we offer the ability to test the implemented control logic for compliance with the CDL specification.

While the above workflow is for an actual building, the above cited publication shows a variant that is applicable for control providers to conduct open-loop tests to verify that they implemented the control logic correctly.

3.1 CDL

CDL is a small subset of Modelica that is needed to implement block diagrams that are sufficient for use in Modelica templates (Gautier et al. 2023), yet simple to implement and modify by non-Modelica experts. CDL defines permissible data types (`Real`, `Integer`, `Boolean`, **`enumeration`**). It also defines immutable, so-called *Elementary Blocks*, such as the `Reals.Add` block that outputs the sum of two real-valued input signals. The language allows single inheritance blocks and the composition of blocks to form a *Composite Block* that may encapsulate a complex hierarchical control logic. It also allows for conditional connectors and instances, enabling the implementation of a room controller that may include, for example, an optional input and associated control logic for the room CO_2 concentration.

To reduce complexity of CDL, and because legacy BAS may not support advanced Modelica concepts, we excluded various Modelica constructs. Excluded constructs are, for example, `Clock`, **`inner`**/**`outer`**, physical ports such as a fluid port, and finite state machines. However, CDL has the concept of an *Extension Block* that allows to encapsulate any Modelica construct. Originally implemented to satisfy the requirement of calling compiled proprietary control logic, an `Extension Block` also allows, for example, the implementation of a finite state machine. ASHRAE Standard 231P requires each *Extension Block* to be exported to an FMU-ME 2.0 when exporting CDL to CXF, which is an exchange format described below. Control vendors are allowed to implement *Elementary Blocks* and *Extension Blocks* in their language of choice, as long as the mapping from inputs, parameters and states produces the same outputs. Thus, these blocks are defined through their inputs, outputs and the mapping from inputs and current states to outputs.

CDL also specifies the Modelica vendor annotation `__cdl`, which is used, among other things, to declare semantic information, or to specify whether an input needs to be hardwired or networked, such as through the BACnet protocol.

3.2 Standardization via ASHRAE Standard 231P

One critical step to digitalizing control delivery is to develop a standard which then provides a solid foundation for industry to adopt and invest in the technology. Approved standards are widely used by government and industry. We chose to develop an ASHRAE/ANSI standard, to facilitate potential future ISO recognition. The ASHRAE/ANSI process is consensus-based and involves various stakeholders. In 2020 the CDL team approached ASHRAE with a proposal and received approval to initiate a standards project. This effort was titled "ASHRAE 231P, Controls Description Language." The project committee consists of 30 members, including a diverse mix of control system manufacturers, HVAC control system designers, and researchers. The committee has been meeting regularly and reviewing and discussing the structure and format of the standard. This has resulted in discussions related to key differences between how control logic is handled in simulation and in building automation product lines. For example, Modelica distinguishes between real and integer data types while many BAS represent integers as integer-valued real data types. Additionally, many BAS lack mechanisms for event iteration, or for synchronizing simultaneous time events. The standards committee discussed and reached an agreement on what blocks should be supported as *Elementary Blocks*.

ASHRAE Standard 231P also specifies the Control eXchange Format (CXF), a representation of CDL in a JSON-LD format that is intended to be used as an intermediate format when importing CDL to a BAS, as JSON-LD is easier to parse than Modelica. For example, a control provider might utilize JSON-LD to import control logic from a design tool and deploy it to their commercial BAS for a particular project. While CDL has language constructs that are used to build library of sequences, CXF was designed to only represent a specifically configured logic. For example, CXF has no mechanism to conditionally remove a block and all its connections, as is provided by Modelica and allowed in CDL. CXF can be exported from CDL using the modelica-json software described below.

In 2024 the committee released an initial draft standard for public review. It is anticipated that a second review will occur in 2025 which, depending on feedback, could result in the publication of an ASHRAE/ANSI Standard in early 2026, which could then be submitted for ISO approval.

3.3 Sequence library and HVAC templates

Concurrently with developing CDL, the authors have created CDL compliant reference implementations of control sequences published in ASHRAE Guideline 36 (ASHRAE 2018), along with the accompanying HVAC system models.[4] These parallel developments created valuable feedback where practical implementation challenges directly informed the CDL specification. One example of such feedback is the introduction of the *Extension Block* concept which provides support for finite state machines (see Section 3.1). This addition helped address one of our main difficulties: using discrete event modeling to represent synchronous logic, as described hereafter.

In sequence documentation, the word "when" (as opposed to "if") typically signals such event-based patterns. Consider the following clauses that handle availability conditions in the staging logic of a plant: "Any unavailable stage is skipped during staging events. *When* the current stage *becomes* unavailable, the transition to the next higher available stage is triggered." We first implemented these clauses by merging them into a single state-based condition: "*If* a stage *is* unavailable, the transition to the next stage is triggered." Now, if the plant is in stage 1 while stage 2 is unavailable and a stage-up command is generated, a simple test shows that, in the model time domain, the plant effectively transitions directly to stage 3. However, issues appear in the event domain where an intermediate step is taken, during which stage 2 is enabled before being disabled at the next event. This intermediate zero-time event triggers all the staging sequence at stage 2, which is unwanted. Refactoring the implementation to be event-safe proved challenging when using only CDL *Elementary Blocks*, but was greatly simplified by using finite state machines. This also improved the block diagram readability, facilitating maintenance tasks. [5]

With our studies involving closed-loop simulations (Wetter, Ehrlich, et al. 2022; Zhang et al. 2022), we realized both the value and the difficulty of integrating equipment models and detailed controls into models that can be easily configured and parameterized. Depending on the system configuration and control options, the set of required actuators and sensors changes. The corresponding components and their connections must be updated consistently, while ensuring that the model remains well formulated for simulation. For example, lumping flow resistances is a good practice that reduces the size of nonlinear algebraic loops when modeling pressure-flow networks (Jorissen, Wetter, and Helsen 2015). But this approach becomes error-prone when the system configuration changes, e.g., when actuated isolation valves are removed, knowing that these are typically the components used to merge the fixed flow resistance of a chiller evaporator and condenser. Parameterizing hydronic system models brings further requirements, such as flow balanc-

[4] Both developments are released as part the Modelica Buildings Library, in the packages `Buildings.Controls.OBC.ASHRAE` and `Buildings.Templates`.

[5] The resulting *Extension Block* `StageIndex` in the Modelica Buildings Library shows the complexity of simply computing the current stage index of a multiple-unit system based on stage change commands. The discussions in the issues #3787, #3952 and #3966 further illustrate the iterations it took to find event-safe implementations for CDL *Elementary Blocks*.

DOI 10.3384/ecp218505

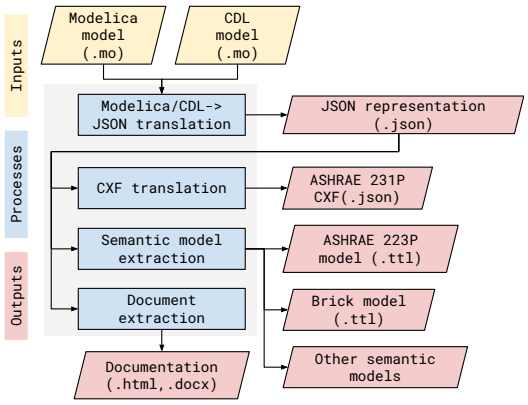


Figure 2. Inputs and outputs supported by modelica-json.

ing between unequally sized equipment in constant primary flow systems. This involves calculating the flow coefficient of balancing valves in case of constant speed pumps, or the pump speed (potentially at each plant stage) in case of variable speed pumps operated at fixed speed. Templates have been developed to programmatically address such tasks, enabling users to configure detailed closed-loop models by simply specifying top-level parameters. The templates handle all the complexity of model creation and parameterization, while embodying modeling best practices and making state-of-the-art control sequences broadly accessible. This is achieved exclusively with Modelica code, leveraging advanced class parameterization features of the language (Gautier et al. 2023).

The templates distributed with version 12.0.0 of the Modelica Buildings Library cover variable-air-volume systems and air-to-water heat pump plants. Templates for boiler and chiller plants are planned for release with the next version of the Library.

3.4 Modelica-json translator

Figure 2 shows modelica-json, a software that can be used to translate Modelica models into a JSON representation (Wetter, Hu, et al. 2021), and from there to generate various other outputs. The JSON representation is used to generate CXF, the representation of the control logic, and to generate a semantic model (see Section 3.4.2). As Modelica and CDL classes support the inclusion of semantic information (Fierro and Pauwels 2022), modelica-json allows the extraction of semantic models for ontologies such as Brick and ASHRAE 223P, as well as other schemas such as Project Haystack. Modelica-json is also used to generate documentation in html or Microsoft Word.

3.4.1 CDL-CXF translation

CXF is a semantic, graph-based representation of a CDL control logic serialized in JSON Linked Data (JSON-LD). Based on JSON-LD and using classes such as `S231:Block`, `S231:Connector` and `S231:Parameter`, and using relationships such as `S231:hasInput`, `S231:containsBlock` and `S231:isConnectedTo`, modelica-json constructs the CXF representation of the CDL control logic. Structurally the content of a logic in CDL and CXF are identical, in that both utilize the same *Elementary Blocks*, *Composite Blocks*, and *Extension Blocks* as well as `Constants`, `Parameters`, `InputConnectors` and `OutputConnectors`. There are, however, a few differences: First, for *Elementary Blocks* in CDL, their CXF representation does not include the implementation (equation section) of the particular block. Second, CXF allows translation tools to either retain array references and expressions present in CDL or to flatten the arrays and evaluate the expressions.

3.4.2 Semantic model extraction

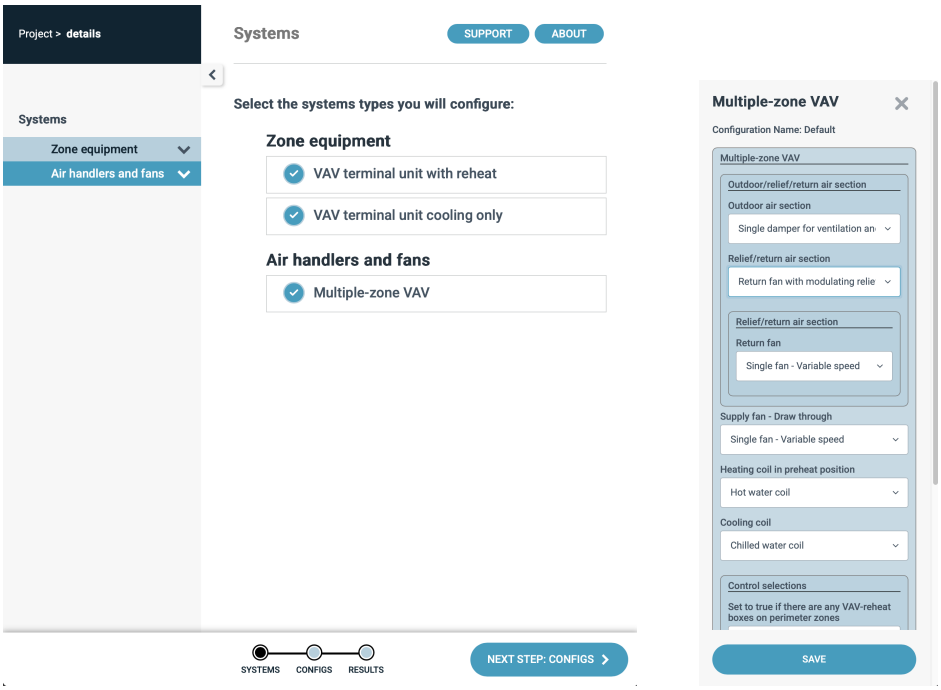
CDL supports the inclusion of semantic information (Fierro and Pauwels 2022) in a control logic using the `__cdl(semantic(...))` Modelica vendor annotation. While, as with other Modelica annotations, these annotations have no influence on the behavior of the control logic, they can be used to add information that is helpful for trending and communication, and for specifying semantic information. These items all support streamlining the configuration of control and communication, connecting or configuring energy analytics programs, and exporting natural language documentation.

Through the `__cdl(semantic(...))` annotation, semantic information can be added to blocks (both the class and the instance definitions), packages, parameters and input and output connectors. Controls developers can include information adhering to a particular semantic ontology or data schema using the `semanticLanguage` keyword. In this case, the included information must conform to the `semanticLanguage` and a supported format, such as `text/turtle`, `application/json` or `application/ld+json`. They can also add documentation in a natural language using the `naturalLanguage` keyword, with the documentation in the `text/plain` format. Similarly, semantic information can also be added to models developed with the Modelica Buildings Library, beyond CDL, by using the `__Buildings(semantic(...))` vendor annotation.

Modelica-json enables the export the semantic information as a separate file that can be shipped along with the CDL (or CXF) control logic or the Modelica model. For example, if a CDL control logic contains semantic shape requirements for the input and output points, described using ASHRAE Standard 223P and SHACL (W3C 2008), modelica-json can export these shapes for each of the connectors. These semantic shapes could then be used to integrate the sequences with BACnet communication points within a BAS.

3.5 ctrl-flow

Figure 3 shows ctrl-flow, a web-based application that aims to facilitate the adoption of the digital workflow by the HVAC designers. The tool is open-source and available at `https://ctrl-flow.lbl.gov`. It solicits user inputs to specify the characteristics of HVAC systems



(a) System selection. **(b)** Configuration panel.

Figure 3. ctrl-flow user interface.

and controls as shown in Figure 3, and outputs standardized specification documents. In its current state of development, the tool enables creating the control sequence documentation for variable-air-volume systems compliant with ASHRAE Guideline 36 (ASHRAE 2018). Ongoing projects are extending both the scope of supported systems and the generated content to include control schematics, point lists, equipment schedules and Modelica models of the specified systems.

At its foundation, ctrl-flow uses templates from the Modelica Buildings Library (see Section 3.3) and can accommodate any Modelica model that conforms to the guidelines available at `https://lbl-srg.github.io/modelica-buildings-templates`. On the server side (*Node Backend* in Figure 4) the tool invokes modelica-json and further processes the JSON representation of the template abstract syntax tree into an intermediate format (*ctrl-flow Schema*) that is more tractable by the React frontend. This schema is then populated by user selections and written back to the *Node Backend*, from where it is returned to the *Python Doc Pipeline*. The *Python Doc Pipeline* creates the sequence documentation using a DOCX template file, and then sends it to the *React Frontend* for user download.

Since templates often rely on advanced Modelica constructs, the tool required an interpreter layer capable of handling Modelica grammar's intricacies such as replaceable elements (instance or class) with optional constraining clauses, conditional declarations or composite bindings with `record` instances. Additionally, the frontend needed an integrated expression evaluation engine to process Boolean expressions used in user interface annotations that control the visibility of input fields.

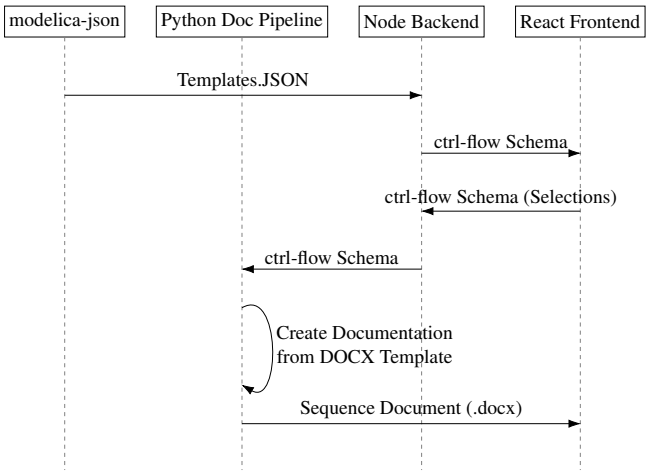


Figure 4. Data flow diagram of ctrl-flow.

In practice, this translates into handling an object of roughly 20 MB in size that gets regenerated at each user interaction while maintaining both responsiveness in the interface and simplicity in the user experience.

3.6 Import of CDL in BAS

Today's BAS generally do not accommodate generating C code and executing the compiled code, as is done, e.g., in embedded systems. Therefore, CDL was designed so that its mathematical mapping of inputs and states to outputs can be replicated by a BAS, and the declarative specification of the block diagram be translated to the programming

DOI
10.3384/ecp218505

language of the target environment. Hence, supporting CDL can be accomplished either through code generation, or through translation of CDL. The translation of CDL consists of mapping the CDL block diagrams to a representation in the target platform that given the same inputs and states, produces the same outputs. Our ongoing collaboration with various BAS providers indicate that only minor changes if any are needed in the target BAS. This requirement for compatibility with existing BAS product lines was the reason for excluding Modelica constructs such as `inner`/`outer`, `expandable connector` and `Modelica.Clocked`.

For a translation from CDL to a BAS, control providers seem to favor the JSON-LD representation rather the CDL representation, as it avoids having to implement a Modelica parser, although we encountered a company that started doing so. If one designs a new control product line, we recommend exploring the use of the eFMI Standard (Lenord et al. 2021) and the SSP standard (Modelica Association 2019) for using code in a BAS.

3.7 Verification

The proposed digital workflow supports the verification of the control logic relative to the original specification, which is expressed as a model in the CDL language. The verification compares the difference between the outputs of the controller minus the outputs computed by the specification, both using the same control inputs, parameters and constants.[6] The verification uses the funnel software to verify that the differences are within user-specified tolerances in time and in signal value. If all differences are within the tolerances, the tested control logic conforms to the specification.

Note that this verification process only tests the control logic implementation relative to the specification. It neither evaluates the suitability of the control intent as specified in the control specification, nor does it test the performance of the closed loop system or the proper operation of the HVAC components. Such functional tests and performance tests have been deliberately excluded from the control logic verification, and should be done separately.

We will now describe two distinct use cases, the first being the verification of a control logic implemented in a building system, and the second being the testing of a control logic as part of product development during which a controller can be run open loop. Both use cases are also described in more detail in Wetter, Gautier, et al. (2019) and Wetter, Ehrlich, et al. (2022)

3.7.1 Controller installed in a building system

If a controller is installed in a building system, it may not be possible to arbitrarily change the control inputs without detrimentally affecting the safe operating of the building system. Therefore, in this workflow, a commissioning agent can run the building system through acceptable operational scenarios and trend the inputs and outputs of the controller. Next, the CDL model is simulated with the same parameters and inputs, and its outputs are recorded. Using the funnel software, tests can be run to verify that the differences between trended and simulated outputs are within a user-set tolerance.

3.7.2 Open loop testing of a controller

Prior to installation in a building system, a controller could also be executed open loop to test its response to different sets of parameters and inputs. Therefore, in this workflow, a control technician executes the CDL model for a variety of parameters and inputs, and records the output. Next, the actual controller is executed for the same parameters and inputs, and its outputs are recorded. As in the previous use case, the outputs of the simulated and actual controller are compared.

4 Industry collaboration

The control industry has shown significant interest in developing and adopting standards for digitalization of control as described above. All of the major US BAS vendors have been actively participating in the creation of ASHRAE Standard 231P, with representatives being voting members of the committee. The authors have been collaborating with Carrier's Automated Logic to prototype a translator from CDL to the EIKON language, as described in Wetter, Ehrlich, et al. (2022), which is now being further developed. Additionally, another BAS vendor, Tridium, and two EMIS vendors, Skyfoundry and Normal inc, have recently begun developing parsers and translators from CDL or CXF to their product lines. Furthermore, the authors have been working with KTC, a controls and energy optimization company, to incorporate ASHRAE Standard 231P into their design-build-operate workflow via a translation to a PLC controller using IEC 61131-10 XML. Notably, Meta has been using the Modelica Buildings Library and CDL to design and operate hyperscale data centers and optimize their operation using Modelica-based digital twins (Rivalin et al. 2023).

5 Outlook

We expect that in 2025, the ASHRAE Standards 231P and 223P undergo a formal vote for approval. Once accepted, this will provide a robust foundation for BAS and EMIS providers for the digitalization of control and energy information system delivery. Through the ESTCP project "DoD Building Controls Design Tool"[7] that is led by the US Army Construction Engineering Research Laboratory, the authors are working on adopting the ctrl-flow software to the needs of DoD who owns around 550,000 buildings and facilities and thus provides an opportunity for scaled

[6] Typically, one also has to set initial states. However, many BAS do not allow setting initial states or recording them. In this case, the verification should start once initial transients disappeared.

[7] See `https://serdp-estcp.mil/projects/details/a964da95-b45e-404a-b7e1-d8dd6fcc246f/dod-building-controls-design-tool`.

deployment. The authors are also working with DOE and DoD/ESTCP in exporting Modelica template models for combined heat pump and chiller plants for buildings and district energy systems, and including them in ctrl-flow as part of a design guide for the robust, scalable deployment of such systems. In addition the authors are working on a California Energy Commission project to implement new demand-flexible control sequences in four California campuses using CDL in collaboration with two BAS and three EMIS providers.

6 Conclusion

Our proof of concept translation and various follow on work showed that a small subset of Modelica that is simple to use and support suffices for many building control representation in support of a digital design-deploy-verify process, with a notable exception of certain complex equipment staging sequences that are much simpler to implement in finite state machines, for which CDL provides a means for including such encapsulated Modelica constructs. With CDL, which is a Modelica subset, and its CXF representation, which is in JSON-LD, languages were developed that allows controls design, closed loop performance assessment with HVAC and building models in the loop, creation of template libraries of HVAC and associated control logic, and digital deployment to control product lines.

The next challenge is to scale its use in the buildings industry to make workflows cheaper, more robust and repeatable based on a highly digitalized workflow that allows formal validation and verification at every step along the design, build, and operate process. With the inherent higher control complexity that we see for energy efficient, grid-responsive and resilient systems, we believe such a new process is essential to control costs and meet the technical performance goals.

7 Data Availability

The CDL reference implementation, the control sequence libraries and the HVAC and control templates are available in the Modelica Buildings Library (`https://github.com/lbl-srg/modelica-buildings`). The modelica-json translator is available at `https://github.com/lbl-srg/modelica-json`. The guide for how to implement Modelica templates for use with ctrl-flow is available at `https://lbl-srg.github.io/modelica-buildings-templates`. The ctrl-flow source code is at `https://github.com/lbl-srg/ctrl-flow-dev` and the online web version is at `https://ctrl-flow.lbl.gov/`. A specification of CDL and CXF, together with accompanying process descriptions and publications, is available at `https://obc.lbl.gov/`. The ASHRAE Standards 231 and 223, once approved, will be available at `https://www.ashrae.org/technical-resources/standards-and-guidelines` (access to draft versions is restricted to committee members) and an unofficial specification of CDL together with other information is available at `https://obc.lbl.gov`.

Acknowledgments

This work was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Office of Building Technologies of the U.S. Department of Energy, under Contract No. DE-AC02-05CH11231, and under Contract No. DE-AC05-76RL01830.

References

ASHRAE (2018-06). *ASHRAE Guideline 36-2018 – High Performance Sequences of Operation for HVAC systems*. ASHRAE.

ASHRAE (2024). *ANSI/ASHRAE Standard 135-2024, BACnet, A Data Communication Protocol for Building Automation and Control Networks*.

Barwig, Floyd E. et al. (2002-08). "The National Building Controls Information Program". In: Summer Study on Energy Efficiency in Buildings. ACEEE. Pacific Grove, CA.

Blochwitz, T. et al. (2011-03). "The Functional Mockup Interface for Tool independent Exchange of Simulation Models". In: *Proc. of the 8-th International Modelica Conference*. Modelica Association. Dresden, Germany. DOI: 10.3384/ecp11063105.

CEC (2021-02). *Open Building Control*. Tech. rep. CEC 500-2021-012. Sacramento, CA: California Energy Commission. URL: https://www.energy.ca.gov/sites/default/files/2021-05/CEC-500-2021-012.pdf.

Crowe, Eliot et al. (2020). "Building commissioning costs and savings across three decades and 1500 North American buildings". In: *Energy and Buildings* 227, p. 110408. ISSN: 0378-7788. DOI: 10.1016/j.enbuild.2020.110408. URL: https://doi.org/10.1016/j.enbuild.2020.110408.

Fierro, Gabe and Pieter Pauwels (2022). "Survey of metadata schemas for datadriven smart buildings (Annex 81)". In: URL: https://annex81.iea-ebc.org/Data/publications/IEA%20Annex%2081%20Survey%20of%20Metadata%20Schemas.pdf.

Gautier, Antoine et al. (2023-12). "HVAC and control templates for the Modelica Buildings Library". In: *Proceedings of the 15th International Modelica Conference 2023*. Ed. by Dirk Müller, Antonello Monti, and Andrea Benigni. Linköping Electronic Conf. Proc. 204. Aachen, Germany: Modelica Association and Linköping University Electronic Press, pp. 217–227. ISBN: 978-91-8075-505-4. DOI: 10.3384/ecp204217. URL: https://doi.org/10.3384/ecp204217.

Jorissen, Filip, Michael Wetter, and Lieve Helsen (2015-09). "Simulation speed analysis and improvements of Modelica models for building energy simulation". In: *Proceedings of the 11th International Modelica Conference 2015*. Ed. by Peter Fritzson and Hilding Elmqvist. Linköping Electronic Conf. Proc. 118. Versailles, France: Modelica Association and Linköping University Electronic Press, pp. 59–69. ISBN: 978-91-7685-955-1. DOI: 10.3384/ecp1511859. URL: https://doi.org/10.3384/ecp1511859.

Lenord, Oliver et al. (2021-09). "eFMI: An open standard for physical models in embedded software". In: *Proc. of the 14th International Modelica Conference*. Ed. by Martin Sjölund et

al. Linköping Electronic Conf. Proc. 181. Linköping, Sweden: Modelica Association and Linköping University Electronic Press, pp. 57–71. ISBN: 978-91-7929-027-6. DOI: 10.3384/ecp2118157. URL: https://doi.org/10.3384/ecp2118157.

Pritoni, Marco et al. (2022). "From fault-detection to automated fault correction: a field study". In: *Building and Environment* 214. DOI: 10.1016/j.buildenv.2022.108900. URL: https://doi.org/10.1016/j.buildenv.2022.108900.

Rivalin, Lisa et al. (2023-05). *Predicting Temperature and Differential Pressure in Data Centers Using Physical Modeling*. Tech. rep. Meta Platforms, Menlo Park, CA. URL: https://research.facebook.com/file/752832263209106/IAQVEC-2023_full-paper.pdf.

Modelica Association (2019-03). *System Structure and Parameterization*. Tech. rep. URL: https://ssp-standard.org/.

Torabi, Narges et al. (2022). "Common human errors in design, installation, and operation of VAV AHU control systems - A review and a practitioner interview". In: *Building and Environment* 221, p. 109333. ISSN: 0360-1323. DOI: 10.1016/j.buildenv.2022.109333. URL: https://doi.org/10.1016/j.buildenv.2022.109333.

W3C (2008). *SPARQL 1.1 overview*. en. https://www.w3.org/TR/sparql11-overview/. Accessed: 2025-3-5.

Wetter, Michael, Kyle Benne, et al. (2023). "Spawn: coupling Modelica Buildings Library and EnergyPlus to enable new energy system and control applications". In: *Journal of Building Performance Simulation*, pp. 1–19. DOI: 10.1080/19401493.2023.2266414. URL: https://doi.org/10.1080/19401493.2023.2266414.

Wetter, Michael, Paul Ehrlich, et al. (2022-01). "OpenBuildingControl: Digitizing the control delivery from building energy modeling to specification, implementation and formal verification". In: *Energy* 238.A. DOI: 10.1016/j.energy.2021.121501. URL: https://doi.org/10.1016/j.energy.2021.121501.

Wetter, Michael, Antoine Gautier, et al. (2019-09). "Verification of Control Sequences within OpenBuildingControl". In: *16-th IBPSA Conference*. International Building Performance Simulation Association. Rome, Italy, pp. 885–892. DOI: 10.26868/25222708.2019.210722. URL: http://simulationresearch.lbl.gov/wetter/download/2019-ibpsa-OpenBuildingControl.pdf.

Wetter, Michael, Milica Grahovac, and Jianjun Hu (2018-08). "Control Description Language". In: *1st American Modelica Conference*. Cambridge, MA, USA. DOI: 10.3384/ecp1815417. URL: https://doi.org/10.3384/ecp1815417.

Wetter, Michael, Jianjun Hu, et al. (2021-09). "Modelica-json: Transforming energy models to digitize the control delivery process". In: *17-th IBPSA Conference*. Intern. Building Performance Simulation Assoc. Brugge, Belgium, pp. 1–8. DOI: 10.26868/25222708.2021.30141. URL: https://doi.org/10.26868/25222708.2021.30141.

Zhang, Kun et al. (2022-01). "Estimating ASHRAE Guideline 36 Energy Savings for Multi-zone Variable Air Volume Systems Using Spawn of EnergyPlus". In: *Journal of Building Performance Simulation* (2), pp. 215–236. DOI: 10.1080/19401493.2021.2021286. URL: https://doi.org/10.1080/19401493.2021.2021286.

DOI
10.3384/ecp218505

CDL-PLC translator: From Modelica HVAC control design to IEC 61131 PLC implementation

Karl Walther[1,2] Michael Wetter[3] Anand Prakash[3] Jianjun Hu[3]

[1]Department of Mechanical Engineering, KU Leuven, Leuven, Belgium, karl.walther@kuleuven.be
[2]School of Architecture and Civil Engineering, University of Wuppertal, Wuppertal, Germany
[3]Lawrence Berkeley National Laboratory, Berkeley, CA, USA

Abstract

This paper presents the CDL-PLC translator, a tool to convert control sequences for building heating, ventilation, and air conditioning (HVAC) systems expressed in a subset of Modelica called Control Description Language (CDL) which is undergoing standardization via the voluntary ASHRAE Standard 231P into the IEC 61131-10 XML format. Such a translation is a step in a model-based design workflow and contributes to digitalizing the control delivery process in the building sector. The translation into the IEC 61131-10 XML is the last step in the implementation of control logic developed in CDL on programmable logic controllers (PLCs) standardized in IEC 61131. The paper presents the details of the translator, an example for the validation of a CDL block vs. the corresponding IEC block, and a practical application example for the translation of a control sequence for a hybrid heat pump plant from a case study neighborhood in Belgium. Practical applications, use cases, and future developments are discussed in the context of building industry processes.

Keywords: HVAC systems, Control Description Language, PLCs, Model-based Design

1 Introduction

1.1 Motivation

The building sector is facing major challenges to improve energy efficiency and load flexibility. In order to increase the share of renewable and residual energy sources, new supply systems will be installed in new and existing buildings the next years. The integration of intermittent renewable energies raises the complexity of design and operation as modern installations often use multiple technologies, such as heat pumps, solar systems and storage in order to meet new requirements for buildings to shift supply and demand and to interact with electrical grid signals. Optimizing controls of air handling units and room-side emission systems is another strategy to increase the efficiency of new and existing HVAC systems in buildings. To perform these engineering tasks in building practice, the methods, tools, and procedures for the development of HVAC control sequences come into focus.

1.2 Industry practice and related problems

In the buildings industry, the initial development and the final implementation of HVAC system controls are separated between design engineers and installation contractors (see 'Building practice' in Figure 1). This is a structural difference to, for example, the manufacturing or automotive industry. The design and the execution phases are separated by the tendering and award procedure.

The design and the development of HVAC control sequences is largely based on static considerations in single operation modes, rules of thumb, and often based on previous projects. Control sequences are documented using graphical control schemes and textual so-called 'functional descriptions' or 'sequence specifications', for example based on VDI 3814-4.3:2022 (Verein Deutscher Ingenieure 2022). These analog documents are then manually interpreted by the control provider who develops control logic for the building automation system. This practice is highly susceptible to human error (Barwig et al. 2002; Torabi et al. 2022), and leads to several well-documented problems, such as inappropriately parameterized control sequences (Crowe et al. 2020; Raftery et al. 2024) and a gap between predicted and measured performance (Wilde 2014). An in-depth revision of the control sequences in the control code, which goes beyond parameter optimization, requires qualified engineers and is therefore costly.

1.3 Model-based HVAC control design

Model-based design is an approach to master the increasing complexity of future HVAC systems and their controls through an increased level of detail and increased quality in the design phase. Based on regulatory or user-defined requirements for the energy and comfort performance, a model-based and digitalized workflow generally includes three parts (see dashed boxes in Figure 1):

1. A Building Performance Simulation (BPS) environment for the integrated simulation of the building envelope, HVAC systems, and their controls, and the verification against performance requirements.

2. A digital exchange format for the lossless and unambiguous documentation and transfer of control sequences.

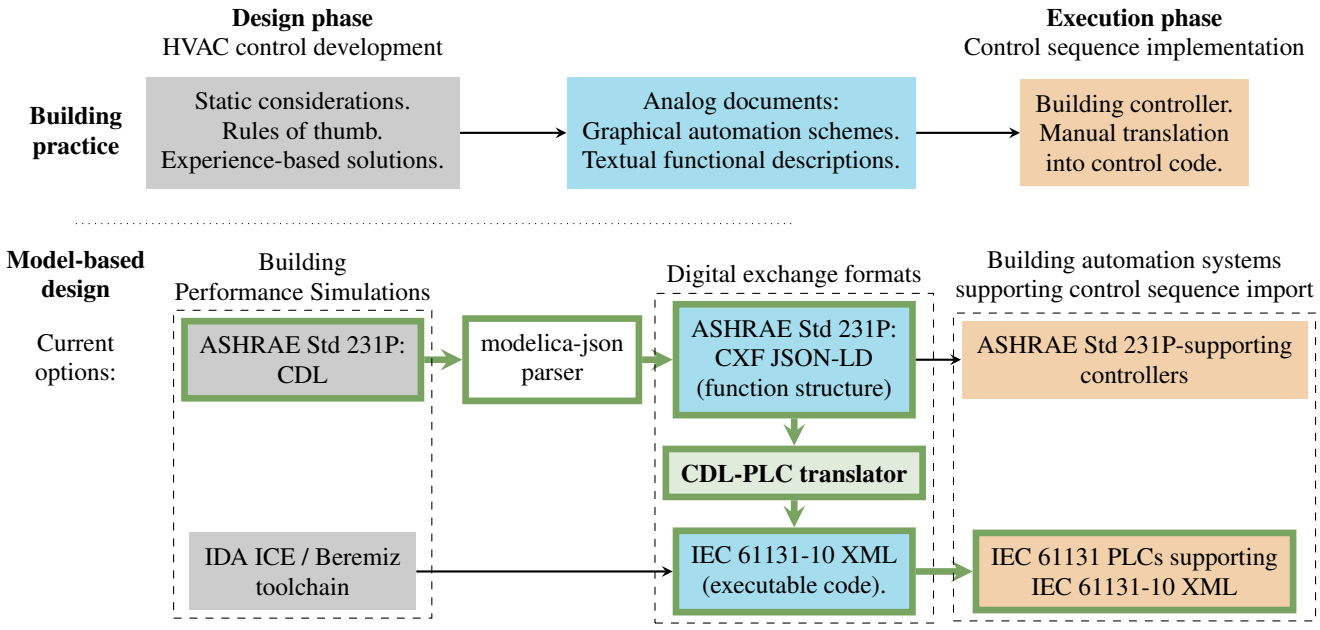


Figure 1. CDL-PLC translator in the context of development and implementation of HVAC controls in building practice, model-based design, and the workflow described in this study (green label). Note that from CDL to CXF, there is an alternate path via the ctrl-flow software[1] that does not require a building performance simulation. This path is not shown here as it does not include performance evaluation as is typical in model-based design.

3. For a seamless and efficient implementation, a building automation system (BAS) supporting the import of control sequences is needed.

Model-based design of HVAC controls is a new approach in the building industry, and generally applicable to all types of buildings. However, due to the effort required for model development, complex and individual HVAC systems in commercial buildings and supply systems in urban districts and neighborhoods would be typical applications. While model-based design is generally independent of the control approach, the toolchains discussed in this paper use conventional Rule-Based Controls (RBCs), which are still the standard in the building sector.

In the literature, two approaches can be distinguished that are presented in the following two sections.

IEC 61131-3 code simulation in IDA ICE The first option (see bottom row in Figure 1) uses the IEC 61131-3 code simulation in the commercial equation-based BPS software IDA ICE, the IEC 61131-10 XML standard as digital exchange format, and IEC 61131 PLCs as target system. The idea behind this toolchain is to simulate the behavior of real controllers in an equation-based simulation environment with a variable step size solver, such as IDA ICE (Sahlin, Bring, and Eriksson 2009). This toolchain was initially developed for Software-In-the-Loop (SIL) testing of PLC programs during the commissioning phase (Sahlin, Skogqvist, and Högberg 2018), however, it can also be used for the development of HVAC controls during the design phase, as demonstrated in a full-scale end-to-end application for air handling units (AHUs) in a factory building (Walther 2025).

IEC 61131-3 (International Electrotechnical Commission 2003) is a standard for the programming of controllers standardized in IEC 61131, called PLCs. IEC 61131 PLCs are commonly used for building automation in Europe. PLC programs are organized in program organization units (POUs) that can be a type or an instance of functions, function blocks, or programs. IEC 61131-3 defines 5 languages, two of which are textual (Structured Text (ST), Instruction List (IL)) and three of which are graphical (Function Block Diagram (FBD), ST, IL), which can be different for each POU. In chapter 2.5.1.5. of IEC 61131-3, standard functions for arithmetic, numerical, type conversion operations etc. are defined. For a technology overview and suggestions for future developments, e.g. to achieve deterministic behavior, we refer to Sehr et al. (2021).

In the IDA ICE toolchain, an XML file standardized in IEC 61131-10 is used for the digital delivery of control sequences from the design to the execution phase. The first version of this XML format was published in 2005 under the umbrella of the PLCopen organization (Marcos et al. 2009). IEC 61131-10 XML was originally designed to exchange control sequences between different PLC Integrated Development Environments (IDEs). Some PLC IDEs support the IEC 61131-10 XML import, for example Codesys, TwinCAT (Beckhoff), or e!cockpit (Wago). However, due to the above mentioned construction-specific workflow procedures (no code development in the design phase, normatively prescribed graphical and textual exchange formats), it is not used in

[1] See `https://ctrl-flow.lbl.gov`.

DOI 10.3384/ecp218515

the building sector today.

Modelica Control Description Language Within the OpenBuildingControl (OBC) project, a new process and supporting tools was developed to digitalize specification, implementation, and verification of HVAC control sequences (Wetter, Ehrlich, et al. 2022) (see middle row in Figure 1). The central part of CDL[2] is a library with standardized function blocks typically used for building applications, as a small subset of Modelica (Wetter, Grahovac, and Hu 2018a). This library alongside with rules for how to compose control logic in a hierarchical way was first presented in (Wetter, Grahovac, and Hu 2018b) and it is currently being standardized in ASHRAE Standard 231P.

To transfer HVAC control sequences defined in Modelica, multiple options are possible.[3] In principle, the (already existing) Functional Mock-up Interface (FMI) standard can be used, however, today's Building Automation Systems mostly do not support the direct execution of Functional Mock-up Units (FMUs) / C code, and they do not support editing Modelica models and regenerating FMUs if the control logic need to be changed after installation. To address this issue, the modelica-json translator[4] was developed to parse CDL sequences into an intermediate JSON file format, called Control Exchange Format (CXF) (Wetter, Hu, et al. 2021). With such an intermediate format, the translation of control sequences developed in CDL into product-line specific languages in general is performed in two steps:

1. *Parsing* control sequences in CDL into a JSON CXF using the open-source modelica-json translator. This step is fully covered by the CDL toolchain (open-source).

2. *Translation* of the CDL JSON into the product-line specific language. If the target system uses proprietary language, this step has to be performed by the controller manufacturer.

CDL vs. IEC 61131 The previously described options have introduced two standards, IEC 61131 and CDL (ASHRAE Standard 231P). Different areas of application are briefly contrasted below to underline that the two standards are not mutually exclusive.

IEC 61131, on the one hand, is a standard for controllers. It standardizes a computational model, programming languages, interfaces, exchange formats, etc. The IEC 61131 standard is only adopted by some manufacturers, while other manufacturers use proprietary control implementations. The crucial step that influences the actual building performance in operation most is, however, the early design phase when control sequences have to be designed in the building context, and when the actual control vendor might not yet be decided.[5] CDL, on the other hand, is specifically designed to allow simulation of control sequences together with HVAC systems in the building context, and subsequent translation to a Building Automation System. CDL is product-neutral and addresses all downstream controller types, including those that follow IEC 61131.

A further crucial aspect is that CDL provides a library of standardized function blocks. Such a library is missing in the PLC-domain where usually each vendor provides proprietary libraries. These vendor-specific libraries have several drawbacks: (a) they are typically proprietary and only documented textually, (b) they are not standardized and vary from vendor to vendor which locks customers into specific systems, and (c), most importantly, they only come into play during the programming of controllers during the execution phase and are therefore inevitably differ from what mechanical engineers have drafted during the design stage.

1.4 Research gap and contribution

In the context of CDL, current efforts focus on the standardization of control blocks and the exchange formats, such as the CXF JSON-LD, in ASHRAE Standard 231P (first step, 'parsing', described in 'Modelica-based Control Description Language' above).

This study presents the CDL-PLC translator to address the final step of transferring CDL sequences to building controllers following IEC 61131. The translator converts the CXF JSON into the IEC 61131-10 XML, and as such link CDL and the PLC-domain (see green path in Figure 1). The particular benefit maximizing the impact of such a translator is that CDL is linked to IEC 61131 which is followed by several automation manufacturers.

The translation of the CDL CXF into the IEC 61131-10 XML is associated with two main challenges:

1. The *translation of the structure* of the CXF JSON-LD into the target structure, here the IEC 61131-10 XML.

2. The *implementation of control functions* defined in CDL blocks into the language of the target system, here control code according to IEC 61131-3.

The remainder of this paper is organized as follows: First, the CDL-PLC translator is presented in section 2. In section 3, the translator is applied to a control sequence from a real project to demonstrate the functionality in a practical case. We conclude with a discussion on industry transformation and future work.

2 CDL-PLC translator

In this section, first, the general approach of the CDL-PLC translator is presented, then, source and target formats are

[2] https://obc.lbl.gov/specification/cdl.html

[3] https://obc.lbl.gov/specification/codeGeneration.html

[4] https://github.com/lbl-srg/modelica-json

[5] This applies for the case of new buildings. In existing buildings, the implemented BAS might or might not follow IEC 61131.

described, and, finally, the program architecture and the features are described.

2.1 Basics

General approach The general approach of the CDL-PLC translator is to convert the information from the CDL CXF JSON-LD format to the IEC 61131-10 XML format. The translator directly maps CDL blocks to IEC blocks, and the output IEC 61131-10 XML uses the graphical FBD language. This enables a direct and visual comparison of the graphical FBD representations in Modelica and IEC 61131-3. The translator is implemented in Python. For all PLC-related tasks within this study, the open-source PLC IDE Beremiz[6] was used.

Workflow and use cases The CDL-PLC translator is meant to be used either by design engineers or by executing contractors depending on how the control delivery is organized. During the execution phase, automation contractors tasked with the programming of IEC 61131 PLCs then import the IEC 61131-10 XML into the product-line specific PLC IDE.

The translator addresses two building-specific use cases:

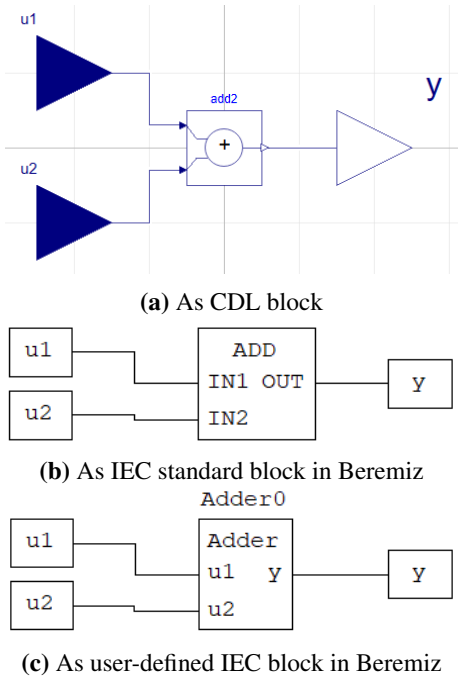
1. The development of control sequences in *new buildings*. In this case, the option to import the IEC 61131-10 XML must be specified in the tender. If a BAS already supports the FMI standard, importing an FMU might be a more straightforward option.

2. The revision of control sequences in *existing buildings*, where the existing BAS is kept, and where an existing HVAC system is either kept or replaced within a renovation. Only BAS with PLCs following IEC 61131 and IDEs supporting the import of the IEC 61131-10 XML are discussed below. As new PLC product lines might directly support the FMI standard (see section 4), such a redesign of existing legacy PLC-based BAS is a likely application for the CDL-PLC translator.

Development status The CDL-PLC translator is currently at a prototypical stage as a proof of concept that CXF can be translated into IEC 61131-10 XML. The translator supports the necessary basic features such as mapping and connecting function blocks. The translator currently supports a small subset of CDL elementary blocks, mainly those needed for the application example presented in section 3. However, as the core features are covered, extending the translator to a fully-fledged software is straightforward.

In its current form, the translator works only in the CDL-to-PLC direction. The PLC-to-Modelica direction may be possible for some control logic and would be beneficial, for example, in case executing contractors have made changes to the PLC code that have to be verified in Modelica, or when operational problems are easier to be addressed using simulation. However, whether a translation from a Building Automation System to CDL is possible is case dependent, as Building Automation Systems allow for formulations that are outside of the allowed formulation of Modelica or its CDL subset. For example, Wetter, Chen, et al. (2023) shows some differences between how control logic is implemented in building automation systems and in Modelica.

[6] https://beremiz.org/



(a) As CDL block

(b) As IEC standard block in Beremiz

(c) As user-defined IEC block in Beremiz

Figure 2. Control sequence with an adder as FBD

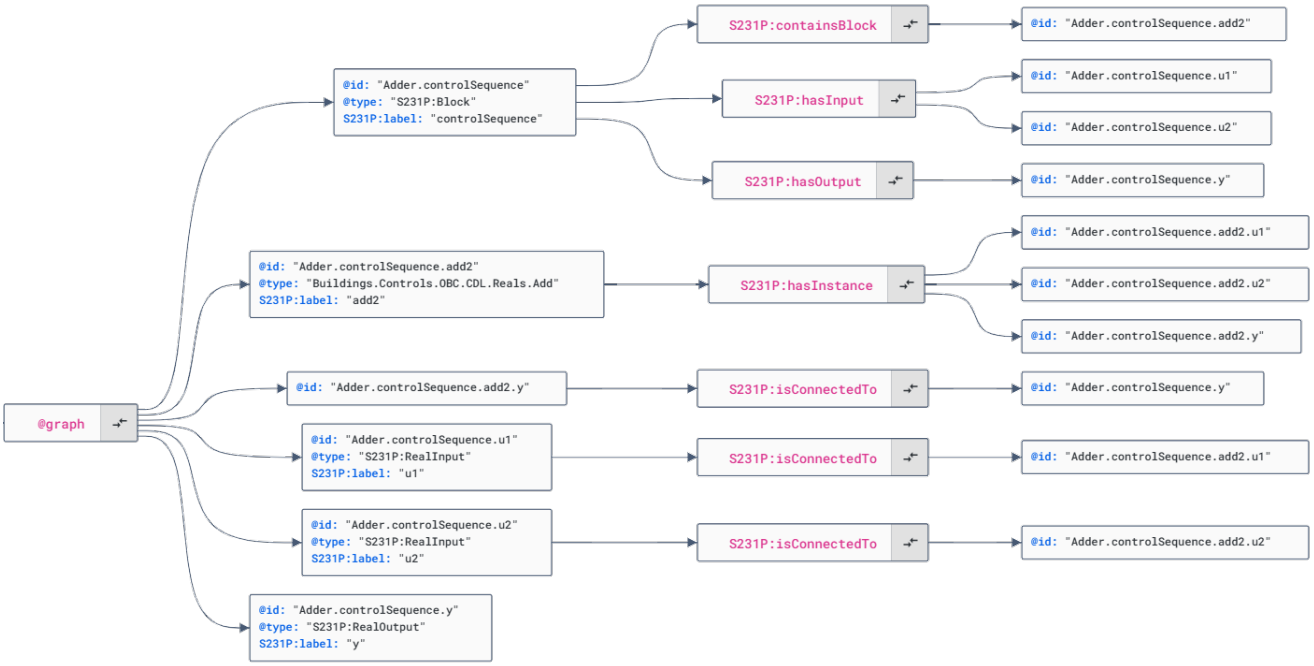
2.2 Source and target formats

In the following two sections, the CXF JSON-LD source format and the IEC 61131-10 XML target format are described in order to outline differences and the functional prerequisites for the translator. The description is done for a simple control sequence with inputs `u1` and `u2`, an adder block, and output `y` as shown in Figure 2.

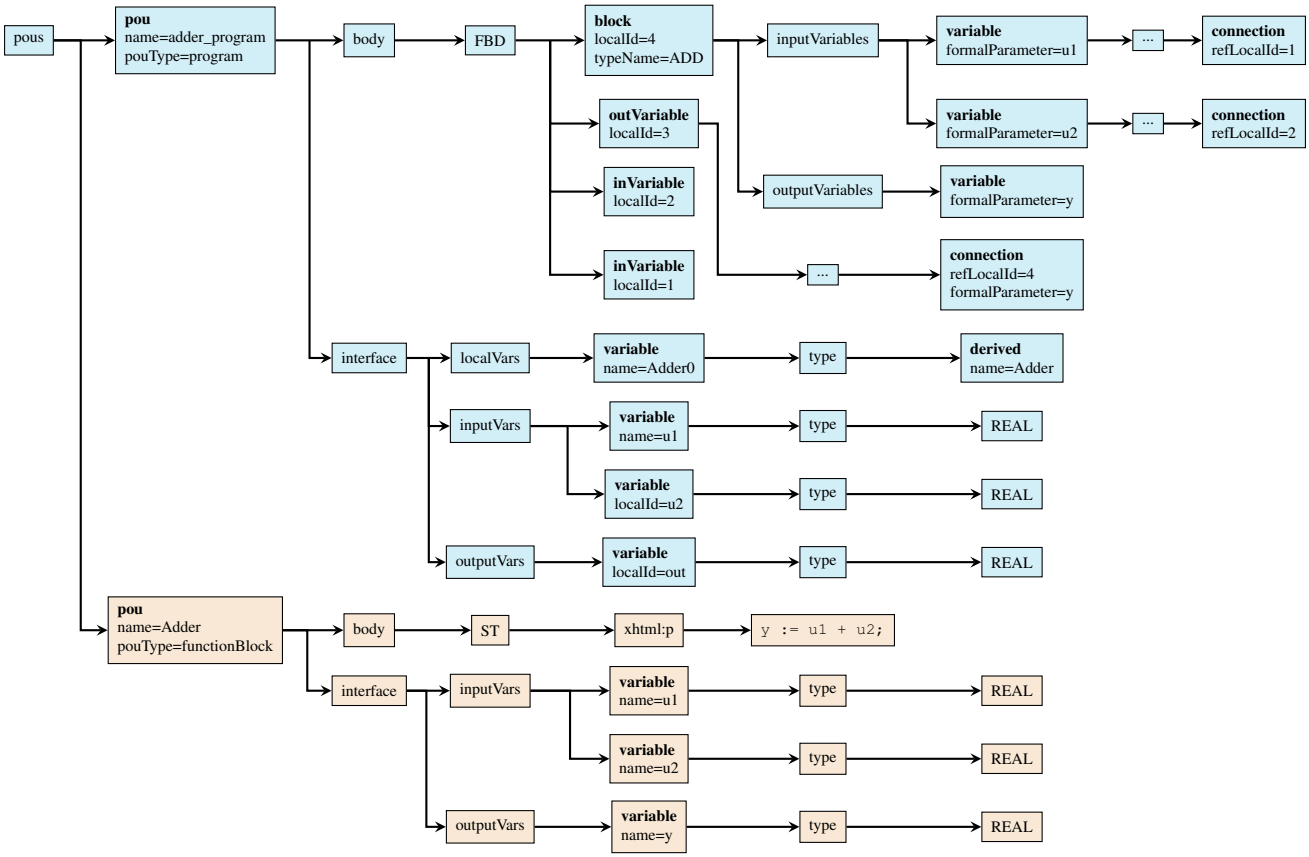
2.2.1 CXF JSON-LD

In CXF, nodes are separated depending on whether they represent composite or extended blocks, inputs (`Real`, `Integer`, or `Boolean`), outputs (`Real`, `Integer`, or `Boolean`), elementary blocks, parameter definitions, parameter assignments, or connections. Figure 2a depicts a simple control sequence with the instance `add1` of `Buildings.Controls.OBC.CDL.Reals.Add`. The visualization in Figure 3a shows the graph objects: the control sequence itself (`@id: "Adder.controlSequence"`), inputs, outputs, the adder block instance, and the output `y` of `add1`.

DOI
10.3384/ecp218515



(a) CXF JSON-LD



(b) IEC 61131-10 XML. Top branch (blue): IEC Program. Bottom branch (orange): IEC function block declaration.

Figure 3. Visualized graphs of the example from Figure 2 (in both visualizations, information, such as for the graphical rendering, is removed to improve readability).

2.2.2 IEC 61131-10 XML

Figure 2b depicts the control sequence as FBD in Beremiz, Figure 3b depicts the structure of the corresponding IEC 61131-10 XML. In Figure 3b, for example, the instance `adder_program` is of `pouType=program` and the instance `Adder` is of `pouType=functionBlock`. While the instance `program` uses the `FBD` language (IEC Function Block Diagram), the instance `adder` is expressed in `ST` (IEC Structured Text). Figure 2b and Figure 2c also illustrate the difference between an IEC 61131-3 standard adder (see Figure 2b) and a user-defined function blocks (see Figure 2c, where the output in the `Adder` class is defined as `y := u1 + u2;`). Connections between block instances and inputs and outputs (IOs) are described in an *input-is-linked-to-output* form. IDs assigned to each block instance, each input variable, and each output variable (see `localId` attribute), are used as identifiers for connections.

2.3 Program flow

The translation is executed in two steps that are explained in the following two sections.

2.3.1 Information extraction and conversion

In a first step, the information included in the CXF JSON-LD is extracted and converted into Python objects. According to the structure of the IEC XML, the objects are grouped as follows:

- Inputs
- Outputs
- Parameters (see next paragraph)
- Blocks (with/without direct CDL-IEC equivalent separated, see next paragraph)
- Composite blocks

Handling of parameters IEC 61131-3 does not support the concept of parameters editable per function block instance as in Modelica or its CDL subset. As a workaround, CDL parameters are converted into program inputs assigned to the corresponding function block instance (see application example in subsection 3.3). The translator also supports global parameters that can be propagated to one or more block instances. An option to avoid additional program inputs would be using IEC 61131-3 *local variables*. However, such *local variables* are fixed per class declaration rather than class instance, which would require creating different class declarations for each set of parameters.

Mapping of CDL elementary blocks and validation CDL blocks are mapped to the equivalent blocks in PLC code as follows:

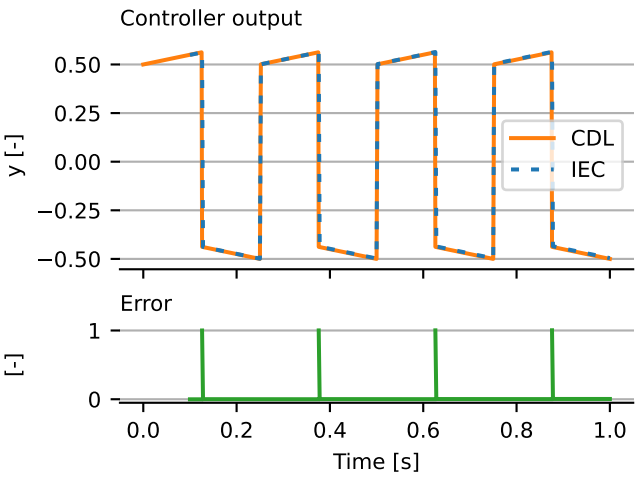


Figure 4. Validation of the `Reals.PID` CDL block vs. the corresponding block translated into IEC 61131-3 code.

- If possible, CDL blocks are directly mapped to the corresponding *IEC standard blocks* (see subsubsection 2.2.2). For example, `Logical.And` (CDL) is mapped to `AND` (IEC 61131-3).
- If a direct mapping is not possible, CDL function blocks are *individually translated* into IEC code and stored as XML snippets (see for example `Adder` function block class in Figure 3b).

An approach to facilitate the translation of each block individually would be using PLC function blocks from the open-source OSCAT library[7]. However, OSCAT blocks often do not fit exactly to the CDL definitions. OSCAT blocks are, however, used to process the PLC time (see section 'Different implementations of IEC 61131-3'), and for the integration block within the PID block.

To check whether the CDL elementary blocks have been correctly translated into the target language, the CDL toolchain offers validation packages. Figure 4 shows the result of such a validation for the `Reals.PID` block with a simulation time of 1 s. For the validation, a pulse input for the setpoint with a period time of 250 ms is prescribed on the PI controller setpoint u_s (fixed measurement input $u_m = 0.5$, proportional gain $k = 1$, time constant of the integrator block $T_i = 1$. The controller output of the IEC block matches with the CDL trajectory with a constant error of $e = 0.004$. The peaks occur because the IEC controller (cycle time 2 ms) switches one tick later than the CDL controller.

Translation of topology While connections are an attribute of output connectors in the CXF, they are an attribute of inputs in the IEC XML. The translator performs the necessary reassignment.

[7] `http://www.oscat.de`

DOI 10.3384/ecp218515

Listing 1. XML template for a function block instance

```
<block localId="{{ values.localId }}"
    typeName="{{ values.ClassName}}"
    instanceName="{{ key }}"
    executionOrderId="0" width="{{ values.
    width }}" height="{{ values.height }}">
        <position x="{{ values.x_absolute
            }}" y="{{ values.y_absolute }}"
            />
        <inputVariables>
                {% for key, value in values
                    .inputs.items() %}{%
                    include "xml_templates/
                    structure/
                    FbdBlockInstanceInVar.
                    xml" %}{% endfor %}
        </inputVariables>
        <inOutVariables/>
        <outputVariables>
                {% for key, value in values
                    .outputs.items() %}{%
                    include "xml_templates/
                    structure/
                    FbdBlockInstanceOutVar.
                    xml" %}{% endfor %}
        </outputVariables>
</block>
```

Different implementations of IEC 61131-3 Despite the standardization in IEC 61131, there are differences in the implementation of the standard between the control manufacturers, e.g., in the way the control time is updated, e.g. for the integrator in PID controllers. This requires that different functions / function blocks have to be mapped depending on the specific IEC 61131 target system. The CDL-PLC translator is currently outputting an IEC XML that is tested in the open-source PLC IDE Beremiz. Further adaptations to the XML might be required depending on the target IDE (not investigated in this study). In the application described in Walther (2025) an XML created with Beremiz is imported into the commercial PLC IDE e!cockpit.

2.3.2 Rendering

In a second step, the information collected in the Python objects is rendered in a IEC 61131-10 XML. For this rendering, the translator iterates over each level of the XML tree, and fills the information included in the corresponding Python objects in predefined XML templates. As an example, Listing 1 shows the template for the function block instance level of a `program` POU (cf. Figure 3b). For the rendering, the Python package 'Jinja2'[8] is used.

3 Application example

In this section, an application example is presented to highlight the currently developed features of the translator.

[8] `https://pypi.org/project/Jinja2/`

3.1 Demonstration building and HVAC system

CDL-PLC is used to translate a control sequence of a heat pump-based supply system for the renewable and residual energy sources (R^2ES)-based renovation of the 'Stijn Streuvelstraat' neighborhood in Bruges, Belgium. The project details are described in Walther et al. (2025).

3.2 Description of the control sequence

The control sequence is used to regulate a hybrid heating supply system with an air-source heat pump (ASHP) and a ground-source heat pump (GSHP). The FBD representation in Modelica is depicted in Figure 5. The control sequence has two inputs (the outdoor temperature `TAmb` and the tank temperature `TTan`) and two outputs (the two control signals `yAsHp` and `yGsHp` for the modulation of the two heat pumps, assuming that the compressor speed can be directly controlled). A so-called 'sequence controller' with a PI controller and a signal splitter is used to initially modulate only one heat pump, and to add the second heat pump if required. The `signalSplitter` block splits the input signal between `0` and `1` into two sequential output signals such that the first output increases from `0` to `1` when the input is between `0` and `0.5`, and the second output increases accordingly from `0` to `1` for inputs between `0.5` and `1`.

The hysteresis instance `hys` is used to switch the priority depending on the outdoor temperature: at lower outdoor temperatures the GSHP is the first generator in the sequence, and at higher outdoor temperatures the ASHP.

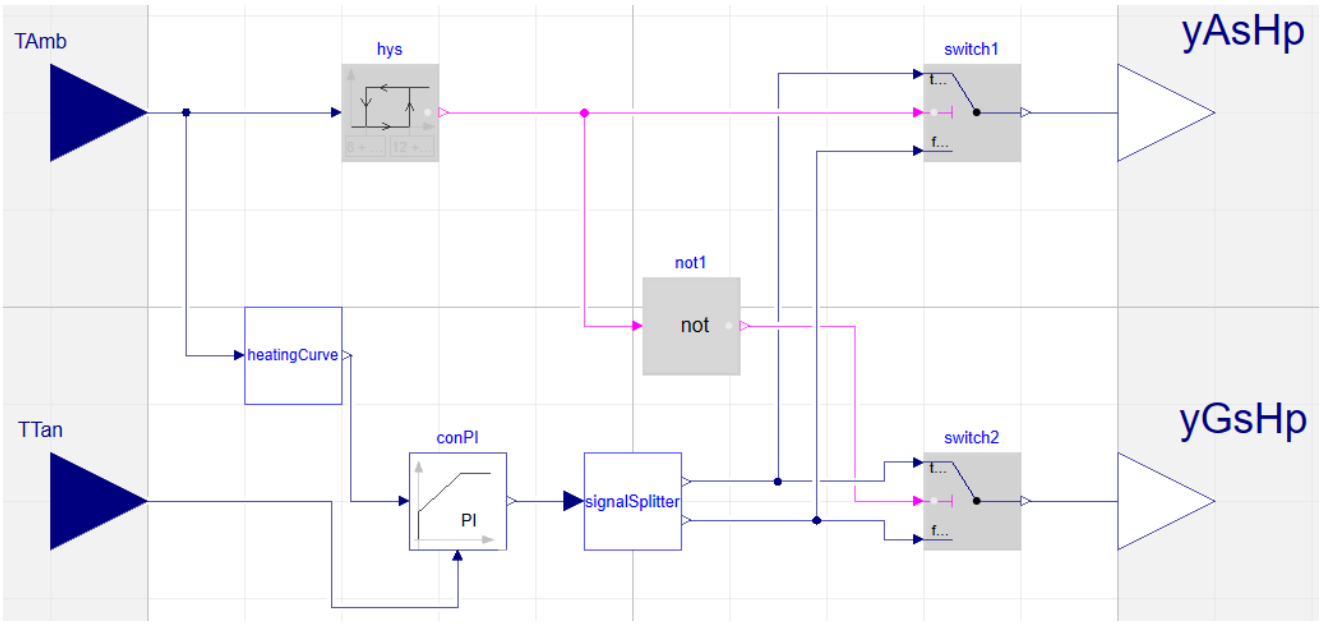
3.3 Comparison of CXF JSON-LD and IEC 61131-10 XML

The CDL source sequence and the translated IEC 61131-3 sequence are depicted in Figure 5 and Figure 5b.

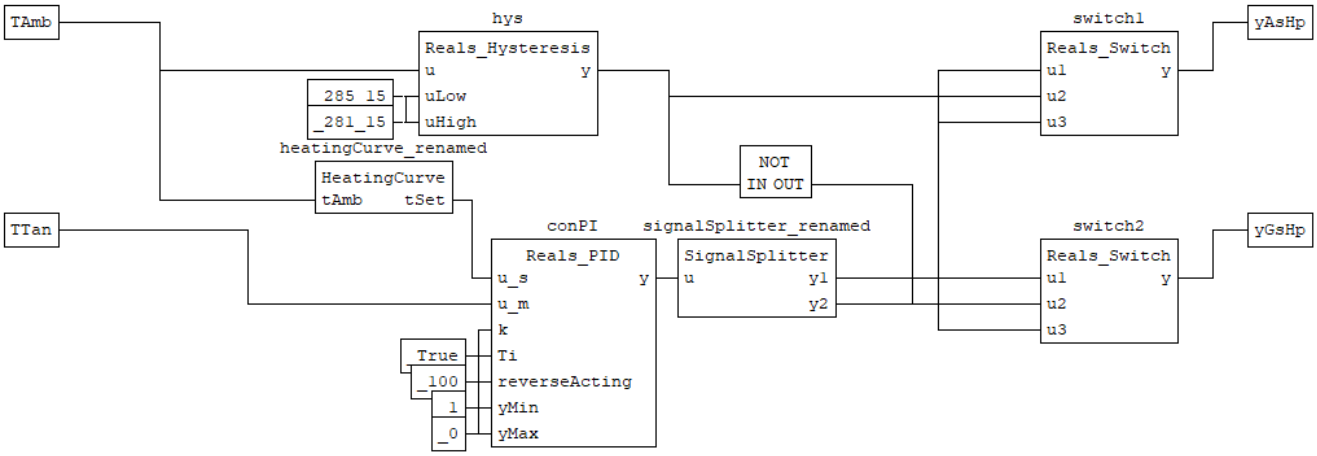
Mapping of inputs, outputs, function blocks, and connections Figure 5b shows that the two inputs (`TAmb` and `TTan`) and the two outputs (`yAsHp` and `yGsHp`) have been transferred accordingly from the CDL representation (Figure 5). This illustrates that the definition of identifiers for inputs and outputs is already done during the design phase in Modelica, and not, as is current practice in construction, only during the commissioning phase.

As described in subsubsection 2.3.1, two cases can be distinguished: (1) CDL blocks that have a direct equivalent in IEC 61131-3 standard functions are mapped to those: the CDL elementary block `not1` (type `Logical.Not`) is mapped to the IEC 61131-3 standard block `NOT`. (2) CDL blocks without a direct equivalent in IEC 61131-3 standard functions, such as `Reals.Hysteresis` or `Reals.PID`, are mapped to the individually translated IEC 61131-3 blocks.

Note that for `Reals.PID`, which is an elementary block, we translated it as if it were a composite block because the CDL implementation is in fact done using only constructs as allowed for composite blocks. In this example, `conPI` is configured as PI controller, which can



(a) Graphical view of CDL, as rendered by Dymola



(b) Translation in PLC code (Beremiz)

Figure 5. Comparison of function block diagrams for a heat pump control sequence

be defined via the `controllerType` parameter. The two user-defined composite blocks `heatingCurve` and `signalSplitter` are translated recursively by iterating through the hierarchical instance tree.

Figure 5b also shows that connections are correctly translated.

Parameter propagation Figure 5b illustrates how parameters of the `hys` and the `conPI` CDL blocks are transformed into IEC program inputs that are connected to the corresponding block inputs. The parameter value with a leading underscore is used as name for the input object.

Graphical representation The comparison of Figure 5 and Figure 5b reveals differences in the graphical rendering of the FBD representation. To take into account the variable size of function blocks in IEC FBDs and avoid overlapping blocks, the IEC FBD layout is stretched in x and y directions. This is, however, a particular issue of Beremiz, where blocks can be positioned freely. Other (commercial) IDEs, such as Codesys or TwinCAT, ignore the x-y-coordinate information and re-position the blocks depending on the relation on inputs and outputs.

4 Conclusion and future work

This study has presented the CDL-PLC translator for the conversion of Rule-Based Control sequences from the Modelica-based CDL (Control Description Language) into the IEC 61131-10 XML for PLCs. The CDL-PLC translator is an example for the translation of CXF to a controller specific language, and the last step for the implementation of HVAC control sequences expressed in CDL on IEC 61131 building controllers. We presented an example for the validation of an IEC block against its

DOI
10.3384/ecp218515

CDL equivalent, and an example for the translation of a real HVAC control sequence.

Implementing model-based design poses several challenges along the building life cycle. First and foremost, developing BPS software that enables the simulation of HVAC systems and controls for buildings with up to several hundreds of zones fast enough to be usable by practitioners remains challenging (Sahlin, Skogqvist, Lebedev, et al. 2019; Wetter, Benne, et al. 2023). In the meantime, simplifying load models such as by adding multiple rooms to one representative room is a recommended work-around.[9] For mechanical / HVAC and control engineers, this transformation can be done in two ways: Either by performing detailed HVAC control sequence development, for example using Modelica editors, during the pre-award design phase, which is not done today. Alternatively, ready-to-configure control and HVAC templates can be developed by library developers as shown in Gautier et al. (2023) and distributed with Modelica libraries, possibly integrated in configuration tools such as ctrl-flow.[10] The challenge for PLC vendors is to provide control sequence import functionality, either, if not already available, for the IEC 61131-10 XML or directly for the FMI standard. A noteworthy development is that a first PLC vendor starts to support the FMI standard[11].

To further develop and establish CDL-based model-based design of HVAC systems, joint efforts of academia and industry partners are required. Joint efforts need to focus not only on software toolchains and interfaces, but also - and perhaps even more importantly - on business models and best practice workflows to show various stakeholders the path to and benefit of adopting such model-based, digitalized processes.

The CDL-PLC translator is currently at a prototypical stage supporting a few CDL blocks as a proof of concept that such a translation is feasible. For a fully-fledged software, all CDL blocks have to be mapped and translated, systematically validated against their CDL counterparts, lacking features, such as the support of arrays[12], have to be added, and the import in different, particularly commercial, PLC IDEs has to be tested. Another challenge is the implementation of state machines in the translator. In CDL, a state machine can be implemented through so-called 'extension blocks' that are translated into FMUs.[13] The implementation of FMUs on the target automation system requires, however, that the target system is FMI-compatible, while the presented translator primarily targets legacy PLC-based automation systems that do not yet support the FMI standard (see '1' in 'Workflow and use cases'). One possible workaround would be a custom translator that converts the state machine code in the extension block to equivalent PLC code.

An important aspect for automated workflows is a complementary semantic model. CDL and its CXF translation supports such a semantic model based on ASHRAE Standard 223P, however, this information is lost when translating from CXF to IEC 61131-10 XML. To link the IEC 61131-10 XML with a semantic model, Ihlenburg et al. (2024) have proposed the PLCont methodology, however, further efforts towards a full integration or an extension of the XML schema are required.

[9] The simulation time for the control sequence used for the application example in section 3 with 61 thermal zones is around 24 h (Walther et al. 2025)

[10] https://ctrl-flow.lbl.gov.

[11] https://github.com/Beckhoff/TE142x---FMU-Samples/

[12] https://obc.lbl.gov/specification/cdl.html#arrays

[13] https://obc.lbl.gov/specification/cdl.html#sec-ext-blo

Data availability

The CDL-PLC translator is available under https://github.com/lbl-srg/cdl-plc.

Acknowledgements

The authors acknowledge the funding by the European Union through the SEEDS project under the Horizon Europe Programme (grant Agreement: 101138211). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Executive Agency (REA). Neither the European Union nor the granting authority can be held responsible for them.

Funded by the European Union

This research was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Building Technologies Office, of the U.S. Department of Energy, under Contract No. DE-AC02-05CH11231.

References

Barwig, Floyd E. et al. (2002-08). "The National Building Controls Information Program". In: Summer Study on Energy Efficiency in Buildings. ACEEE. Pacific Grove, CA.

Crowe, Eliot et al. (2020). "Building commissioning costs and savings across three decades and 1500 North American buildings". In: *Energy and Buildings* 227, p. 110408. ISSN: 0378-7788. DOI: 10.1016/j.enbuild.2020.110408.

Gautier, Antoine et al. (2023-10). "HVAC and Control Templates for the Modelica Buildings Library". In: *Proceedings of the 15th International Modelica Conference*. Linköping Electronic Conference Proceedings. Aachen, Germany: Modelica Association and Linköping University Electronic Press, pp. 217–228. DOI: 10.3384/ecp204217.

Ihlenburg, Moritz et al. (2024). "Towards a Digital Representation of Building Systems Controls". In: *International Sustainable Energy Conference - Proceedings*. ISEC 2024 – 3rd International Sustainable Energy Conference. Vol. 1. Graz, Austria: AEE - Institut für Nachhaltige Technologien (AEE INTEC). DOI: 10.52825/isec.v1i.1217.

International Electrotechnical Commission (2003). *Programmable Controllers: Part 3: Programming Languages*.

Marcos, Marga et al. (2009). "XML Exchange of Control Programs". In: *IEEE Industrial Electronics Magazine* 3.4, pp. 32–35. ISSN: 1932-4529. DOI: 10.1109/MIE.2009.934794.

Raftery, Paul et al. (2024). "Insights from Hydronic Heating Systems in 259 Commercial Buildings". In: *Energy and Buildings*, p. 114543. ISSN: 03787788. DOI: 10.1016/j.enbuild.2024.114543.

Sahlin, Per, Axel Bring, and Lars Eriksson (2009). "Real Controllers in the Context of Full-Building, Whole-Year Simulation". In: *Eleventh International IBPSA Conference*. Building Simulation 2009. Glasgow, Scotland: IBPSA, pp. 72–79. URL: https://publications.ibpsa.org/conference/paper/?id=bs2009_0072_79.

Sahlin, Per, Patrik Skogqvist, and Markus Högberg (2018). *IEC 61131-3 Code Simulation for Software-in-the-loop PLC Testing with EQUA Tools*. 15 pp. URL: https://itea3.org/project/workpackage/document/download/5090/D3.5%20OPENCPS%20-%20IEC61131-3%20Code%20Simulation%20For%20SiL%20PLC%20Testing%20M36.pdf.

Sahlin, Per, Patrik Skogqvist, Alexey Lebedev, et al. (2019). "On the Scalability of Equation-Based Building and District Simulation Models". In: *Proceedings of Building Simulation 2019: 16th Conference of IBPSA*. Ed. by Vincenzo Corrado et al. Building Simulation Conference Proceedings. IBPSA, pp. 2584–2590. DOI: 10.26868/25222708.2019.210130.

Sehr, Martin A. et al. (2021). "Programmable Logic Controllers in the Context of Industry 4.0". In: *IEEE Transactions on Industrial Informatics* 17.5, pp. 3523–3533. ISSN: 1551-3203. DOI: 10.1109/TII.2020.3007764.

Torabi, Narges et al. (2022). "Common Human Errors in Design, Installation, and Operation of VAV AHU Control Systems – A Review and a Practitioner Interview". In: *Building and Environment* 221, p. 109333. ISSN: 03601323. DOI: 10.1016/j.buildenv.2022.109333.

Verein Deutscher Ingenieure (2022). *Building Automation and Control Systems (BACS). Methods and Tools for Planning, Building, and Acceptance Tests. BACS Automation Scheme, BACS Function List, BACS Function Description*.

Walther, Karl (2025). "Model-based design, digital delivery, and implementation of HVAC controls — Lessons learned from a building-scale application for AHUs and PLCs". In: *Building and Environment* 282, p. 113126. ISSN: 03601323. DOI: 10.1016/j.buildenv.2025.113126.

Walther, Karl et al. (2025). "Simplifying Decision-Making in the Model-Based Co-Design of Building Energy Systems through Automatically Generated Optimal Controllers". In: Building Simulation 2025. Brisbane, Australia. DOI: https://doi.org/10.26868/25222708.2025.1690.

Wetter, Michael, Kyle Benne, et al. (2023). "Spawn: Coupling Modelica Buildings Library and EnergyPlus to Enable New Energy System and Control Applications". In: *Journal of Building Performance Simulation*, pp. 1–19. ISSN: 1940-1493. DOI: 10.1080/19401493.2023.2266414.

Wetter, Michael, Yan Chen, et al. (2023-09). "Bridging the gap between building energy modeling and controls implementation - experiences of, and best practice for, coupled HVAC-control modeling". In: *18-th IBPSA Conference*. Vol. 18. International Building Performance Simulation Association. Shanghai, China, pp. 791–798. DOI: 10.26868/25222708.2023.1258.

Wetter, Michael, Paul Ehrlich, et al. (2022). "OpenBuildingControl: Digitizing the Control Delivery from Building Energy Modeling to Specification, Implementation and Formal Verification". In: *Energy* 238, p. 121501. ISSN: 03605442. DOI: 10.1016/j.energy.2021.121501.

Wetter, Michael, Milica Grahovac, and Jianjun Hu (2018a). "Control Description Language". In: *Proceedings of the 1st American Modelica Conference*. Cambridge, MA: Modelica Association and Linköping University Electronic Press, pp. 17–26. ISBN: 978-91-7685-148-7. DOI: 10.3384/ECP1815417.

Wetter, Michael, Milica Grahovac, and Jianjun Hu (2018b-08). "Control Description Language". In: *1st American Modelica Conference*. Cambridge, MA, USA. DOI: 10.3384/ecp1815417.

Wetter, Michael, Jianjun Hu, et al. (2021). "Modelica-Json: Transforming Energy Models to Digitize the Control Delivery Process". In: *Proceedings of Building Simulation 2021a: 17th Conference of IBPSA*. Building Simulation Conference Proceedings. Bruges. Belgium: IBPSA, pp. 1–8. DOI: 10.26868/25222708.2021.30141.

Wilde, Pieter de (2014). "The Gap between Predicted and Measured Energy Performance of Buildings: A Framework for Investigation". In: *Automation in Construction* 41, pp. 40–49. ISSN: 09265805. DOI: 10.1016/j.autcon.2014.02.009.

Yet Another Residential District Simulator: *yards* for Controller Development in the Residential Built Environment

Lucas Bex[1,3,*] Muhammad Hafeez Saeed[1,3] Lucas Verleyen[2,3] Lieve Helsen[2,3] Geert Deconinck[1,3]

[1]Department of Electrical Engineering, KU Leuven, Belgium,
[2]Department of Mechanical Engineering, KU Leuven, Belgium
[3]EnergyVille, Belgium
[*]cas.bex@kuleuven.be

Abstract

Innovation in residential energy systems drives research efforts towards novel building and district control strategies. Development and testing of such strategies requires advanced, interactive building and district simulators. This paper presents *yards*, an interactive simulator that combines the modelling capabilities of Modelica with a language-agnostic and cross-platform interface for controller development. *yards* offers modelling flexibility beyond that of existing tools, as any custom, user-provided building or district model can be implemented easily. A case study demonstrates how *yards* can be used to simulate a tiny cluster of buildings.

Keywords: BOPTEST, Control Benchmarking, Functional Mock-Up Interface, Residential Energy Systems, yards

1 Introduction

Driven by climate targets, energy systems in the residential built environment are transforming. Fossil fuel-based technologies are being replaced by a combination of distributed renewable energy-based technologies e.g. photovoltaics (PV), heat pumps, batteries, thermal storage… To ensure an affordable, reliable and secure energy supply, hybrid configurations including different energy technologies at community level, rather than individual building level, have been gaining interest. Additionally, several energy services, such as heating, cooling and transport, are electrified, leading to the interconnection of the entire energy system via the electrical energy carrier. The increase in degrees of freedom, the interactions between the energy vectors and the wide range in time constants (sub-second-scale for electricity, minute-scale for thermal systems, daily-scale for short-term storage and seasonal-scale for long-term storage) lead to increasingly complex, but simultaneously increasingly flexible energy systems. Hence, to operate such energy systems and to use the full flexibility potential, advanced control strategies, customised to the specific energy system design, are necessary. Software-based emulators are essential for prototyping and validating such control strategies, as real-world experiments are often costly and time-consuming.

Emulators and controllers differ fundamentally in how they model and interact with physical systems, and as such tooling requirements for both are incompatible. Emulators describe acausal relationships between physical quantities in the form of equations. As emulators do not have fixed input-output relationships, they benefit from a declarative programming style that mimics the equation-based format. Since correctness is of major importance, emulators benefit from features such as static type and unit checking. In contrast, controllers inherently represent an input/output relation between sensor measurements and control signals sent to actuators. These relationships are more conveniently described through algorithms, in an imperative style, as opposed to equations. A focus on rapid prototyping and performance metrics rather than correctness favours dynamically typed programming languages for controller development, especially in research contexts where robustness and real-time requirements are typically less emphasised. For these reasons, software tools which focus on emulator development, such as Modelica, tend to be a poor fit for implementing some types of advanced controllers, such as data-driven model predictive control (MPC) or reinforcement learning. Indeed, while Modelica can be used to, for example, describe the system model within an MPC implementation, it is ill-suited to tasks such as training a neural network or tuning a gray-box model, which are essential in data-driven control strategies. Likewise, high-level programming languages that are suitable for controller development, such as Python or Julia, are often not ideal for creating an emulator. In these languages, tools for emulator development do exist, e.g. ModelingToolkit (Ma et al. 2021), but their syntax often bears little resemblance to that of the programming language they are used with. As such, control research for the residential built environment benefits from a multiple-tool approach: on the one hand, a declarative and statically typed programming language for the emulator; on the other hand, an imperative, dynamically typed programming language for the controller.

Various frameworks have been previously developed to interface building and district emulator models with controllers. The BOPTEST (Blum et al. 2021) and related

DOPTEST (Arroyo et al. 2023) frameworks have emerged as benchmarks for building and district controllers, respectively. Whereas the emulators in these frameworks have been implemented in Modelica, the simulation[1] interface is web-based and, as such, is controller-agnostic. This simulator-as-a-server approach allows complete freedom for control development. Both BOPTEST and DOPTEST support a curated set of emulators which serve as a global benchmark for advanced control strategies.

In contrast, CityLearn provides an efficient framework for benchmarking control algorithms, including rule-based control, MPC, and multi-agent reinforcement learning, at the urban scale, but it presents several limitations compared to more comprehensive platforms such as BOPTEST and DOPTEST (Vázquez-Canteli et al. 2019). First, its modelling approach consolidates district energy systems, heat pump behaviour, and building thermal dynamics into simplified representations, limiting the fidelity of simulations, especially for multi-zone or commercial applications (Nweye et al. 2025). Thus, while CityLearn is well-suited for rapid comparative analyses, high-fidelity studies may benefit from alternative, more detailed modelling environments. Second, the CityLearn framework is not language-agnostic: the framework is Python-based and requires controllers to be implemented as such.

Furthermore, TACO (Toolchain for Automated Control and Optimisation) (Jorissen, Boydens, and Helsen 2019) is a Modelica-based framework to develop white-box nonlinear MPC strategies for any energy system. However, TACO only implements and solves white-box MPC for detailed emulator models. Hence, energy system configurations can be compared, but no control strategies.

In this context, the field lacks a customisable, high-fidelity emulation tool for buildings and districts, which could further support the development of novel control strategies and the deployment of sustainable energy systems. BOPTEST and DOPTEST implement high-fidelity benchmark models, but these frameworks are not intended for simulating custom user models, yet alternatives such as CityLearn are not suitable for studies requiring high fidelity, while tools such as TACO support detailed emulator models but don't offer controller freedom. Therefore, this paper presents **y**et **a**nother **r**esidential **d**istrict **s**imulator, *yards*.[2,3] *yards* targets interactive, high-fidelity simulations of buildings and districts with support for custom, user-defined, energy system models and Key Performance Indicators (KPIs). *yards* bridges the gap between emulator designers who use Modelica to create emulators, and controller developers, who may interact with the *yards* simulator through a web Application Programming Interface (API). The *yards* simulator takes inspiration from the BOPTEST and DOPTEST frameworks and adopts the same interface, retaining near-compatibility with BOPTEST and DOPTEST from the perspective of the controller developer. *yards* is open source, cross-platform, and supports multiple (open source and proprietary) Modelica compilers.

The rest of the paper is structured as follows: Section 2 discusses *yards* design and implementation, and compares its design with that of the BOPTEST framework. Section 3 showcases *yards* from both emulator designer and controller developer perspective. The case study repackages and simulates a residential district model using the *yards* framework. Section 4 concludes the paper.

2 Design and Implementation

yards (Fig. 1) is consists of two components: the build pipeline and the simulator server. The build pipeline transforms the user[4]-provided model into a *yards*-compatible Functional Mock-up Unit (FMU) (Modelica Association 2022), and generates additional metadata files for the simulator based on the model and user-provided metadata files. The simulator defines a high-level web API to run the simulation, log the results, provide forecasts and calculate KPIs. This section first presents the build pipeline, followed by a discussion on the simulator.

2.1 Build Pipeline

The build pipeline generates all files required to run the simulator, i.e. the FMU and the files managed by the data manager in Fig. 1. To this end, *yards* generates a model description in XML format using OpenModelica and requires the user to specify any remaining metadata. As such, the user provides not only a Modelica model, but also several configuration files in JSON format (left side of Fig. 1): the `forecastables.json` file provides a description of all forecasts and how to calculate them; `libraries.json` is a list of Modelica dependencies of the model; `build_config.json` and `config.json` configure the build pipeline and simulator.

2.1.1 Model and Metadata

The emulator designer must provide *yards* with a simulatable Modelica model without inputs or outputs at the top level. Instead, the model includes read and overwrite blocks at arbitrary depth, indicating variables to be exposed or overwritten, respectively (Fig. 2). *yards* searches the generated XML description of the model for these read and overwrite blocks, and generates a wrapper model that contains the original model and exposes the inputs and outputs of these read and overwrite blocks at the top level. Subsequently, a *wrapped* FMU of this model can be generated using either OpenModelica or Dymola. The latter option requires the user to manually compile the wrapper model and provide *yards* with the compiled FMUs.

[1] This paper makes a distinction between emulators and simulators: emulators are models to test/benchmark a controller against; simulators are software that simulates a model.

[2] *yards* is intentionally all lower case.

[3] Available at `https://gitlab.kuleuven.be/positive-energy-districts/yards`

[4] User refers to the emulator designer

DOI 10.3384/ecp218525

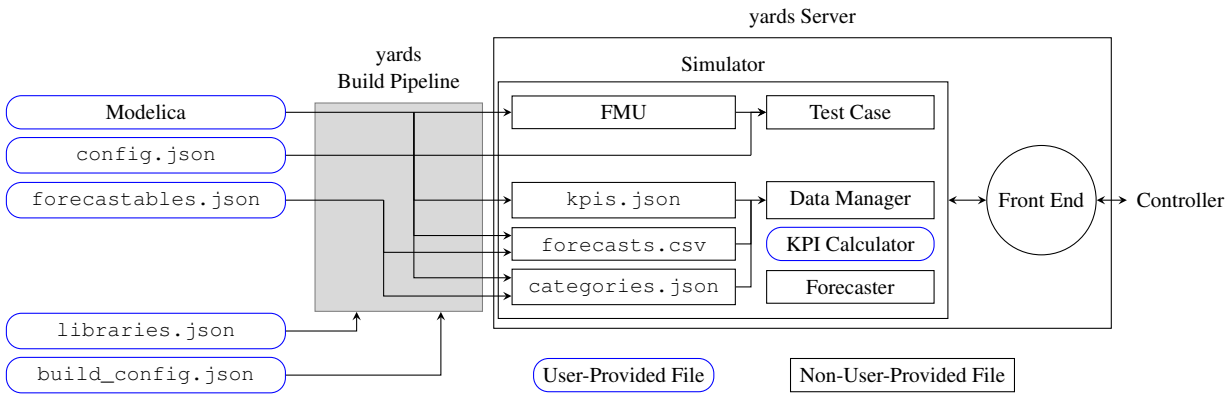


Figure 1. *yards* design overview: the emulator designer provides a Modelica model, corresponding metadata, a KPI calculator and a list of Modelica dependencies and how to install them. *yards* uses these inputs to build an FMU of the model, which is managed by the *yards* server and exposed to the controller via a web interface. The controller is not part of *yards*.

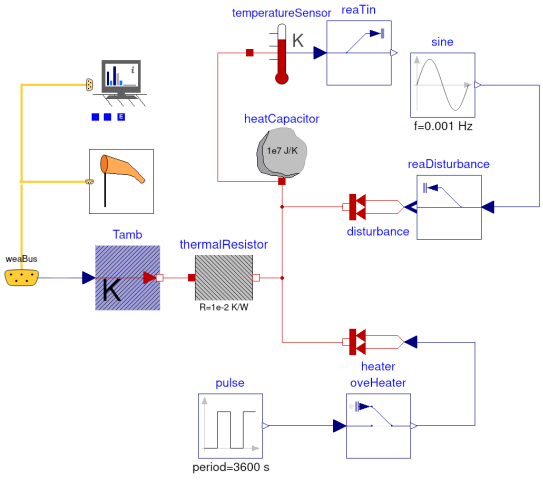


Figure 2. Example of a thermal RC model with read and overwrite blocks. The `reaDisturbance` and `reaTin` read blocks, and the `oveHeater` overwrite block indicate that these quantities should be exposed at the top level.

The read and overwrite blocks indicate which quantities are measurable or controllable, but they do not provide any information about their forecastability. *yards* relies on the emulator designer to specify which quantities have available forecasts, as defined in the `forecastables.json` file. Quantities should only be indicated as forecastable if there is no causal relationship between the quantity and any control input. For example, weather conditions are forecastable, but the building's indoor temperature depends on the control actions of the heating system and, as such, is not forecastable. The forecastables file takes the format of Listing 1: at the first level, the forecast interpolation method is specified (linear or zero-order hold); at the second level, forecast names are specified as keys, and their corresponding values indicate how the forecast is to be calculated and provide additional properties when necessary. In Listing 1, *forecast1* corresponds directly to a variable in the model. *yards* will search the XML description for metadata and replicate that metadata for this forecast. *forecast2* is an expression of multiple variables. In this case, *yards* is unable to determine the correct unit and description from the XML description. Hence, the emulator designer must provide this information.

Listing 1. Forecast metadata example

```
{ "linear": {
    "forecast1": "model.reaVar1.y",
}, "zoh": {
    "forecast2": {
        "expression": "model.reaVar2.y +
            model.reaVar3.y",
        "unit": "A unit",
        "description": "Var2 + Var3"
    }
  }
}
```

2.1.2 Implementation

The *yards* build pipeline (Fig. 3) uses a combination of Bash, Python and OpenModelica. To organise the build process, recipes are noted in a Makefile that can be parametrised via a simple configuration file (`build_config.json`). The pipeline runs in a containerised environment.

The build pipeline (Fig. 3) works as follows: First, the Modelica dependencies are downloaded to the container, and the OpenModelica path is populated with the model dependencies. Second, a prelude script to load all model dependencies is generated and combined with predefined templates to generate Modelica scripts. The script generation also involves filling in arguments such as model names, since custom command line arguments cannot be provided to `.mos` scripts. Third, the generated `dump_xml` script is used to obtain the XML description of the model. Fourth, the wrapped model is generated, and the user-provided metadata is combined with the XML description to generate the forecast data and metadata files for the simulator. Generating the forecast data involves simulating the original model for a user-specified time frame and resolution. Since this step can be costly, *yards* provides

options to accelerate this step using a simplified model. Fifth, the wrapped model is compiled into an FMU. Sixth, the generated forecast and metadata files are bundled with the wrapped FMU, and the pipeline is finished.

Whereas the FMU compilation step by default uses OpenModelica, *yards* allows compiling FMUs with other tools, such as Dymola.[5] In that case, the build pipeline stops after generating the wrapped Modelica model (step four above). The user should then compile the FMU manually. *yards* checks (and repairs) the model description of the manually compiled FMU for deficiencies against the OpenModelica XML description. The repaired FMU is bundled with the generated metadata files, after which the pipeline is finished.

2.1.3 Comparison with BOPTEST

The *yards* and BOPTEST build pipelines differ in three aspects: design approach, tooling, and model description generation. *yards* presents a configurable monolithic build pipeline aimed at non-expert emulator designers who require only minimal knowledge of the build process. In contrast, BOPTEST implements and maintains a separate build pipeline for each model but reuses some common utility functions across test cases. The declarative configuration required by *yards* including model metadata, is arguably easier to use compared to manually writing build scripts. BOPTEST uses the JModelica compiler (Åkesson, Gäfvert, and Tummescheit 2009), the Optimica Compiler Toolkit (OCT) and Dymola (Dempsey 2006) to build its test cases. OCT and Dymola are proprietary software, whereas the open source development of JModelica has been discontinued (Modelon 2019). *yards* depends on the open source OpenModelica compiler (Fritzson et al. 2020) but also supports other compilers such as Dymola (Dempsey 2006). The choice of tooling leads to differences in model generation as well. Whereas BOPTEST extracts model metadata from an FMU of the original model, *yards* uses OpenModelica to generate an easily parseable model description directly from the Modelica model using OpenModelica's `dumpXMLDAE` function. The approach of *yards* avoids compiling an intermediate FMU, which is typically an expensive step in the build process.

2.2 Simulator and API

2.2.1 Simulator

The *yards* simulator contains four core components (see Fig. 1): the forecaster, the KPI calculator, the data manager, and the test case. The forecaster provides predictions for forecastable quantities. The KPI calculator computes model-specific key performance indicators. The default *yards* KPI calculator returns an empty dictionary when KPIs are queried, but emulator designers may provide a custom implementation (in Python) instead. KPI calculators may implement arbitrary KPIs based on the exposed measurements, overwrites and forecasts. *yards* provides a base class to simplify the implementation of custom KPI calculators. The data manager tracks control signals and measurements, providing a convenient interface for other components to access simulation data. The test case handles the FMU simulation and processes incoming requests from the front end.

yards reuses a substantial part of the BOPTEST codebase. The simulator is implemented in Python and uses the PyFMI package (Andersson, Åkesson, and Führer 2016) to interact with the FMU.

[5] Our approach could hypothetically work with any tool that compiles FMUs from Modelica models, but we have only tested it with Dymola and OpenModelica

Table 1. Types of requests that *yards* accepts and their behaviour

Type	Behaviour
GET	Obtain model/simulator metadata
PUT	Obtain data/set simulator parameter/initialise
Identical sequential PUT requests → same result	
POST	Advance simulation
Identical sequential POST requests → different results	

Table 2. *yards* API and compatibility with BOPTEST (v0.5.0). Additions to the API after BOPTEST v0.5.0 are unimplemented.

Command	BOPTEST compatibility
GET forecast_points	Compatible
GET inputs	Compatible
GET kpi	Model-specific return
GET kpi_absolute	Model-specific return, superset
GET measurements	Compatible
GET name	Compatible
GET scenario	Compatible
GET step	Compatible
GET version	Compatible
POST advance	Compatible
POST submit	Silent or loud failure
PUT forecast	Compatible
PUT initialize	Compatible
PUT results	Compatible
PUT scenario	Model-specific input and return
PUT step	Compatible

2.2.2 API

The *yards* simulator provides a REST API for interactively simulating the model, obtaining measurements and forecasts, and for calculating KPIs. The API is available through HTTP, such that *yards* is not restricted to controllers implemented in any specific programming language. Commands are grouped into HTTP GET, PUT and POST requests by intended behaviour (Table 1). Responses are JSON strings and always take the format of Listing 2. The front end implementation uses the Flask framework (Pallets Projects 2010).

DOI 10.3384/ecp218525

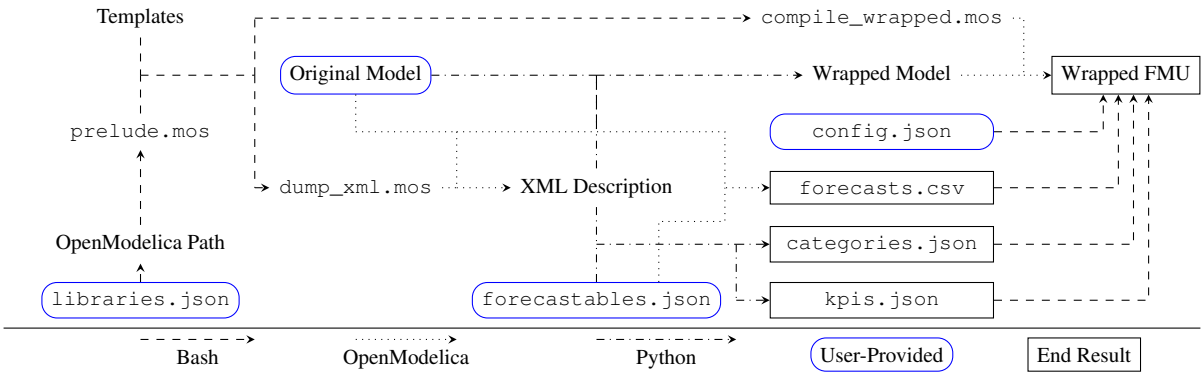


Figure 3. *yards* build pipeline. This diagram represents the case where OpenModelica is chosen as the compiler and the entire pipeline can be run in the container. Additional steps are needed if external compilers are used.

Listing 2. *yards* API response. The status value is 200, 400 or 500, indicating success, invalid request or internal errors, respectively.

```
{
  "message": "Informative message",
  "payload": {
    "a request-specific": "data structure"
  },
  "status": 200
}
```

The *yards* API is fully compatible with that of DOPTEST, which is in turn a superset of the BOPTEST (v0.5.0) API, in terms of request format. Responses and behaviour are, however, not identical for the complete API. Differences are listed in Table 2. Commands related to BOPTEST dashboard submission fail either silently or loudly (depending on configuration). As the KPIs and scenario are defined by the emulator designer, related commands and their return values may take arbitrary formats.

3 Case Study

This case study[6] showcases the *yards* framework, from both an emulator designer standpoint and a controller developer perspective. The section starts with the model description, followed by an overview of the KPIs. Next, the implementation of the model and KPIs in the *yards* framework is described, comprising the emulator designer's tasks. Afterwards, three controllers and their implementation are treated. The section ends with a discussion of the results.

3.1 Model Description

This case study uses the `MoPED.Districts.Layout4` model from the open source MoPED library (Verleyen et al. 2022). This model is similar to the `MoPED.Districts.Layout3` model, which is used in the DOPTEST framework (Arroyo et al. 2023). The district model is implemented using component models from the Modelica Standard Library (Modelica Association 2020), the Modelica Buildings library (Wetter et al. 2014), and the IDEAS library (Jorissen, Reynders, et al. 2018). For detailed information on the model equations, refer to (Arroyo et al. 2023) and the respective library documentation.

The model `MoPED.Districts.Layout4` represents a tiny cluster consisting of three residential, detached, single-family buildings. Each house is modelled as a single-zone building, relying on the detailed, white-box IDEAS (Jorissen, Reynders, et al. 2018) *RectangularZoneTemplate*. This building model template models each building envelope component as a construction consisting of several material layers and includes one-dimensional conduction, internal and external convection, and long-wave and short-wave radiation, allowing for dynamic heat transfer simulations. Occupancy behaviour is included to account for the internal heat gains of the occupants. More information on the building geometry and occupancy can be found in (Arroyo et al. 2023).

Each building of the tiny cluster is equipped with an individual energy system to provide heat for space heating and domestic hot water, cold for space cooling, and electricity for appliances. The only interconnection between the buildings is via the electrical distribution grid. Each building has an individual air-to-water heat pump connected to a simple parallel throttling circuit with an embedded floor heating system for space heating and a domestic hot water storage tank. Domestic hot water is drawn from the tank via a tap, resulting in the inflow of cold city water into the tank. Space cooling is provided by a separate air-to-air heat pump. The electrical load of each house is a combination of an inflexible demand profile for plug loads and flexible loads consisting of the air-to-water heat pump and the air-to-air heat pump. Each house has a PV installation with an inverter that curtails power generation when the grid voltage deviates by more than 10% from the nominal voltage.

To obtain realistic results, the use case is subject to consistent boundary condition data of 2021 linked to Leuven (Belgium). Weather data has been measured by the Build-

[6] Available at `https://gitlab.kuleuven.be/positive-energy-districts/yards-case-study`

ing Physics Section of the KU Leuven on campus. Occupancy profiles, including plug loads, have been generated using StROBe (Baetens and Saelens 2016). Hourly emission profiles have been derived from the Belgian generation mix (ENTSO-E 2025) and emission factors published by ElectricityMaps (ElectricityMaps 2025). Hourly electricity prices correspond to the hourly day-ahead price (Elexys 2024) without additional taxes.

3.2 Key Performance Indicators

The emulator designer can customise the KPI calculation according to their needs. This case study implements the KPIs below. All KPIs are calculated over the full simulation horizon, spanning from the start time t_s to the final time t_f. KPIs can be calculated on a per-house basis (κ^i, $i \in \{1,2,3\}$) or on a district level (κ^{tot}).

3.2.1 Thermal Discomfort

For space heating and space cooling, the thermal discomfort, $\kappa_{tdis,shc}$, in $[Kh]$, is calculated as the cumulative deviation of the operative zone temperature $T_o(t)$ from its lower $\underline{T}_o$ and upper $\overline{T}_o$ comfort limits for each house i (Eq. (1)).

$$\kappa^i_{tdis,shc} = \int_{t_s}^{t_f} \delta_{T_{o,i}}(t)\,dt \qquad i \in \{1,2,3\} \tag{1a}$$

$$\delta_{T_{o,i}}(t) = \begin{cases} \underline{T}_{o,i} - T_{o,i}(t) & \text{if } T_{o,i}(t) < \underline{T}_{o,i} \\ 0 & \text{if } T_{o,i}(t) \in [\underline{T}_{o,i}, \overline{T}_{o,i}] \\ T_{o,i}(t) - \overline{T}_{o,i} & \text{if } T_{o,i}(t) > \overline{T}_{o,i} \end{cases} \tag{1b}$$

The comfort limits are determined according to the EN16798-1:2019 standard (Eq. (2), where $n_{occ,i}$ is the amount of building occupants present in house i).

$$\underline{T}_{o,i} = \begin{cases} 20°\text{C} & \text{if } n_{occ,i} > 0 \\ 15°\text{C} & \text{otherwise} \end{cases} \qquad i \in \{1,2,3\} \tag{2a}$$

$$\overline{T}_{o,i} = \begin{cases} 26°\text{C} & \text{if } n_{occ,i} > 0 \\ 30°\text{C} & \text{otherwise} \end{cases} \qquad i \in \{1,2,3\} \tag{2b}$$

For domestic hot water, the thermal discomfort, $\kappa_{tdis,dhw}$, also in $[Kh]$, is calculated as the cumulative deviation of the tank temperature $T_{tan}(t)$ from its lower limit $\underline{T}_{tan} = 45°\text{C}$ for each house i (Eq. (3)).

$$\kappa^i_{tdis,dhw} = \int_{t_s}^{t_f} \delta_{T_{tan,i}}(t)\,dt \qquad i \in \{1,2,3\} \tag{3a}$$

$$\delta_{T_{tan,i}}(t) = \begin{cases} \underline{T}_{tan,i} - T_{tan,i}(t) & \text{if } T_{tan,i}(t) < \underline{T}_{tan,i} \\ 0 & \text{if } T_{tan,i}(t) \geq \underline{T}_{tan,i} \end{cases} \tag{3b}$$

Thermal discomfort is aggregated at the district level as in Eq. (4).

$$\kappa^{tot}_{tdis,shc} = \sum_{i=1}^{3} \kappa^i_{tdis,shc} \tag{4a}$$

$$\kappa^{tot}_{tdis,dhw} = \sum_{i=1}^{3} \kappa^i_{tdis,dhw} \tag{4b}$$

3.2.2 Electric Energy Use

Electric power is aggregated at the building and district levels. For an individual building i, electric power exchange at the building's grid connection point P_i comprises the PV installation, plug loads, the air-to-water heat pump for heating and the air-to-air heat pump for cooling. Injection and offtake electric power of individual buildings is defined in Eq. (5).

$$P_{inj,i}(t) = \max\{0, P_i(t)\} \qquad i \in \{1,2,3\} \tag{5a}$$

$$P_{off,i}(t) = \max\{0, -P_i(t)\} \qquad i \in \{1,2,3\} \tag{5b}$$

At the district level, energy exchange between individual houses is taken into account, resulting in the district injection and offtake electric power (Eq. (6)).

$$P_{inj,tot}(t) = \max\{0, \sum_{i=1}^{3} P_i(t)\} \tag{6a}$$

$$P_{off,tot}(t) = \max\{0, -\sum_{i=1}^{3} P_i(t)\} \tag{6b}$$

The electric energy use KPIs, κ^{inj}_{ener} and κ^{off}_{ener}, aggregate electric power injection and offtake over the simulated time period, resulting in Eq. (7) (with abuse of notation).

$$\kappa^{inj,i}_{ener} = \int_{t_s}^{t_f} P_{inj,i}(t)\,dt \qquad i \in \{1,2,3,tot\} \tag{7a}$$

$$\kappa^{off,i}_{ener} = \int_{t_s}^{t_f} P_{off,i}(t)\,dt \qquad i \in \{1,2,3,tot\} \tag{7b}$$

3.2.3 Operational CO_2 Emissions

Since all energy services are electrified, the operational CO_2 emissions only originate from the offtake of electricity $P_{off}(t)$ from the grid.

$$\kappa^{off,i}_{emis} = \int_{t_s}^{t_f} \varepsilon(t)\,P_{off,i}(t)\,dt \qquad i \in \{1,2,3,tot\} \tag{8}$$

$\varepsilon(t)$ is the hourly CO_2 emission factor. The houses are rewarded for injecting emission-free electricity generated by their PV installation, $P_{inj}(t)$. Sharing energy between buildings within the district is considered emission-free, but when injecting energy into the grid, an injection compensation equal to 50% of the emission intensity is assumed.

$$\kappa^{inj,i}_{emis} = \int_{t_s}^{t_f} 0.5\,\varepsilon(t)\,P_{inj,i}(t)\,dt \qquad i \in \{1,2,3,tot\} \tag{9}$$

A clear distinction between the emissions due to offtake, κ^{off}_{emis}, and the emission compensation due to injection, κ^{inj}_{emis}, is made to have a better understanding of the system's behaviour.

3.2.4 Operational Costs

Since all energy services are electrified, the operational costs only originate from the offtake of electricity, $P_{off}(t)$, from the grid.

$$\kappa_{cost}^{off,i} = \int_{t_s}^{t_f} c(t)\, P_{off,i}(t)\, dt \qquad i \in \{1,2,3\} \qquad (10)$$

$c(t)$ is the hourly electricity price. The houses are rewarded for injecting electricity generated by their PV installation, $P_{inj}(t)$. An injection tariff equal to 50% of the offtake price is assumed.

$$\kappa_{cost}^{inj,i} = \int_{t_s}^{t_f} 0.5\, c(t)\, P_{inj,i}(t)\, dt \qquad i \in \{1,2,3\} \qquad (11)$$

The case study does not consider any financial agreements at the district level. As such, the district injection revenue and offtake cost are simply the sum of the individual revenues and costs (Eq. (12)).

$$\kappa_{cost}^{inj,tot} = \sum_{i=1}^{3} \kappa_{cost}^{inj,i} \qquad (12a)$$

$$\kappa_{cost}^{off,tot} = \sum_{i=1}^{3} \kappa_{cost}^{off,i} \qquad (12b)$$

3.3 Implementation

To prepare the model for the *yards* build pipeline, metadata, dependencies, and the builder configuration must be provided.

3.3.1 Modelica and Model Metadata

The `MoPED.Districts.Layout4` model is obtained from the MoPED repository (Verleyen et al. 2022). This model has been expressly built for DOPTEST and as such, the necessary read/overwrite blocks are already present. The Modelica dependencies of MoPED are specified in the `libraries.json` file. These dependencies can be either installed directly through OpenModelica or downloaded using Git. Alternatively, the user may mount a local version of the dependencies in the builder container and provide an empty `libraries.json` file. Forecast metadata is provided in the format of Listing 1.

The configuration of the build pipeline (Listing 3) specifies various technicalities such as the name of the models to compile, the path to the model package and metadata files, but also includes instructions for generating the forecast data. Since the district model is quite computationally expensive, the simplified `MoPED.Districts.Layout4RC` model is used to generate the forecasts for weather and occupancy instead. This model uses linear second-order RC models for the building envelope instead of the more complex IDEAS *RectangularZoneTemplate*. The `MoPED.Districts.Layout4RC` model is simulated with the provided arguments, and the simulation results are used as forecasts.

Listing 3. Build pipeline configuration for the case study

```
{
  "models": ["MoPED.Districts.Layout4"],
  "modelicafile": "MoPED/package.mo",
  "metadata": "metadata",
  "compiler": "omc",
  "forecast": {
        "forecast_model": "MoPED.Districts.
            Layout4RC",
        "start_time": 0.0,
        "stop_time": 691200.0,
        "number_of_intervals": 768
  }
}
```

3.3.2 KPI Calculator

The KPI calculator (Listing 4) implements each KPI as a separate method. The *yards*-provided base class is used for easy access to model variables at simulation runtime. Some KPIs, such as those related to operational emissions take extra parameters, which can be passed by setting the scenario, either in the `config.json` or during the simulation through a request (Table 2). The structure and contents of the scenario can be fully customised by the emulator designer.

Listing 4. KPI calculator for the use case. The code is simplified and abbreviated. Class methods have been de-indented one extra level for readability.

```
from .baseclass import BaseKPICalculator
class KPICalculator(object):
# mandatory methods
def __init__(self, testcase):
  self.base = BaseKPICalculator(testcase)
  self.house_prefixes = ["house1", "house2"
      , "house3"]
def initialize(self):
  self.base.initialize()
def get_kpis_absolute(self, scenario=None):
def get_core_kpis(self, scenario=None):
# helper methods
def emissions(self, prefix,
    injection_scaling_factor=0.5):
def thermal_discomfort(self, prefix):
...
def get_house_kpis(self, prefix, scenario):
def get_district_kpis(self, scenario):
```

3.3.3 Building the Model

To build the FMU, the *yards* builder image must be started as a container with local development files mounted (Listing 5). The `buildcache` and `target` volumes cache intermediate build steps and store the build result. Using this approach, changes in model, metadata and KPI calculator files are available immediately and build results can be accessed by inspecting the `buildcache` volume.

Listing 5. Run the *yards* build pipeline

```
$ docker run \
    -v buildcache:/yards/build \
    -v target:/yards/target \
    -v ./MoPED:/yards/MoPED \
```

```
    -v ./metadata:/yards/metadata \
    -v ./build_config.json:/yards/
        build_config.json \
    yards/builder
```

To run the simulator after finishing the build, a different container based on the *yards* simulator image must be started. When starting this container (Listing 6), the `target` volume and custom KPI calculator must be mounted and the network must be configured.

Listing 6. Run the *yards* simulator. After running this command the simulator can be accessed via `http://localhost:5001`.

```
$ docker run -p 5001:5000 \
    -v target:/home/user/target \
    -v ./kpis/kpi_calculator.py:/home/user/
        kpis/kpi_calculator.py \
    yards/simulator
```

Above approach is suitable for active development of the simulator, but requires that model files are available on the host machine. To simplify distribution of the packaged model, one may create a new image that includes the built FMU and custom KPI calculator (Listing 7) and distribute this image.

Listing 7. Package the model by running the *yards* build pipeline as an intermediate stage in a multi-stage image build resulting in a self-contained simulator image with the custom model and KPIs.

```
FROM yards/builder:latest AS build
# copy model and metadata config files from
    host to image
COPY . /yards/
RUN ./entrypoint.sh make target
FROM yards/simulator:latest
COPY --chown=user --from=build /yards/
    target/models/wrapped.fmu target/models
    /wrapped.fmu
COPY --chown=user kpis/kpi_calculator.py ./
    kpis/kpi_calculator.py
```

3.4 Controller Development

This section introduces and compares three controllers for space heating. The three controllers are the baseline (C0), the price-based controller (C1) and the price-difference-based controller (C2). Controller C0 reduces the temperature setpoint when no occupants are present, to reduce energy usage. The price-based controller C1 preheats the building when the electricity price is low, to avoid energy use when the price is high. Controller C2 forecasts the electricity price and preheats the building only before a significant price increase. All three controllers aim to balance thermal comfort with energy use and cost.

3.4.1 Controller Design

In the district model, the control logic for space heating contains two control loops. The outer loop determines the temperature setpoint $T_{o,\mathrm{set}}(t)$ for the zone. The inner hysteresis-based loop operates the heat pump to drive the zone temperature towards this setpoint. The hysteresis band has a width of 1°C and is centred around the current setpoint $T_{o,\mathrm{set}}$. The heat pump is switched on or off to maintain the zone temperature within this band. All controllers in this section determine the temperature setpoint $T_{o,\mathrm{set}}$ and rely on the inner control loop to operate the heat pump.

First, controller C0 implements the rule in Eq. (13). Since comfort bounds are less strict when the building is empty (Eq. (2)), the temperature setpoint can be decreased significantly at this time. Second, the rule for C1 is given by Eq. (14): when the electricity price is low, temperature is increased. The thermal inertia of the building can be used to reduce energy consumption when prices rise above this threshold, reducing costs.

$$T^{\mathrm{C0}}_{o,\mathrm{set},i} = \begin{cases} T^{\mathrm{C0}}_{\mathrm{high}} = 21\,^{\circ}\mathrm{C} & \text{if } n_{occ,i} > 0 \\ T^{\mathrm{C0}}_{\mathrm{low}} = 15\,^{\circ}\mathrm{C} & \text{otherwise} \end{cases} \tag{13}$$

$$T^{\mathrm{C1}}_{o,\mathrm{set},i} = \begin{cases} T^{\mathrm{C1}}_{\mathrm{high}} = 24\,^{\circ}\mathrm{C} & \text{if } c(t) \leq c_{\mathrm{th}} \\ T^{\mathrm{C1}}_{\mathrm{low}} = 21\,^{\circ}\mathrm{C} & \text{otherwise} \end{cases} \tag{14}$$

Finally, controller C2 aims to anticipate large changes in electricity price. It modulates the temperature setpoint between upper and lower limits depending on the forecasted short-term change in price Δc (Eq. (15), where Δt is the timestep, $\alpha \in [0,1]$ the modulation factor, and $\Delta c_{scale} = 0.05$ €/kWh, a parameter to tune controller sensitivity).

$$\Delta c = \max\{c(t + j\Delta t) - c(t) | j = 0 \ldots 6\} \tag{15a}$$

$$\alpha = \min\{\Delta c / \Delta c_{scale}, 1\} \tag{15b}$$

$$\begin{aligned} T^{\mathrm{C2}}_{o,\mathrm{set},i} &= (1-\alpha) T^{\mathrm{C2}}_{\mathrm{low}} + \alpha T^{\mathrm{C2}}_{\mathrm{high}} \\ &= 21^{\circ}\mathrm{C} + \alpha \cdot 3^{\circ}\mathrm{C} \end{aligned} \tag{15c}$$

3.4.2 Implementation and Simulation

Controller C0 is implemented directly in the Modelica model and is part of the MoPED library; it can be simulated by calling the *POST advance* method of the *yards* API without arguments. Controllers C1 and C2 are implemented in Python (Listing 8). They first query the electricity price via a *PUT forecast* call, and use this information to compute the heating setpoint. The heating setpoint is then supplied as an argument to the *POST advance* call to *yards*.

Listing 8. Implementation of C1. Code altered for brevity.

```
def do_step(url, Tlow, Thigh, cth):
  price_key = '
      PriceElectricPowerHighlyDynamic'
  price = requests.put(f'{url}/forecast',
      json={'point_names': [price_key], '
      horizon': 0, 'interval': 1}).json()['
      payload'][price_key]
  Tset = Tlow if price[0] > cth else Thigh
  setpoints = {
    'house1_hva_oveTSetHeaDay_u': Tset,
```


DOI
10.3384/ecp218525

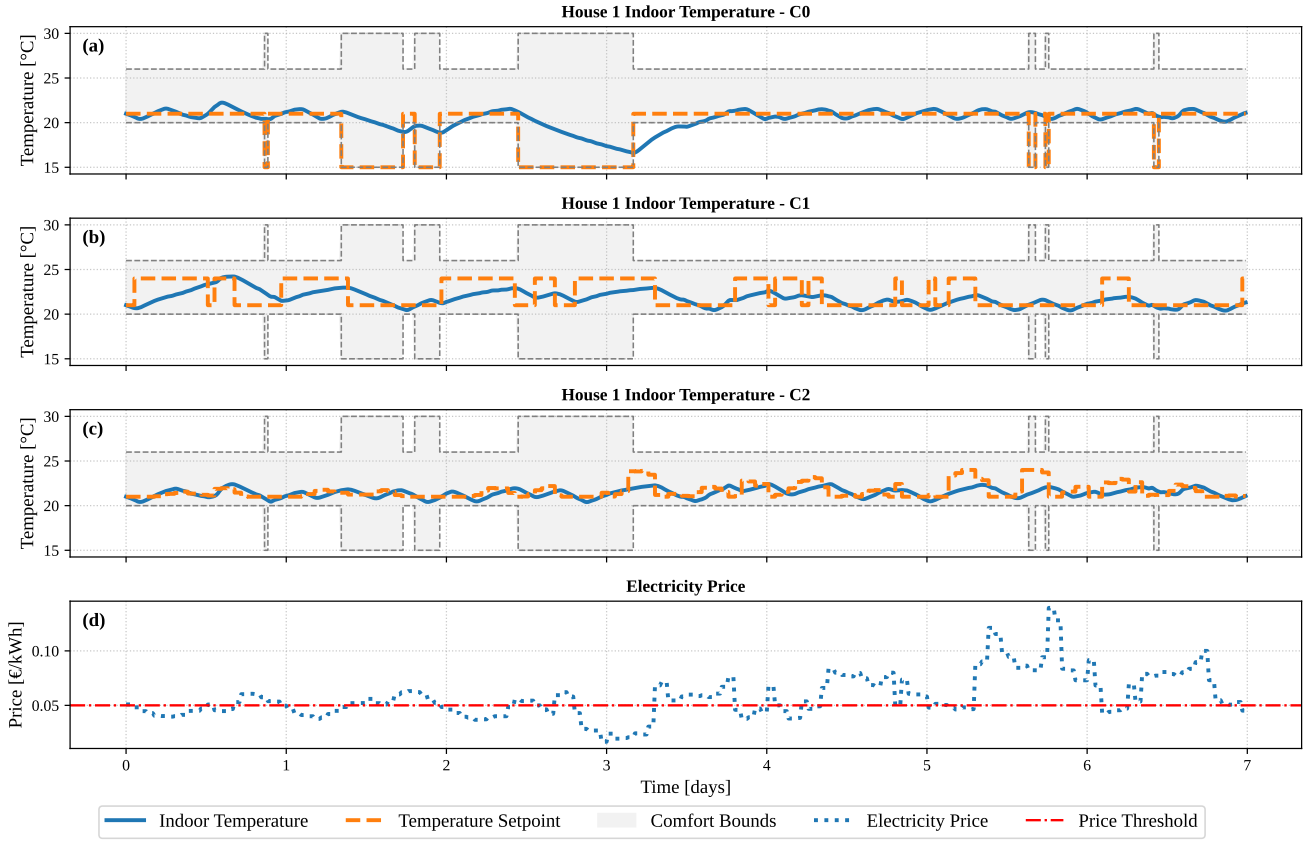


Figure 4. (a)-(c) Indoor temperature in house 1 (solid line) over a one week period under each of the three control strategies ((a) - C0, (b) - C1, (c) - C2). Dashed lines show the heating setpoints computed by the controllers, and the lightly shaded band represents the occupancy-dependent comfort bounds (Eq. (2)). (d) Dynamic electricity price (dotted) and C1 price threshold (dash-dotted).

```
    'house1_hva_oveTSetHeaDay_activate': 1,
    # set other houses...
}
requests.post(f'{url}/advance', json=
    setpoints).json()['payload']
```

All simulations are performed for one week, starting on the 1st of January, with a time step Δt of 1 hour. For each of the three simulations, the same controller was applied to all three houses of the tiny cluster simultaneously.

3.5 Results and Discussion

The KPI calculator allows to easily compare the implemented controllers (Table 3). For all three controllers, district electricity offtake is substantially larger than injection, which aligns with expectations for a winter month with limited PV production and consistent heating demand. The baseline C0 achieves the lowest energy use and operational emissions, but incurs higher costs and space heating-related thermal discomfort than the custom C1 and C2 controllers. Figure 4 clearly shows the cause of the thermal discomfort: C0 is unable to anticipate the sudden change in comfort bounds, resulting in thermal discomfort (around day 2 and 3.25 in Fig. 4a). In contrast, C1 and C2 (Figs. 4b and 4c) use more conservative temperature setpoints, and thus avoid this problem. Controller C1 achieves the lowest offtake cost (€43.00, -4.97% vs. C0) but also the highest offtake-related CO_2 emissions (44.10 $kgCO_2$, +2.8% vs. C0).

Table 3. District-level KPIs for all three controllers. These results were obtained through a *GET kpi* call to *yards* after finishing each simulation.

Metric		C0	C1	C2
Operational Costs [€]	$\kappa_{cost}^{inj,tot}$	0.49	0.24	0.36
	$\kappa_{cost}^{off,tot}$	45.25	43.00	43.31
Operational Emissions [$kgCO_2$]	$\kappa_{emis}^{inj,tot}$	0.32	0.15	0.26
	$\kappa_{emis}^{off,tot}$	42.90	44.10	43.93
Energy Use [kWh]	$\kappa_{ener}^{inj,tot}$	10.25	4.76	8.11
	$\kappa_{ener}^{off,tot}$	760.5	773.5	775.2
Thermal Discomfort [Kh]	$\kappa_{tdis,shc}^{tot}$	18.05	0.00	0.00
	$\kappa_{tdis,dhw}^{tot}$	0.00	0.00	0.00

A follow-up to this case study could be to develop a novel, more performant and more advanced controller. This may be done by another researcher, using a different programming language, on a different operating system. The flexibility of *yards* to package custom models with a consistent interface opens up a lot of possibilities for

collaborative and open research: emulator designers can independently and rapidly publish their models for controller developers to optimise. Where centralised, curated benchmarks like BOPTEST and DOPTEST allow comparisons between controllers across the globe on a curated set of models, *yards* is ideal for collaborations comparing a limited set of controllers for an arbitrary model or even for use in a one-person project. As such, the work presented in this paper can be seen as complementary to existing efforts, rather than competitive.

4 Conclusion

yards is a novel building and district simulator for developing and testing advanced control strategies with high-fidelity models. It allows emulator designers to package Modelica models as an interactive web-based simulator with which controllers can interact. Custom KPI calculators can optionally be embedded alongside the models.

The presented case study demonstrates *yards*' capabilities on a small residential district, showing how *yards* facilitates cross-language co-simulation of emulator and controller, and allows to easily compare controller performance using custom KPIs. *yards* provides a versatile environment for evaluating and testing advanced control concepts in residential energy systems.

Acknowledgements

The authors acknowledge the financial support by KU Leuven through the TECHPED - C2 project (C24M/21/021). The TECHPED project investigates TECHnically feasible and effective solutions for Positive Energy Districts. Lucas Bex and Muhammad Hafeez Saeed ackowledge the support from Research Foundation Flanders - FWO (research fellowships 1S00325N and 1S66625N, respectively). Climatic data were collected in Leuven by the Building Physics Section of KU Leuven.

References

Åkesson, Johan, Magnus Gäfvert, and Hubertus Tummescheit (2009). "JModelica—an Open Source Platform for Optimization of Modelica Models". In: 6th Vienna International Conference on Mathematical Modelling.

Andersson, Christian, Johan Åkesson, and Claus Führer (2016). *PyFMI: A Python Package for Simulation of Coupled Dynamic Models with the Functional Mock-up Interface*. Vol. LUTFNA-5008-2016. Technical Report in Mathematical Sciences. Centre for Mathematical Sciences, Lund University.

Arroyo, Javier et al. (2023). "Prototyping the DOPTEST Framework for Simulation-Based Testing of System Integration Strategies in Districts". In: *Proceedings of the 18th IBPSA Conference*. Shanghai, China.

Baetens, Ruben and Dirk Saelens (2016). "Modelling uncertainty in district energy simulations by stochastic residential occupant behaviour". In: *Journal of Building Performance Simulation* 9.4, pp. 431–447. DOI: 10.1080/19401493.2015.1070203.

Blum, David et al. (2021). "Building Optimization Testing Framework (BOPTEST) for Simulation-Based Benchmarking of Control Strategies in Buildings". In: *Journal of Building Performance Simulation* 14.5, pp. 586–610. ISSN: 1940-1493. DOI: 10.1080/19401493.2021.1986574.

Dempsey, Mike (2006). "Dymola for Multi-Engineering Modelling and Simulation". In: IEEE Vehicle Power and Propulsion Conference, pp. 1–6. DOI: 10.1109/VPPC.2006.364294.

ElectricityMaps (2025). *Emission factors*. URL: https://github.com/electricitymaps/electricitymaps-contrib/wiki/Emission-factors (visited on 2025-01-29).

Elexys (2024-01-15). *Spot Belpex*. URL: https://my.elexys.be/MarketInformation/SpotBelpex.aspx (visited on 2024-01-15).

ENTSO-E (2025). *Actual Generation per Production Type - Belgium*. URL: https://transparency.entsoe.eu/ (visited on 2025-01-29).

Fritzson, Peter et al. (2020-01-10). "The OpenModelica Integrated Environment for Modeling, Simulation, and Model-Based Development". In: *Modeling, Identification and Control* 41.4, pp. 241–295. DOI: 10.4173/mic.2020.4.1.

Jorissen, Filip, Wim Boydens, and Lieve Helsen (2019). "TACO, an Automated Toolchain for Model Predictive Control of Building Systems: Implementation and Verification". In: *Journal of Building Performance Simulation* 12.2, pp. 180–192. ISSN: 1940-1493. DOI: 10.1080/19401493.2018.1498537.

Jorissen, Filip, Glenn Reynders, et al. (2018). "Implementation and Verification of the IDEAS Building Energy Simulation Library". In: *Journal of Building Performance Simulation* 11.6, pp. 669–688. DOI: 10.1080/19401493.2018.1428361.

Ma, Yingbo et al. (2021). *ModelingToolkit: A Composable Graph Transformation System For Equation-Based Modeling*. arXiv: 2103.05244 [cs.MS].

Modelica Association (2020-06-04). *Modelica Standard Library*. Version v4.0.0. URL: https://github.com/modelica/ModelicaStandardLibrary (visited on 2025-04-16).

Modelica Association (2022-11-10). *Functional Mock-Up Interface for Model Exchange and Co-Simulation*. Version 2.0.4. URL: https://github.com/modelica/fmi-standard/releases/download/v2.0.4/FMI-Specification-2.0.4.pdf (visited on 2023-12-13).

Modelon (2019-12-18). *JModelica.Org*. URL: https://jmodelica.org/ (visited on 2025-04-11).

Nweye, Kingsley et al. (2025). "CityLearn v2: Energy-flexible, resilient, occupant-centric, and carbon-aware management of grid-interactive communities". In: *Journal of Building Performance Simulation* 18.1, pp. 17–38.

Pallets Projects (2010). *Flask*. URL: https://github.com/pallets/flask (visited on 2025-04-16).

Vázquez-Canteli, José R et al. (2019). "Citylearn v1. 0: An openai gym environment for demand response with deep reinforcement learning". In: *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, pp. 356–357.

Verleyen, Lucas et al. (2022-11). "Identifying technically feasible and effective solutions towards Positive Energy Districts (PEDs)". In: *Proceedings of the Urban Energy in a Net Zero World Conference*. Glasgow, Scotland.

Wetter, Michael et al. (2014). "Modelica Buildings library". In: *Journal of Building Performance Simulation* 7.4, pp. 253–270. DOI: 10.1080/19401493.2013.765506.

Facilitating the use of Physics-Based Simulations on Embedded Devices by running FMUs from MicroPython

Tom REYNAUD[1] Erfan ENFERAD[2] Maxime LEFRANCOIS[3]

[1]Mines Saint-Étienne, France, tom.reynaud@etu.emse.fr
[2]erfan.enferad@gmail.com
[3]Mines Saint-Étienne, Univ Clermont Auvergne, INP Clermont Auvergne, CNRS, UMR 6158 LIMOS, F - 42023, Saint-Étienne, France, maxime.lefrancois@emse.fr

Abstract

Physics-based simulations (PBS) are increasingly valuable for real-time applications in embedded systems, yet integrating them on resource-constrained devices remains challenging. This paper presents ufmu, a lightweight framework that enables execution of source-code FMI 2.0-compliant Functional Mock-up Units (FMUs) within the MicroPython environment, targeting platforms such as the ESP32. The proposed approach integrates into the MicroPython firmware the FMU source code and the FMU model description translated into C structures, and exposes a minimal Python API for simulation control, enabling model-based computations on-device without cloud dependencies. We evaluate the framework using a standard FMU model, comparing performance across ESP32, Unix, and plain C environments in terms of memory usage, execution time, and firmware size. Despite the ESP32's hardware limitations, the results demonstrate that meaningful simulations can be achieved efficiently, with minimal memory overhead. All code, documentation, and experiment instructions are freely available under an MIT license, supporting reproducibility and adoption in education, prototyping, and embedded research. This work also lays the foundation for future integration with eFMI and the FMI 3.0 standard.

Keywords: Physics-Based Simulation, FMI, FMU, eFMI, MicroPython

1 Introduction

The simulation software market is expected to reach 40.5 billion US dollars in 2030 according to a report (Zoting 2023). In recent years physics-based simulation (PBS) has become a very important tool in different domains of both science and industry. PBS support for the analysis and prediction of events, with applications ranging from breaking systems in vehicles (Magargle et al. 2017) to earthquake prediction (Saad et al. 2023), having the potential to save lives.

Embedded devices support physics-based simulation (PBS) by supplying data. In a cloud–edge architecture, they typically collect sensor data, such as temperature or humidity, and transmit it to the cloud for processing and storage. Still, performing real-time calculations in the cloud is costly, as it increases network traffic and latency, lowers energy efficiency, and can hinder scalability (Savaglio et al. 2019). Recent developments enable embedded devices to run simulations themselves, helping to address these issues while also simplifying the overall architecture (Lenord et al. 2021). However, the skill gap between simulation development and embedded system programming remains a barrier to broader adoption and integration of these capabilities, especially for hobbyists or DIY (Do-It-Yourself) enthusiasts.

The Functional Mockup Interface (FMI), developed by the Modelica Association, is an open standard for sharing and co-simulating dynamic models, supported by tools for monitoring, evaluation, and plotting (Blockwitz et al. 2012). Models are packaged as Functional Mock-up Units (FMUs), which encapsulate code, data, and metadata in a standardized format for easy integration across tools.

Developed as part of the EMPHYSIS project, the embedded Functional Mock-up Interface (eFMI) extends FMI to enable simulation execution on embedded devices (Lenord et al. 2021). eFMI introduces a dedicated programming language, GALEC, with the goal to ensure PBS execution stays within the resource constraints of embedded systems. GALEC guarantees static worst-case execution time and memory usage, and prevents illegal memory access. Additionally, eFMI's container architecture supports multiple model representations, including algorithmic, production, and binary code. These features make eFMI suitable for safety-critical applications and adaptable to various embedded software environments.

Optimized for microcontrollers, MicroPython (George and MicroPython community 2014) is a streamlined implementation of Python 3 designed to compile and execute code directly on microcontrollers, which are at the heart of most embedded devices. MicroPython is easier to learn and use than C (Fangohr 2004) and provides a rich set of libraries, making it well-suited for education and rapid prototyping in embedded systems (Tollervey 2017). To our knowledge, embedding FMI simulations directly within the MicroPython environment is unprecedented and complements the eFMI paradigm.

Therefore, this paper addresses the following research

question: *How can FMUs be integrated into MicroPython for execution on embedded devices ?* We propose the ufmu framework, which enables execution of FMI 2.0-compliant FMUs within MicroPython by:

1. translating the FMU model description into a C structure,
2. integrating it and the FMU source code into the MicroPython build, and
3. exposing the FMU via a lightweight Python API through the ufmu library, allowing simulations to run directly within MicroPython.

This paper makes the following key contributions:

- A framework for exporting and running one source-code FMI 2.0-compliant FMU embedded inside the MicroPython firmware;
- Performance evaluation of the proposed integration;
- Exploration of future perspectives, including the potential use of eFMI for further optimizations and adaptation to the FMI 3.0 standard.

The rest of this paper is structured as follows. Section 2 provides background information. Section 3 outlines the ufmu framework, detailing our methodology, associated challenges, and optimizations. It also validates the implementation through test cases, including running the bouncing ball simulation on an ESP32 microcontroller. Section 4 discusses related work. Finally, Section 5 concludes the paper by summarizing the contributions, discussing the implications of the work, and outlining future directions for research and development.

2 Background

2.1 The need for PBS in embedded devices

PBS are models that represent the physical behavior of a system. Applications include predictive maintenance, where a component's remaining life is estimated based on its current condition to enable timely intervention before failure (Magargle et al. 2017), and fault diagnosis, where the source of faults is identified from observed behavior to support corrective actions and improve system reliability (Aivaliotis et al. 2019).

The use of PBS directly inside embedded systems is an emerging field that holds great promise for improving the accuracy, efficiency, and performances of these systems (Tavella et al. 2016). Accuracy is improved by offering a more precise representation of the system's physical behavior, which is crucial for real-time operations in safety-critical systems, medical devices, and automobiles (Armugham et al. 2021). Efficiency is improved by enabling auto-calibrating scenarios, reducing the need for extensive testing and debugging (Liu and Negrut 2021; Gundermann et al. 2017). Performance is enhanced by optimizing the architecture and operation of the system, leading to a more efficient and responsive system (Vanommeslaeghe et al. 2018; Tavella et al. 2016). Additionally, performing the process on the embedded device itself boosts responsiveness and scalability.

However, several challenges remain. First, PBS can be computationally demanding, restricting their deployment on resource-constrained devices. Second, integration with existing embedded software can be complex and may require significant development effort and expertise. Finally, their sensitivity to input data quality makes them difficult to apply in real-world scenarios (Duraiswami and Zotkin 2016). Despite these challenges, the benefits of integrating physics-based simulations into embedded systems are evident. As technology advances, their adoption is likely to grow across a wide range of applications.

2.2 FMI and FMUs

The Functional Mockup Interface (FMI), developed by the Modelica Association, is an open standard for sharing and co-simulating Modelica models, which are typically exported as Functional Mock-up Units (FMUs). FMUs encapsulate code, data, and metadata in a uniform package, enhancing the credibility and usability of simulations in complex engineering processes (Gall et al. 2021).

FMI 2.0 is the most widely adopted and extensively supported version of FMI, with over 210 tools offering compatibility and ensuring seamless integration across industries such as automotive, aerospace, and energy.[1] While FMI 3.0 introduces advanced co-simulation features, richer data-type support, and scheduled-execution interfaces—such as 8-, 16-, 32-, and 64-bit integer types, single-precision floats, and opaque binary variables—its adoption remains nascent compared to 2.0. Given its proven stability, broad toolchain compatibility, and full backward-compatibility with existing FMUs, FMI 2.0 is the preferred choice for most engineering workflows today, and the one we will focus on in this paper.

The following outlines the sequence of FMI function calls that ensure accurate state propagation, event handling, and system updates within the simulation. All FMU function calls omit the `fmi2_` prefix for better readability. During initialization, the FMU is created with `instantiate`, and memory is allocated for state variables and event indicators. The experiment is configured using `setupExperiment`, followed by a transition into initialization mode with `enterInitializationMode`. Initial variable values are retrieved using `getReal` and `getInteger`. After exiting initialization mode with `exitInitializationMode`, an initial event iteration is performed using `newDiscreteStates`. The simulation loop starts by retrieving continuous states and derivatives via `getContinuousStates` and `getDerivatives`. Time advancement is handled with `setTime`, and the next simulation step is performed using `setContinuousStates`. The event handling mechanism is activated when necessary, transitioning into event mode with `enterEventMode` and resolving

[1] `https://fmi-standard.org/tools/`

DOI 10.3384/ecp218535

discrete state updates via `newDiscreteStates`. If required, the simulation re-enters continuous-time mode using `enterContinuousTimeMode`. Throughout the process, output variables are gathered by invoking `getReal` and `getInteger`.

2.3 MicroPython

MicroPython (George and MicroPython community 2014) is a lightweight implementation of Python designed for microcontrollers. Unlike CPython, which requires a full OS, MicroPython runs directly on bare-metal hardware, providing direct access to peripherals like GPIO, I2C, SPI, and UART, making it ideal for resource-constrained environments. Arduino has embraced MicroPython as a supported platform for programming microcontrollers, providing tools like the Arduino Lab for MicroPython to facilitate development. This integration allows developers to leverage the simplicity of Python while utilizing Arduino's extensive hardware ecosystem.

MicroPython leverages one of Python's key advantages: its ease of use, especially in comparison to C/C++, the dominant languages for embedded systems programming. According to (Fangohr 2004), Python-based development can significantly reduce the time needed for implementation and debugging.

Additionally, as discussed in (Tollervey 2017), MicroPython includes an extensive standard library and supports external modules, allowing developers to leverage a broad range of functionalities without needing to write low-level firmware from scratch. It also has built-in libraries to harness as much as its support can offer, like the for the ESP32, as shown in Figure 1

Despite its compact size, MicroPython remains highly extensible. Developers can integrate it with C/C++ code for performance-critical tasks, ensuring that time-sensitive operations are not compromised. Furthermore, the active community and robust documentation provide support, making MicroPython an attractive option for both hobbyists and professionals.

```
from machine import Pin
from onewire import OneWire
from ds18x20 import DS18X20
# Initialize DS18B20 sensor on GPIO pin 4
ds = DS18X20(OneWire(Pin(4)))
# Scan for connected devices
roms = ds.scan()
# Start temperature conversion
ds.convert_temp()
print("Temperature:", ds.read_temp(roms[0]),
↪  "°C")
```

Figure 1. Snippet of the MicroPython code to read temperature on an embedded device from an external sensor

3 The ufmu framework

3.1 Overview

Our approach integrates an FMU following the FMI 2.0 standard into a MicroPython-compatible C module, enabling users to interact with FMUs directly from MicroPython scripts. Figure 2 illustrates the process, which involves: (A) Exporting the simulation as FMU, (B) Decompressing the FMU, (C) Parsing and modifying the .c, .h, .xml files, (D) Building MicroPython with the ufmu library, (E) Flashing the device with the firmware.

The following subsections detail the process. Section 3.2 covers the extraction and preparation of the FMU archive for integration into the embedded environment. Section 3.3 discusses the process of adapting the FMU to work seamlessly with MicroPython, including wrapping simulation functions for compatibility. Several challenges had to be addressed, such as adapting FMU execution for a resource-constrained embedded environment, handling compilation differences between Unix and ESP32, and ensuring modularity and reusability of the functions. Section 3.4 discusses how these issues were tackled and the optimizations applied. Finally, Section 3.5 reports on the evaluation of the ufmu framework.

ufmu supports both the Unix and the ESP32 ports of MicroPython. All the code and necessary instructions to reproduce the experiments are openly available on Github under the open-source license of the MIT,[2] alongside the end-user documentation.[3]

3.2 FMU Preprocessing

The first step of the framework involves exporting the simulation as an FMU (Step A). FMUs are zip archives typically containing C source files, an XML model description file, and precompiled binaries for various platforms, as can be seen in States 2 and 3 of Figure 2. The framework then consists in preprocessing the FMU.

First, the FMU file is unzipped (Step B) using a Makefile, revealing its internal structure, including the `modelDescription.xml` file and source code (State 3). Then, the `modelDescription.xml` file is parsed to extract metadata about the FMU, such as variable definitions, simulation parameters, and function mappings (Step C). This parsing is performed using the `xmllint` software to dynamically configure the library.

Once the relevant information is extracted, the source files are modified to integrate with the MicroPython-compatible library. This includes wrapping core FMU functions to expose them as callable functions from MicroPython, adjusting variable storage and memory handling to fit the microcontroller's limitations, and adding required definitions to handle different prefixes for function names. Additionally, configuration headers are generated based on the extracted metadata, storing variable

[2] https://github.com/Imaginus02/MicroFMU-library
[3] https://imaginus02.github.io/MicroFMU-library

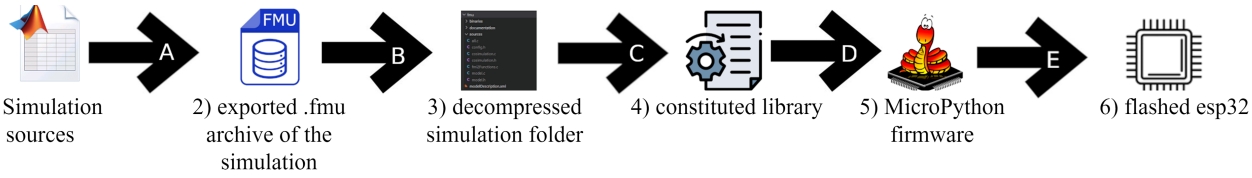


Figure 2. Flow diagram of the MicroPython firmware creation process, showing the integration of the FMU execution engine, preprocessing, and compilation steps.

definitions and simulation parameters efficiently (State 4).

This preprocessing ensures that each FMU is correctly configured before compilation, allowing it to run on the target environment with no manual intervention (either Unix or ESP32 in our case).

3.3 Library Integration into MicroPython

Once the FMU source code has been preprocessed, the next step involves integrating it into the MicroPython firmware (Step D), and flashing it on the device (Step E).

Integration is achieved using a C module that bridges the FMU execution engine and the MicroPython interpreter. This module exposes the core FMU functions via the minimal MicroPython-compatible ufmu API. As detailed in Annex A, the API includes a main class and utility functions to initialize a simulation instance with specified time parameters, access variable names and descriptions, read and modify variable values, and iterate through the simulation instance with output at each step.

The build process then differs by target: building for Unix uses a traditional Makefile workflow, while building for the ESP32 relies on CMake and the ESP-IDF framework, with different flags for function name overrides and linker configuration. Once the FMU execution engine is shipped into the MicroPython firmware, the end-developer only needs to define how the simulation is controlled and executed using this API.

Figure 3 illustrates the use of the ufmu API to control and execute the **BouncingBall.fmu** simulation. A simulation instance is created by calling the `setup_simulation` function with a start time, a stop time, and a step size. Once initialized, simulation variables can be accessed using `get_variable_names`, which returns an array containing the names of all FMU variables, and `get_variables_description`, which provides additional details on each variable's purpose and properties. The simulation instance is implemented as a Python *generator*, allowing users to step through it with a simple `for` loop. At each iteration, it yields an array of current variable values in a consistent order. Users can dynamically modify some of the variables during execution using `change_variable_value`, in line with the FMI 2.0 specification. This design ensures both simplicity and flexibility when interacting with the simulation.

```
>>> import ufmu as fmu
>>>
>>> # Print available variables and their
 ↪  descriptions
>>> for name in fmu.get_variables_names():
>>>     description =
 ↪  fmu.get_variables_description(name)
>>>     base_value =
 ↪  fmu.get_variables_base_values(name)
>>>     print(f"'{name}' (= {base_value} at
 ↪  t0): {description}")
'step' (= 0 at t0): Simulation step count
'time' (= 0.0 at t0): Simulation time
'h' (= 1.0 at t0): Position of the ball
'der(h)' (= 0.0 at t0): Derivative of h
'v' (= 0.0 at t0): Velocity of the ball
'der(v)' (= 0.0 at t0): Derivative of v
'g' (= -9.81 at t0): Gravity acting on the ball
'e' (= 0.7 at t0): Coefficient of restitution
>>>
>>> # Set up the simulation parameters
>>> startTime, stopTime, stepSize = 0, 3, 0.1
>>>
>>> # Initialize simulation instance
>>> sim = fmu.setup_simulation(startTime,
 ↪  stopTime, stepSize)
>>>
>>> # Change a variable (e.g., 'h' for height)
>>> fmu.change_variable_value(sim, 'h', 20)
>>>
>>> # Run the simulation and print time and
 ↪  height for each step
>>> for s, t, h, der_h, v, der_v, g, e in sim:
>>>     print(f"t: {t:.2f}s, h: {h:.2f}m")
t: 0.10s, h: 19.95m
t: 0.20s, h: 19.80m
...
```

Figure 3. REPL (Read-Evaluate-Print-Loop) snippet of MicroPython code to retrieve simulation information and run the **BouncingBall.fmu** simulation.

3.4 Challenges and Optimizations

DOI 10.3384/ecp218535

Developing the ufmu framework to run FMUs from MicroPython on an ESP32 presented several challenges that required careful consideration and optimization to ensure the library's functionality and efficiency. Since microcontrollers in embedded systems are typically dedicated to a single task, the approach adopted was to embed a single FMU within the MicroPython environment and expose it through the fmu module.

One of the main challenges was the discrepancy between the compilation processes for Unix and ESP32 targets. The Unix port relies on Makefiles, while the ESP32 port uses CMake and the ESP-IDF framework. This difference necessitated adjustments in build configurations to ensure compatibility across both platforms. Specifically, the ESP32 compilation process required additional flags to override certain function name prefixes in the FMU source code, which were incompatible with the linker's expectations. By dynamically adjusting the build flags, we ensured proper linking of functions during compilation, resolving this issue effectively.

Memory management was another critical challenge, given the ESP32's limited RAM (approximately 4MB). To optimize memory usage, the library reuses the same variable to store the results of each simulation step. Instead of retaining historical data, each new step overwrites the previous values, ensuring minimal memory consumption. This approach allows the library to handle simulations with a large number of variables and steps without exceeding the microcontroller's memory limits. The end-developer decides whether to retain the values or not.

Function name prefix conflicts also posed a significant challenge. The FMU source code often included prefixes that conflicted with the linker's expectations during the ESP32 compilation process. To address this, the library dynamically modifies the function names during preprocessing, ensuring they align with the linker's requirements. This adjustment was crucial for enabling successful compilation and execution on the ESP32.

3.5 Evaluation

While the Unix port served as a reliable testing ground, the ESP32 port required additional debugging due to its unique constraints. The bouncing ball simulation, provided as a FMU test case, was instrumental in identifying and resolving issues related to memory management, function naming, and execution speed.

We assessed the ufmu framework by measuring the impact of shipping the **BouncingBall.fmu** on the firmware size, and measuring the execution time and peak memory usage for running this model from 0 to 10 with two distinct configurations: one with a step size of 0.01 and another with a step size of 0.0001.

Two ports were compared to a plain C simulator[4] developed alongside the ufmu library: MicroPython for the ESP32 target, and MicroPython for Unix. To ensure the reliability and accuracy of our performance measurements, each simulation was executed 10 times under identical conditions, with the same code as shown in Figure 4, and the mean value of the results was calculated. This approach minimizes the impact of any outliers or transient system behaviors that could skew the data. The Plain C simulator and the MicroPython Unix port were both executed within a WSL2 environment on a Windows 10 system, leveraging the computational power of an Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz. The MicroPython ESP32 implementation, on the other hand, was run directly on the embedded hardware to evaluate its performance in a resource-constrained environment.

The results are summarized in Table 1. Integrating ufmu with the **BouncingBall.fmu** model increased the ESP32 MicroPython firmware by 220 KB (15 %), bumping it from 1.5 MB to roughly 1.72 MB. The Plain C simulator outperformed both Unix and MicroPython implementations in terms of execution speed (respectively 2 and 100 times faster). However, its memory usage, because of the fact that it keep all step result, is hardly comparable to the MicroPython one. The executable size is minimal, as it avoids the overhead associated with interpreted languages like MicroPython.

The MicroPython Unix port exhibited a bug when handling simulations with more than 5,000 steps, preventing it from completing the larger simulation (step size = 0.0001).[5] This issue is reflected in the results with an "X" in Table 1. Despite this limitation, the Unix port demonstrated competitive performance for smaller simulations.

The MicroPython ESP32 port, while significantly slower than the other two due to the hardware limitations of the ESP32, successfully completed both simulations. These results highlight the trade-offs between performance and resource efficiency when deploying simulations on embedded systems.

```
import ufmu as fmu
import time
import micropython
start_time = 0
end_time = 10
step_size = XX
start = time.ticks_ms()
simInstance = fmu.setup_simulation(start_time,
↪  end_time, step_size)
fmu.change_variable_value(simInstance, 'h', 20)
for i in simInstance:
        pass
print("Time taken: ",
↪  time.ticks_diff(time.ticks_ms(), start),
↪  "ms")
micropython.mem_info()
```

Figure 4. The code run by MicroPython to get the result presented in Table 1. The value replaced here by XX is respectively 0.01 and 0.00001

[4] https://github.com/Imaginus02/FMUSimulator

[5] This corresponds to a known issue: https://github.com/Imaginus02/MicroFMU-library/issues/1

4 Related Work

4.1 Portable runtime environments for Python-based FMUs

The concept of portability extends beyond just MicroPython. In the field of FMUs, ensuring that Python-based FMUs can execute across various systems without dependency conflicts is crucial. One approach to achieving this portability is through containerization.

A notable effort in this direction is the addition of Docker support to UniFMU (Schranz et al. 2021), a tool designed for building and executing FMUs. By encapsulating all necessary runtime dependencies in a Docker container, this approach allows FMUs to be distributed and deployed seamlessly across different environments. This capability is particularly beneficial in scenarios requiring co-simulation, as demonstrated in the case of a robotic arm and its controller, where two Python-based FMUs were successfully executed within a Docker container. This method eliminates compatibility issues and simplifies the deployment process, aligning with the broader goal of making Python-based embedded applications more accessible and portable.

By combining the lightweight nature of MicroPython with advances in FMU portability, developers can leverage Python for a wider range of embedded and simulation applications. The ability to run MicroPython on constrained hardware while simultaneously supporting Python-based FMUs in containerized environments highlights the versatility and growing influence of Python in embedded systems and simulation workflows.

This approach becomes particularly valuable when deploying large-scale embedded systems for co-simulation scenarios, such as in (Jung, Shah, and Weyrich 2018), where a dynamic, multi-agent-based simulation framework was used to model a smart warehouse. In this scenario, various IoT components—such as storage racks, temperature sensors, forklifts, and incoming goods—communicate and make autonomous decisions in real time. Each component is represented as an independent simulation agent capable of dynamically joining an ongoing co-simulation, enabling a flexible "Plug-and-Simulate" behavior. By leveraging modular co-simulation techniques, the study demonstrates how distributed embedded systems can interact seamlessly, allowing for efficient and adaptive digital twin implementations. The ability to deploy Python-based FMUs within portable runtime environments, such as Docker, further enhances this flexibility, enabling scalable and reproducible simulations across heterogeneous IoT ecosystems.

4.2 FMI and eFMI in Embedded, Automotive, and Industrial Applications

Recent advancements have demonstrated the feasibility of using FMUs and their embedded counterpart, eFMU, for real-time simulation on microcontrollers. The increasing computational capabilities of embedded hardware, along with the standardization of the FMI, have unlocked new possibilities for Cyber-Physical Systems (CPS), industrial automation, and automotive systems.

In the context of CPS, (Hong et al. 2020) proposed a simulation framework leveraging FMI, incorporating a redundancy reduction algorithm to optimize performance for resource-constrained microcontrollers. Similarly, (Gundermann et al. 2017) validated embedded FMI for lifetime testing of electromechanical systems through a 60-day real-time study on a linear stepper motor, confirming its reliability in embedded environments.

The eFMI standard, developed under the EMPHYSIS project and detailed in (Lenord et al. 2021), is tailored to meet the constraints of embedded systems, including hard real-time requirements and limited memory. It enables seamless integration of high-fidelity, physics-based models into embedded control systems, making it particularly impactful in domains like automotive, aerospace, and industrial automation, where such models can enhance control strategies, diagnostics, and predictive maintenance.

In the automotive sector, FMI plays a pivotal role in software validation, virtual Electronic Control Unit (ECU) testing, and real-time simulation of vehicle dynamics. (Mikelsons and Samlaus 2017) demonstrated the integration of FMI-based models into ECU validation workflows, identifying software issues early in the development process and improving reliability.

To overcome the incompatibility of traditional Co-Simulation FMUs with embedded automotive applications—due to stringent real-time constraints, MISRA compliance, and AUTOSAR architecture—eFMI has been extended to AUTOSAR environments (Armugham et al. 2021). In a case study, a physics-based model was integrated into a Bosch ECU using an AUTOSAR toolchain, illustrating a compliant, efficient, and cost-effective approach that significantly reduced the manual coding effort typically required.

Beyond automotive use cases, FMI and eFMI are also advancing industrial automation by enabling embedded simulations for real-time failure detection and predictive diagnostics. As embedded FMI technology evolves alongside edge computing and optimized firmware, its role in real-time embedded systems is expected to grow substantially, driving innovation across various high-demand sectors.

Building upon these advancements, our MicroPython FMU implementation bridges a gap in the embedded systems landscape by making simulation capabilities accessible to Python programmers in resource-constrained environments. While existing work has demonstrated FMU viability on dedicated microcontrollers, our approach extends these benefits to the growing ecosystem of MicroPython-enabled devices, enabling rapid prototyping, interactive debugging, and simplified deployment without sacrificing real-time performance. This contributes to the democratization of complex simulation

DOI
10.3384/ecp218535

Table 1. Executing the BouncingBall.fmu simulation in different Python environments with setting start_time = 0, end_time = 10, and step_size = 10^{-2} or 10^{-5}. h is changed to 20 at the start of the simulation. X means that the test resulted in an error.

Exec. environment	**Plain C simulator**		**MicroPython Unix**		**MicroPython ESP32**	
Executable size (KB)	112		843		1,725 (+220 / +15 %)	
Step size	10^{-2}	10^{-5}	10^{-2}	10^{-5}	10^{-2}	10^{-5}
Time (s)	0.001	1.005	0.002	X	0.139	141.309
Memory usage (KB)	1,632	64,032	355	X	21	46

capabilities, allowing domain experts without extensive embedded C programming knowledge to leverage FMI's standardized approach for applications ranging from educational settings to industrial IoT deployments where Python's flexibility and MicroPython's efficiency can significantly accelerate development cycles.

4.3 Cloud vs. Edge Deployment for FMI-Based Simulations

Several studies have investigated the trade-offs between cloud-based and edge-based IoT simulations. (Savaglio et al. 2019) assessed the performance, latency, scalability, and resource utilization of both deployment options. Their findings suggest that edge computing provides lower latency and better scalability for IoT applications, whereas cloud computing offers superior data handling and processing reliability. Recent advancements in edge deployment, such as those proposed by (Bu, Filipau, and Baklanov 2024), have further streamlined the process of deploying cyber-physical models at the edge or in the cloud. Their framework utilizes FMUs as executable binaries and JavaFMI as the simulation engine, encapsulating each model deployment within a microservice. This approach ensures flexibility and scalability, as demonstrated by their examples of a winch controller at the edge and a wireline logging unit simulator in the Azure DevOps pipeline.

For FMI-based simulations, the choice between cloud and edge computing depends on factors such as real-time constraints, network bandwidth, and security considerations. Many real-time embedded applications benefit from edge-based execution, as it allows FMUs to run with minimal latency directly on the device. The classification of hardware types, as outlined by (Bormann et al. 2025), provides further clarity on this decision. Table 2 summarizes the classification of common devices used in IoT simulations and deployments.

For instance, devices like the Raspberry Pi, classified as Class 15, are well-suited for fog computing, while microcontrollers such as the ESP32, classified as Class 10, are ideal for edge computing. This classification helps in aligning the hardware capabilities with the specific requirements of IoT simulations and deployments.

The work of (Jung, Shah, and Weyrich 2018) complements this discussion by proposing a dynamic co-simulation approach for IoT systems, where each component is simulated independently and can dynamically join the co-simulation during runtime. This "Plug-and-Simulate" behavior is particularly advantageous for edge deployments, where heterogeneous devices with varying computational capabilities must interact seamlessly. Their modular framework aligns with the microservice architecture proposed by (Bu, Filipau, and Baklanov 2024), highlighting the importance of flexibility in edge-based FMI deployments.

Table 2. Classification of devices for IoT simulations and deployments by (Bormann et al. 2025)

Device	**Classification**
Arduino Uno	Class 5 (Edge Computing)
ESP32	Class 10 (Edge Computing)
Raspberry Pi	Class 15 (Fog Computing)
Modern Smartphone	Class 16 (Edge/Cloud Computing)
Laptop	Class 17 (Cloud Computing)
High end Home Computer	Class 18 (Cloud Computing)

On the cloud side, (Stelzig and Rodenberg 2023) introduces the concept of Simulation Model as a Service (SMaaS), which leverages cloud infrastructure to deploy, execute, and track simulation models efficiently. While cloud-based solutions excel in handling large-scale simulations and complex data processing, they often face challenges related to latency and real-time responsiveness. This trade-off underscores the need for hybrid approaches, where computationally intensive tasks are offloaded to the cloud, while time-critical operations are handled at the edge.

The integration of these approaches is further supported by (Vanommeslaeghe et al. 2018), which explores the co-simulation of embedded platforms and physics-based models. Their work emphasizes the importance of optimizing hardware architectures to support FMI-based simulations, particularly in edge environments where resource constraints are a significant concern. By combining insights from these studies, it becomes clear that the choice between cloud and edge deployment is not binary but rather a spectrum, where the optimal solution depends on the specific requirements of the application, such as latency tolerance, computational complexity, and scalability

needs.

Building on these deployment insights, our MicroPython FMU implementation presents a good middle ground particularly well-suited for Class 10 devices like the ESP32. By embedding FMU capabilities directly into MicroPython, we enable sophisticated edge-based simulations without requiring cloud connectivity or fog computing resources. This approach addresses the latency concerns highlighted in edge computing research while maintaining the flexibility needed for dynamic co-simulation scenarios. The memory-optimized design of our module responds directly to the resource constraints emphasized in the literature, while its Python interface removes barriers to implementation that typically accompany embedded FMU deployments. This positions our work as a practical enhancement to the "Plug-and-Simulate" paradigm, extending FMI's benefits to resource-constrained IoT devices where traditional deployment methods would be prohibitively complex or performance-intensive.

4.4 Discussion

By combining insights from these studies, it becomes clear that hardware optimization for FMI execution involves a multi-faceted approach, encompassing efficient memory management, real-time scheduling, and hardware-specific optimizations. These advancements are paving the way for more sophisticated FMI-based simulations on resource-constrained devices, enabling their use in a wider range of applications, from industrial automation to autonomous systems. The state of the art in FMI-based simulations for embedded systems demonstrates significant progress across multiple fronts, from portable runtime environments to hardware optimization. The integration of Python-based FMUs with containerization technologies, such as Docker, has enhanced portability and reproducibility, enabling seamless deployment across diverse environments. This is particularly valuable for co-simulation scenarios, where modular and dynamic frameworks, as proposed by (Jung, Shah, and Weyrich 2018), allow heterogeneous systems to interact efficiently.

Despite these advances, there remain significant limitations to the adoption of MicroPython-based FMU implementations in industrial settings. Professional environments typically demand rigorous certification, long-term support, and deterministic performance guarantees that interpreted languages like Python may struggle to provide. While our implementation offers valuable prototyping capabilities, industry practitioners will likely continue favoring native C/C++ implementations for production systems where reliability and performance predictability are paramount.

The development of eFMI has further expanded the applicability of FMI in resource-constrained environments, enabling high-fidelity physics-based models to run on microcontrollers. This advancement is particularly transformative for automotive and industrial applications, where real-time execution and compliance with standards like AUTOSAR are critical. The successful integration of eFMI into automotive ECUs, as demonstrated by (Armugham et al. 2021), highlights its potential to bridge the gap between simulation and real-world execution. eFMI being compatible with FMI, we expect that our framework could be used to run FMUs optimized for resource-constrained environments thanks to GALEC.

Our current implementation focuses exclusively on FMI 2.0 standard compatibility, which presents both advantages and limitations. While FMI 2.0 offers widespread tool support and established functionality, it lacks newer features from FMI 3.0 such as binary co-simulation interfaces and enhanced array variable handling that could potentially improve performance on constrained devices. Future work could explore adapting our approach to leverage these newer standard capabilities as tool support matures.

The debate between cloud and edge deployment for FMI-based simulations underscores the importance of balancing computational power, latency, and scalability. While cloud-based solutions excel in handling large-scale simulations, edge computing offers the low latency and real-time responsiveness required for many embedded applications. Hybrid approaches, combining the strengths of both paradigms, are emerging as a promising solution for complex, distributed systems. Our choice of the ESP32 platform was strategically motivated by its unique combination of computational capability, memory resources, and wide adoption in the IoT ecosystem. With dual-core processing up to 240MHz and typically 4MB of flash memory, it represents an optimal middle ground between severely constrained Class 1-5 devices and more powerful Class 15+ systems. Furthermore, the ESP32's native support for both WiFi and Bluetooth positions it ideally for co-simulation scenarios where edge devices must communicate with other simulation components.

As the field continues to evolve, the convergence of these technologies—portable runtime environments, eFMI, cloud-edge hybrid architectures—will play a pivotal role in addressing the challenges of real-time execution, resource constraints, and system integration.

5 Conclusions

This paper introduced ufmu, a lightweight library that enables FMU simulations to run efficiently on resource-constrained microcontrollers. By integrating and executing FMU models within the MicroPython environment, ufmu provides an accessible and flexible FMU tool that may prove useful for education and rapid prototyping in embedded systems.

The library achieves this by translating FMU model descriptions into a C-based structure and embedding them within the MicroPython build process. This approach ensures seamless execution while exposing a minimal high-level Python API for user interaction. As a result, ufmu allows developers to run model-based simulations directly

DOI
10.3384/ecp218535

on microcontrollers without relying on a full desktop or cloud computing setup.

While the current implementation successfully enables FMU simulations to run on an ESP32 microcontroller, there are several areas for future improvement to enhance the library's functionality, usability, and performance.

As a natural progression of this work, adding support for the FMI 3.0 standard is a key objective. The current implementation supports basic simulation execution but supporting advanced states such as setup, pause, and resume modes, would be desirable. For example, users could pause a simulation to modify parameters and then resume execution without restarting the simulation.

Future development will also focus on expanding support to other microcontroller platforms, such as STM32 or Raspberry Pi Pico.

References

Aivaliotis, Panagiotis. et al. (2019). "Methodology for enabling Digital Twin using advanced physics-based modelling in predictive maintenance". In: *Procedia CIRP* 81, pp. 417–422. ISSN: 22128271. DOI: 10.1016/j.procir.2019.03.072. (Visited on 2023-10-08).

Armugham, Siva Sankar et al. (2021). *eFMI (FMI for Embedded Systems) in AUTOSAR for Next Generation Automotive Software Development*. URL: https://www.sae.org/publications/technical-papers/content/2021-26-0048/ (visited on 2025-02-19).

Blockwitz, Torsten et al. (2012). "Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models". en. In: *Functional Mockup Interface 2.0*. DOI: 10.3384/ecp12076173. URL: https://ep.liu.se/en/conference-article.aspx?series=ecp&issue=76&Article_No=17.

Bormann, Carsten et al. (2025-01). *Terminology for Constrained-Node Networks*. Internet Draft draft-ietf-iotops-7228bis-01. Num Pages: 28. Internet Engineering Task Force. URL: https://datatracker.ietf.org/doc/draft-ietf-iotops-7228bis (visited on 2025-02-20).

Bu, Fanping, Mikalai Filipau, and Nikolay Baklanov (2024). "Advanced Edge Deployment: Abstracting Cyber-Physical Models via FMU Mastery". en. In: *Modelica Conferences*, pp. 170–177. ISSN: 1650-3740. DOI: 10.3384/ecp207170. URL: https://www.ecp.ep.liu.se/index.php/modelica/article/view/1142 (visited on 2025-02-19).

Duraiswami, Ramani and Dmitry N. Zotkin (2016-10-01). "Efficient physics based simulation of spatial audio for virtual and augmented reality". In: *Journal of the Acoustical Society of America* 140.4. Number: 4_Supplement, pp. 2999–3000. ISSN: 0001-4966, 1520-8524. DOI: 10.1121/1.4969294. (Visited on 2023-10-08).

Fangohr, Hans (2004). "A comparison of C, MATLAB, and Python as teaching languages in engineering". In: *Computational Science-ICCS 2004: 4th International Conference, Kraków, Poland, June 6-9, 2004, Proceedings, Part IV 4*. Springer, pp. 1210–1217.

Gall, Leo et al. (2021-09). "Continuous Development and Management of Credible Modelica Models". en. In: *Modelica Conferences*, pp. 359–372. ISSN: 1650-3740. DOI: 10.3384/ecp21181359. URL: https://www.ecp.ep.liu.se/index.php/modelica/article/view/214 (visited on 2025-02-19).

George, Damien P. and the MicroPython community (2014). *MicroPython*. Ed. by George Robotics Limited. https://micropython.org/. Accessed: 2025-07-30.

Gundermann, Julia et al. (2017-07-04). "The Embedded Simulation via FMI and its Application to the Simulation of Lifetime Tests Including Wear". In: The 12th International Modelica Conference, Prague, Czech Republic, May 15-17, 2017, pp. 541–545. DOI: 10.3384/ecp17132541. (Visited on 2023-10-09).

Hong, Seokjoon et al. (2020-01-01). "F-DCS: FMI-Based Distributed CPS Simulation Framework with a Redundancy Reduction Algorithm". In: *Sensors* 20.1. Number: 1, p. 252. ISSN: 1424-8220. DOI: 10.3390/s20010252. (Visited on 2023-10-08).

Jung, Tobias, Payal Shah, and Michael Weyrich (2018). "Dynamic Co-Simulation of Internet-of-Things-Components using a Multi-Agent-System". In: *Procedia CIRP* 72, pp. 874–879. ISSN: 22128271. DOI: 10.1016/j.procir.2018.03.084. (Visited on 2023-10-08).

Lenord, Oliver et al. (2021-09-27). "eFMI: An open standard for physical models in embedded software". In: 14th Modelica Conference 2021, pp. 57–71. DOI: 10.3384/ecp2118157. (Visited on 2023-10-02).

Liu, C. Karen and Dan Negrut (2021-05-03). "The Role of Physics-Based Simulators in Robotics". In: *Annual Review of Control, Robotics, and Autonomous Systems* 4.1. Number: 1, pp. 35–58. ISSN: 2573-5144, 2573-5144. DOI: 10.1146/annurev-control-072220-093055. (Visited on 2023-10-02).

Magargle, Ryan et al. (2017-07-04). "A Simulation-Based Digital Twin for Model-Driven Health Monitoring and Predictive Maintenance of an Automotive Braking System". In: The 12th International Modelica Conference, Prague, Czech Republic, May 15-17, 2017, pp. 35–46. DOI: 10.3384/ecp1713235. (Visited on 2023-12-04).

Mikelsons, Lars and Roland Samlaus (2017-07-04). "Towards Virtual Validation of ECU Software using FMI". In: The 12th International Modelica Conference, Prague, Czech Republic, May 15-17, 2017, pp. 307–311. DOI: 10.3384/ecp17132307. (Visited on 2023-10-15).

Saad, Omar M. et al. (2023-12-01). "Earthquake Forecasting Using Big Data and Artificial Intelligence: A 30-Week Real-Time Case Study in China". In: *Bulletin of the Seismological Society of America* 113.6, pp. 2461–2478. ISSN: 0037-1106, 1943-3573. DOI: 10.1785/0120230031. (Visited on 2023-12-04).

Savaglio, Claudio et al. (2019-04). "IoT Services Deployment over Edge vs Cloud Systems: a Simulation-based Analysis". In: *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). Paris, France: IEEE, pp. 554–559. ISBN: 978-1-72811-878-9. DOI: 10.1109/INFCOMW.2019.8845305. (Visited on 2023-10-02).

Schranz, Thomas et al. (2021-09). "Portable runtime environments for Python-based FMUs: Adding Docker support to UniFMU". en. In: *Modelica Conferences*, pp. 419–424. ISSN: 1650-3740. DOI: 10.3384/ecp21181419. URL: https://www.ecp.ep.liu.se/index.php/modelica/article/view/220 (visited on 2025-02-19).

Stelzig, Philipp Emanuel and Benjamin Rodenberg (2023-12). "Simulation Model as a Service (SMaaS): A concept for integrated deployment, execution and tracking of system simulation models". en. In: *Modelica Conferences*, pp. 33–42. ISSN:

1650-3740. DOI: 10.3384/ecp20433. URL: https://ecp.ep.liu.se/index.php/modelica/article/view/911 (visited on 2025-02-19).

Tavella, Jean-Philippe et al. (2016-09). "Toward an accurate and fast hybrid multi-simulation with the FMI-CS standard". In: *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*. 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA). Berlin, Germany: IEEE, pp. 1–5. ISBN: 978-1-5090-1314-2. DOI: 10.1109/ETFA.2016.7733616. (Visited on 2023-10-08).

Tollervey, Nicholas H. (2017). *Programming with MicroPython: Embedded Programming with Microcontroller and Python*. First edition. Beijing Boston Farnham: O'Reilly. 195 pp. ISBN: 978-1-4919-7273-1.

Vanommeslaeghe, Yon et al. (2018). "Towards Co-Simulation of Embedded Platforms and Physics-Based Models". In: *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. ISSN: 978-1-5386-7383-6.

Zoting, Shivani (2023). *Simulation Software Market Size, Trends, Growth, Report 2030*. URL: precedenceresearch.com/simulation-software-market (visited on 2023-12-01).

A ufmu class and methods description

`simulation_instance`
Class representing a configured simulation instance:

- Implements iterator protocol for step-by-step simulation
- Maintains internal simulation state

`simulate(tStart, tEnd, h)`
Executes a complete simulation in one call.

- *Parameters:* `tStart`, `tEnd`, `h` (float): start time, end time, step size
- *Returns:* list of output value tuples

`setup_simulation(tStart, tEnd, h)`
Creates a simulation instance for step-by-step execution.

- *Parameters:* same as above
- *Returns:* a `simulation_instance` object

`get_variable_count()`
Returns the number of variables in the model.

- *Returns:* integer count

`get_variables_names(...)`
Retrieves variable names.

- *Parameters:* optional indices/names
- *Returns:* tuple of variable name strings

`get_variables_base_values(...)`
Retrieves initial values of variables.

- *Parameters:* optional indices/names
- *Returns:* tuple of values (float/int)

`get_variables_description(...)`
Retrieves variable descriptions.

- *Parameters:* optional indices/names
- *Returns:* tuple of variable description strings

`change_variable_value(...)`
Modifies a variable's value during simulation.

- *Parameters:*
  - `sim_instance` (object)
  - `var_ref` (int or str)
  - `value` (float)
- *Returns:* boolean success indicator

FMI Meets SystemC: A Framework for Cross-Tool Virtual Prototyping

Nils Bosbach[1] Meik Schmidt[1] Lukas Jünger[2] Matthias Berthold[3] Rainer Leupers[1]

[1]RWTH Aachen University, Aachen, Germany, {bosbach,schmidtm,leupers}@ice.rwth-aachen.de
[2]MachineWare GmbH, Aachen, Germany, lukas@mwa.re
[3]tracetronic GmbH, Dresden, Germany, Matthias.Berthold@tracetronic.de

Abstract

As systems become more complex, the demand for thorough testing and virtual prototyping grows. To simulate whole systems, multiple tools are usually needed to cover different parts. These parts include the hardware of a system and the environment with which the system interacts. The Functional Mock-up Interface (FMI) standard for co-simulation can be used to connect these tools.

The control part of modern systems is usually a computing unit, such as a System-on-a-Chip (SoC) or Microcontroller Unit (MCU), which executes software from a connected memory and interacts with peripherals. To develop software without requiring access to physical hardware, full-system simulators, the so-called Virtual Platforms (VPs), are commonly used. The IEEE-standardized framework for VP development is SystemC TLM. SystemC provides interfaces and concepts that enable modular design and model exchange. However, SystemC lacks native FMI support, which limits the integration into broader co-simulation environments.

This paper presents a novel framework to control and interact with SystemC-based VPs using the FMI. We present a case study showing how a simulated temperature sensor in a SystemC simulation can obtain temperature values from an external tool via FMI. This approach allows the unmodified target software to run on the VP and receive realistic environmental input data such as temperature, velocity, or acceleration values from other tools. Thus, extensive software testing and verification is enabled. By having tests ready and the software pre-tested using a VP once the physical hardware is available, certifications like ISO 26262 can be done earlier.

Keywords: SystemC, TLM, FMI, Virtual Platform

1 Introduction

In today's rapidly evolving technology landscape, systems are becoming more complex, requiring advanced development cycles and rigorous testing methodologies. As industries, such as automotive, tend to solve more problems in software rather than hardware, early and intensive software testing is paramount (Ondrej Burkacky et al. 2018). To start software development and testing while the hardware is still under design, Virtual Platforms (VPs), also known as *Level 4 Virtual Electronic Control Units (L4 vECUs)*, can be used. It has been shown that the earlier bugs are found, the lower the cost of rectifying them. Fixing a bug during the development phase can be up to 100 times cheaper than fixing it while the product is already in use (Boehm 1976). VPs are the key enabler of early software testing. By simulating a System-on-a-Chip (SoC), the unmodified target[1] software can be developed and tested without requiring access to the physical hardware.

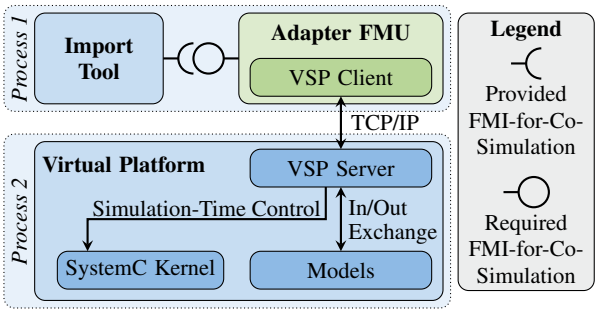


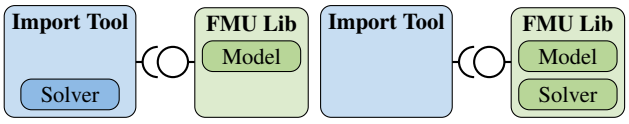
Figure 1. SystemC-FMU adapter approach.

The IEEE-standardized framework for VP design is SystemC with its Transaction-Level Modeling (TLM) extension (IEEE Standards Association and others 2023). While VPs are well-suited tools for software development and verification, they reach their limits when it comes to realistic testing of different scenarios. Because embedded systems, such as Electronic Control Units (ECUs), interact with their physical environment by reading sensor values and controlling devices, it is not enough to simulate only the system itself. In addition, environmental effects must also be considered and simulated.

To simulate physical systems and define complex test cases, there are many tools available. In this paper, we show how a SystemC-based simulation can be connected to those tools using the Functional Mock-up Interface (FMI) standard (Modelica Association 2024). The FMI co-simulation standard allows to synchronize the simulation time of multiple simulations and to exchange data.

Figure 1 shows the integration of our approach in a co-simulation. We developed an Functional Mock-up Unit (FMU) that can be imported by any FMI-compatible tool to control a VP. This FMU exposes the specified output of

[1]*target*: architecture that is simulated; *host*: architecture that runs the simulation



(a) Model Exchange. **(b)** Co-Simulation.

Figure 2. FMI types.

VP and accepts the inputs that are forwarded to VP. The VP runs in a second process, either on the same host as the FMI import tool, or on a different one. The connection between the FMU and the VP is TCP/IP based.

We present a case study where we show how an open-source VP can be connected to tracetronic's *ecu.test* tool using FMI. ecu.test allows defining complex test scenario where environmental data can be specified and responses of the tested system are compared against expectations.

In the end, we propose a software design flow that uses our integration for early testing and verification. This design flow helps to shorten the time to reach specifications like ISO 26262 (ISO/TC 22/SC 32 2018).

2 Background and Related Work

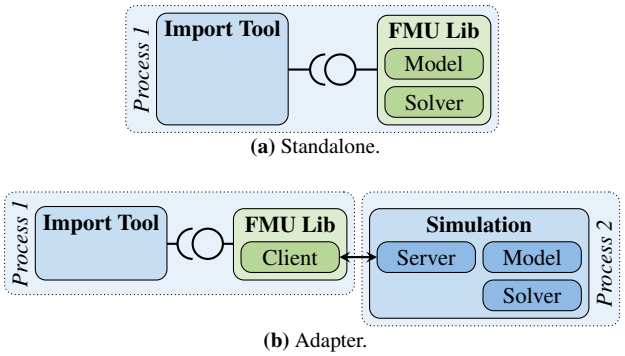
This section provides an overview of the related work that forms the foundation of this study. Section 2.1 explains the FMI standard and the features used in this work. Section 2.2 provides an overview of virtual prototyping, including background information on the industry-standard framework SystemC and the Virtual Components Modeling Library (VCML), which builds on top of SystemC. In Section 2.3, an overview of existing approaches to integrate SystemC modules into an FMI co-simulation is given.

2.1 Functional Mock-up Interface (FMI)

The Functional Mock-up Interface (FMI) is an open standard designed to enable seamless integration and interoperability of simulation models from various tools and vendors (Modelica Association and MODELISAR Consortium 2024). Initially introduced by the Modelica Association in 2010, FMI has become a widely adopted standard for modular and tool-agnostic simulation workflows and is supported by more than 200 tools.

Simulation models are encapsulated into so-called FMUs. This encapsulation process is performed by *export tools*, which are typically provided by simulation-software vendors. FMUs can then be imported and utilized by *import tools*, which coordinate the simulation and facilitate data exchange between multiple FMUs.

The original standard supports two interface types, *Model Exchange* (Modelica Association 2010b) and *Co-Simulation* (Modelica Association 2010a), as illustrated in Figure 2. As physical systems are often described by differential equations, a numerical solver is needed to solve and evaluate these equations. The difference between the FMU interface types is primarily in the placement of the numerical solver. In the Model-Exchange use-case, the solver is not part of the FMU and therefore needs to be implemented in the import tool.



(a) Standalone.

(b) Adapter.

Figure 3. FMI Co-Simulation scenarios.

With the introduction of FMI 3.0 (Modelica Association and MODELISAR Consortium 2024), a third interface type, *Scheduled Execution*, was added. For this work, we only use the Co-Simulation interface, so the other ones are not further discussed. We demonstrate how a full VP can be encapsulated into an FMU. Because VPs usually operate at a much higher level of abstraction than differential equations, we do not require a numerical solver. Consequently, Co-Simulation is the preferred approach.

When the Co-Simulation interface is used, two scenarios are possible, a *standalone* and an *adapter* scenario (Blochwitz et al. 2011). Both scenarios are visualized in Figure 3. The FMU contains a shared library (*FMU Lib*) that is loaded by the import tool. It implements standardized FMI functions, the import tool calls. In the standalone scenario, as shown in Figure 3a, the shared library contains the full simulation model. The simulation model can be directly executed in the process of the import tool.

In the adapter scenario, illustrated in Figure 3b, the shared library acts as an interface to another process that executes the simulation. This connection can, e.g., be established using network-based protocols, such as TCP/IP. One key advantage of this approach is the ability to run the import tool and simulation on different machines, which can offer potential performance benefits by distributing computational load. Additionally, this configuration allows the simulation and import tool to run on different Operating Systems (OSs) or even different hardware architectures, such as x86 and Arm.

2.2 Virtual Prototyping

Virtual prototyping is a technique in which electronic systems are modeled entirely in software. This work focuses on VPs, virtual prototypes designed for software development for the simulated target platform. VPs can simulate the execution of an unmodified target-software stack. In the automotive domain, they are referred to as *L4 vECUs*.

The industry-standard framework for VP development is SystemC with its TLM extension (IEEE Standards Association and others 2023). SystemC standardizes the interface and functionality of a C++ library that features interfaces, data types, and a Discrete Event Simulation

DOI
10.3384/ecp218545

```
<fmiModelDescription ... fmiVersion="3.0" modelName="myVP" ...>
    <CoSimulation modelIdentifier="myVP" needsExecutionTool="true"
    canHandleVariableCommunicationStepSize="true" ... />
    <DefaultExperiment startTime="3" stopTime="5" stepSize="0.01"/>
    <ModelVariables>
        <Float64 name="time" valueReference="0" causality="independent"
    variability="continuous" />
        <Float32 name="system.max31855.temp" valueReference="1" causality="input"
    variability="continuous" start="10.0" />
        <UInt32 name="system.gpio.data" valueReference="2" causality="output"
    variability="discrete" />
    </ModelVariables>
    <ModelStructure>
        <InitialUnknown valueReference="1"/>
        <Output valueReference="2"/>
    </ModelStructure>
     <Annotations>
        <Annotation type="VCML">
            <VP host="localhost" port="8888"  executable="resources/vp" ... />
        </Annotation>
      </Annotations>
</fmiModelDescription>
```

Listing 1. Model-description file.

(DES)-based scheduler to simulate parallelism. A simulation usually contains multiple modules that can be connected via TLM sockets. TLM abstracts the underlying communication protocol and offers low-overhead data exchange. The standardized interfaces guarantee seamless integration of different models into a simulation.

On top of SystemC, modeling libraries provide additional functionalities. One open-source modeling library is VCML (MachineWare 2024). VCML extends SystemC with essential components, such as register models, prebuilt hardware components, and specialized TLM-based communication protocols. A key feature used in this work is the VCML Session Protocol (VSP). VSP is a remote-control protocol that is based on GDB's Remote Serial Protocol (RSP) protocol (Free Software Foundation, Inc. 2024). VCML includes a VSP server module, which allows clients to connect via TCP to control the simulation.

Another crucial VCML feature used in this work is *properties*, which encapsulate configuration variables within modules. These properties can be set at VP launch and accessed via the VSP protocol, facilitating dynamic reconfiguration.

2.3 Related Work

Previously, various publications have described methods for integrating SystemC-based simulations into FMI co-simulations. Most of these approaches are based on the FMI 2 standard and involve tight coupling between the FMI interface and the VP.

One approach integrates an FMI import tool directly into the VP to co-simulate SystemC models with FMUs that provide environmental models (Safar et al. 2018). This approach means that the VP cannot be used with other import tools, e.g., for setting up test scenarios, as described in Section 4. Furthermore, the VP is no longer standalone, so co-simulated FMUs are always required.

For detailed Register-Transfer Level (RTL)-like SystemC simulations, a different approach can be used to expose SystemC ports to FMI (Centomo, Deantoni, and De Simone 2016). This approach also requires modifications to the VP, which prevents a standalone simulation. In our work, we focus on SystemC-TLM-based VPs that use a higher abstraction level to achieve increased performance.

For SystemC Analog/Mixed-Signal (AMS) simulations, wrapper classes can be created to wrap models and connect the needed inputs and outputs to an FMI integration (Krammer et al. 2015). This kind of simulation also targets more detailed abstraction levels and requires modifications to the simulation.

In another approach, *FMI variables* are introduced, that can be bound to different protocols like CAN, or I^2C (Saidi et al. 2019). When a variable changes, different actions can be carried out. For example, a CAN frame carrying the updated variable as payload can be injected onto the bus. This approach is useful for simple sensors that use common protocols. An advantage is that the sensor itself does not need to be part of the VP but can be completely used from the FMI integration. This approach also does not allow for standalone use of the VP.

In our work, we present an approach that does not require VP modifications and therefore no source-code access. The requirement is that theVP is based on the used modeling library VCML and the VSP server is not disabled. The VP can then be used without the FMI integration when environmental input are not needed. When environmental inputs become important, the VP can be integrated into an FMI co-simulation without any changes by using our standalone FMI-adapter FMU.

3 Approach

In this section, we present our approach. First, Section 3.1 describes the application scenario. Then, Section 3.2 explains how we integrate a VCML-based VP into an FMU. In the end, Section 3.3 provides further details on the FMU

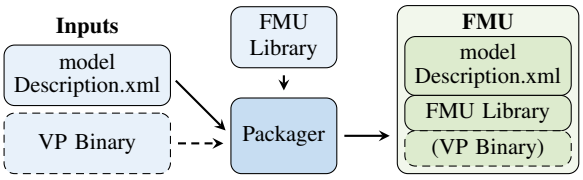


Figure 4. FMU creation.

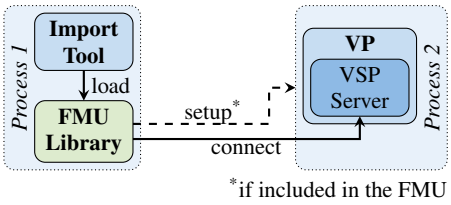


Figure 5. Working principle.

implementation and the integration process.

3.1 Application Scenario

Imagine a VP that models an ECU that consists of various peripherals including sensing devices for measuring environmental conditions. Such a VP is crucial for target software development and verification but traditionally lacks dynamic environmental modeling capabilities. For instance, a virtual model of a temperature sensor simulates all hardware interfaces other components can access but typically returns constant values during standalone VP execution, limiting its ability to test software responses to varying conditions. Directly integrating environmental simulations, such as temperature models, into device models like sensors would have multiple drawbacks:

- **Simplicity**: The model should be kept simple to have a high performance. Complex algorithms to model the environment would unnecessarily slow down the model and increase its complexity when the environmental simulation is not needed.
- **Genericity**: The model should only reflect the behavior of the hardware to use it in various scenarios. An environmental simulation model heavily depends on the scenario and not the component itself.
- **Specialization of Tools**: There are well-established tools to create environmental models, e.g., based on physical equations or test cases.

Because of those reasons, the returned temperature value from our model is read from a VCML property (see Section 2.2). This means that in a standalone simulation, the temperature value is set at startup and remains constant throughout execution. While this setup allows for full software stack execution including sensor interaction, it does not support testing of how the software reacts to changing environmental conditions.

To overcome this limitation, our FMI integration enables co-simulation of the VP together with external tools that dynamically provide environmental data. This approach allows for realistic scenario testing and a more comprehensive evaluation of the system's behavior using dedicated tools to solve various tasks.

3.2 FMU Creation

To create an FMU from a VCML-based VP, our *Packager* tool can be used. Figure 4 illustrates the creation process. The resulting FMU includes our VP-independent FMU library and a model-description file, which defines its properties. Section 3.3 explains the working principle of the FMU library in detail. A simplified example of a model-description file is shown in Listing 1.

The `ModelVariables` element lists the VP's inputs and outputs that are accessible via FMI. Each variable's name corresponds to the property name within the module hierarchy of the VP. For example, the `system.max31855.temp` property defines the property `temp` of the SPI temperature sensor `max31855`, which is a submodule of the `system` module. It is sufficient to list the hierarchical name of the property in the model-description file. There are no changes in the VP needed to expose the property. The VSP server can clearly identify and access the properties by their hierarchical names. Additionally, data types are specified, and it is indicated whether the property functions as an input or output. For inputs, a start value can be assigned. During simulation, FMI uses the `valueReference` numbers when updating the variables.

The `Annotations` element defines VP-specific properties using the available attributes listed in Table 1. The VP binary can be embedded within the FMU and launched by the FMU. In this case, the `executable` attribute points to the relative path of the VP binary within the FMU (e.g., `resources/my_vp`). Additionally, the `host` is set to *localhost*, and the `port` specifies the TCP port the VP's VSP server is listening.

If the VP is not embedded in the FMU, e.g., because it is manually launched on a separate machine, only the `host` and `port` attributes need to be configured. The `host` attribute defines the hostname of the machine that executes the VP. This machine needs to be reachable by the import tool via a TCP connection.

3.3 Working Principle

An exported FMU can be loaded by an FMI-3-compatible import tool. Figure 5 illustrates the workflow.

Table 1. Attributes of the VCML annotation node.

Attribute	Meaning
`executable`	Path of the VP executable
`args`	Args that are passed to the VP executable during startup
`host`	IP address or hostname of the machine that executes the VP
`port`	Listening port of the VP server

DOI
10.3384/ecp218545

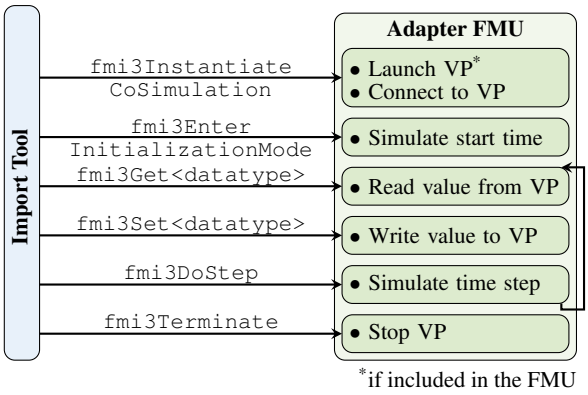


Figure 6. FMI functions implemented by our FMU.

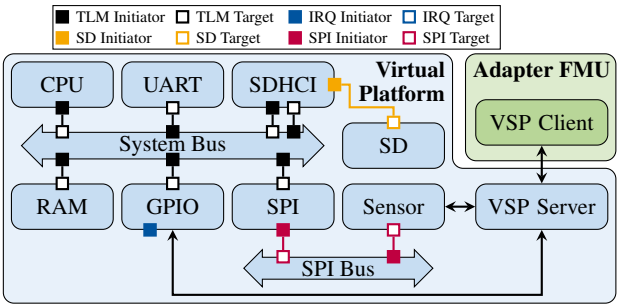


Figure 7. AVP64-based ECU with connected FMI adapter.

The import tool unpacks the FMU and reads the model-description file. It then loads the FMU library (*.dll* or *.so*). Afterward, the import tool can call the standardized FMI functions that are implemented by the FMU library. If the VP binary is packed into the FMU, the FMU library starts the VP in a second process. The library establishes a TCP-based connection to the VSP server of the VP and uses this connection to control the simulation.

Figure 6 shows the FMI functions our FMU implementation. The tasks that are executed by those functions are explained in the following in more detail.

First, the `fmi3InstantiateCoSimulation` function is usually called. If the VP is part of the FMU, a second process is started to run the VP. Parameters can be passed to the VP via command-line arguments. If the VP is not part of the FMU, it needs to be started manually. This can be done either on the same machine or on a different one that is located in the same network as the machine that runs the FMU. In this case, the corresponding hostname of the machine that runs the VP needs to be set in the `VCML` node of the model description's `Attribute` node. Once the VP is started, the VSP server listens on the TCP port for a connection and suspends the VP until the FMU is connected. The FMU connects the VSP server of the VP.

When the `fmi3EnterInitializationMode` function is called by the import tool, a command is sent to the VSP server to simulate until the virtual simulation time of the SystemC simulation reaches the specified `startTime`. The start time parameter of the FMU can be used to, e.g., skip the boot process of the OS running on the system. If the start time is set to 0, our FMU does nothing in this step. After the `fmi3EnterInitializationMode` function call, the VP is ready for simulation.

The import tool can read model variables that have been declared as *output* in the model description by calling the `fmi3Get<datatype>` functions. For example, `fmi3GetFloat64` is used when the variable is declared as *Float64* in the model description. The variable is referenced by its `valueReference` property. The FMU translates the `valueReference` into the `name` using the model description. This name corresponds to the hierarchical name of the property in the VP (see Section 3.2). This hierarchical name can be directly used by the VSP server to identify, read, and write the property value within the simulation. All available properties of the simulation can be used as FMU inputs and outputs by simply specifying their hierarchical name in model description.

To update an FMU input, the `fmi3Set<datatype>` function can be used in the same way. Again, the variables are referenced by their `valueReference` property. The updated input values are directly sent to the VP to update the corresponding model properties.

After variables have been updated, the `fmi3DoStep` function can be called to simulate a time step. The amount of simulation time that is simulated on a call is usually defined by the `stepSize` attribute in the model description (see Listing 1). Our FMU sends a command to the connected VP to simulate the step. The VP's SystemC scheduler simulates events until the virtual simulation time is increased by the specified step. This allows the virtual CPU model to execute target software, and interact with other simulated peripherals. The import tool repeatedly calls the `fmi3Get<datatype>`, `fmi3Set<datatype>`, and `fmi3DoStep` functions as long as the test or the simulation is running.

To end the simulation, the `fmi3Terminate` function is called, which stops the VP and shuts down the simulation.

4 Case Study

To demonstrate the effectiveness of our FMI integration approach for SystemC-based VPs, we present a case study of a software-based Schmitt trigger application with a temperature input. We show how the different components of the system can be efficiently modeled and integrated into an FMI-based co-simulation using our developed framework. The full system consists of three parts:

ECU: The ECU consists of an ARMv8 processor, memory, and peripherals. The components and their connections are shown in Figure 7. A *MAX31855* (Maxim Integrated 2015) temperature sensor (denoted as *Sensor* in Figure 7) can measure the ambient temperature. The sensor is connected via an SPI interface to the SPI controller, which is accessible by the CPU. Additionally, the system features a General Purpose Input/Output (GPIO) controller that allows the CPU to control external signals. To simulate the ECU hardware, the open-source, SystemC-TLM-based ARMv8 Virtual Platform (AVP64) (Jünger et

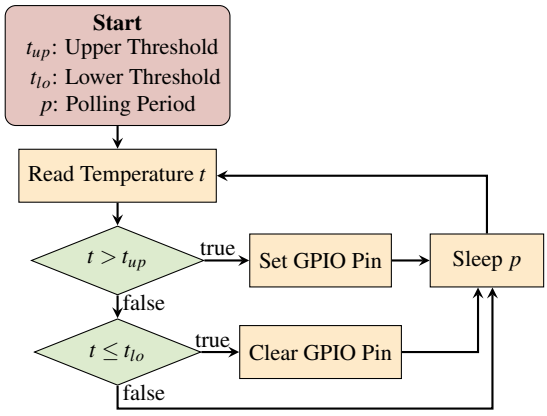


Figure 8. Schmitt-trigger algorithm executed by the VP.

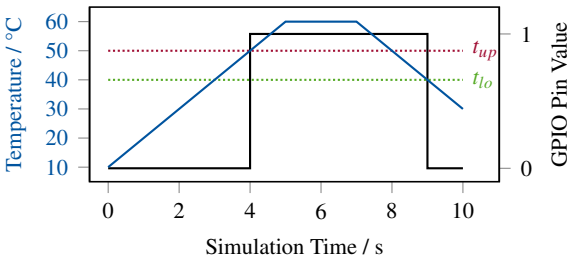


Figure 9. Temperature input and expected GPIO-pin output for $t_{lo} = 40\,°\text{C}$ and $t_{up} = 50\,°\text{C}$.

al. 2019; Jünger 2023) is used. The VP is capable of executing the full target-software stack.

Software Application: The ECU boots a *Buildroot*-based Linux OS (Buildroot 2024). Upon completion of the boot process, a simple Schmitt-trigger application is automatically launched. The Schmitt-trigger application implements the algorithm depicted in Figure 8. This algorithm monitors the temperature input of the sensor and sets or clears a GPIO pin based on the predefined threshold values t_{lo} and t_{up}. When the temperature exceeds t_{up}, the GPIO pin is set. When it falls below t_{lo}, the GPIO pin is cleared. The application periodically polls the temperature sensor at a configurable interval p. An example of a temperature curve and the expected output for $t_{lo} = 40\,°\text{C}$ and $t_{up} = 50\,°\text{C}$ is shown in Figure 9.

Environmental Impacts: The ECU interacts with its environment by measuring the ambient temperature and reacting to changes. To add the simulation of changing environmental conditions and enable comprehensive testing with dynamic environmental scenarios, we package the VP into an FMU using our FMI integration approach.

This allows for co-simulation with external tools that can provide varying temperature inputs. The FMU configuration for our scenario includes one input and one output as shown in Figure 10. The input is connected to the temperature property of the MAX31855 sensor model, and the output retrieves the GPIO pin value from the GPIO controller (see Figure 7). There are no changes in the VP required to be connected to the FMU. It is sufficient to specify the hierarchical names of the properties for the temperature value and the GPIO pins in the model description (see Section 3.2).

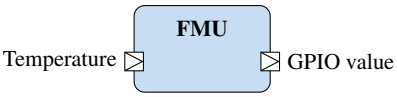


Figure 10. FMU with inputs and outputs.

4.1 Target-Software Verification

Software verification and testing play a crucial role during the design process of a system. The earlier a bug is found, the less effort and cost it will take to fix it. As described in Section 1, if a bug is first found during product operation, the cost of fixing it can be up to 100 times higher than if it is found during software development (Boehm 1976). By integrating the VP into an FMI co-simulation using our approach, complex scenarios can be tested to find bugs as early as possible. This leads to lower costs, a shorter time to market, and overall improved development efficiency.

A typical measure that is used to determine the percentage of tested code is code coverage (Ivanković et al. 2019). Traditional line coverage (Miller and Maloney 1963) describes the number of executed lines of code divided by the total number of lines of code. More advanced metrics like Modified Condition/Decision Coverage (MC/DC) (Kelly J. Hayhurst et al. 2001) are required for ISO-26262-based Automotive Safety Integrity Level (ASIL) D certification (ISO/TC 22/SC 32 2018). Besides counting the executed lines of code, MC/DC adds additional metrics like counting if Boolean expressions have been evaluated as both, true and false. The VP can directly extract coverage metrics of the target software after execution without requiring instrumentation of the target software (Bosbach et al. 2024).

For extensive testing and validation, we utilize *ecu.test* from *tracetronic GmbH* (tracetronic GmbH 2024). This tool allows defining complex test scenarios, specify environmental input data, and compare the system's responses against expected outcomes to test if the system fulfills its requirements. ecu.test is commonly used for ECU testing and verification. It has native support for FMI.

For easy setup and test configuration, ecu.test can be executed on Windows-based machines and controlled using a Graphical User Interface (GUI). The VP is executed on a Linux cloud server. ecu.test loads our adapter FMU on the Windows machine. The FMU connects via a TCP/IP connection to the VSP server that runs on a Linux machine in the same network. For scaled testing, the whole setup can be executed on a single Linux machine using tracetronic's Linux version of ecu.test. By combining our VP FMU with ecu.test, we can perform realistic and thorough testing of the Schmitt trigger application by simulating various temperature conditions. An example of a successful test run is illustrated in Figure 11.

Execution traces of the CPU model can be used to generate coverage reports. To showcase the usefulness of a coverage report, we turned on the code-coverage feature

DOI
10.3384/ecp218545

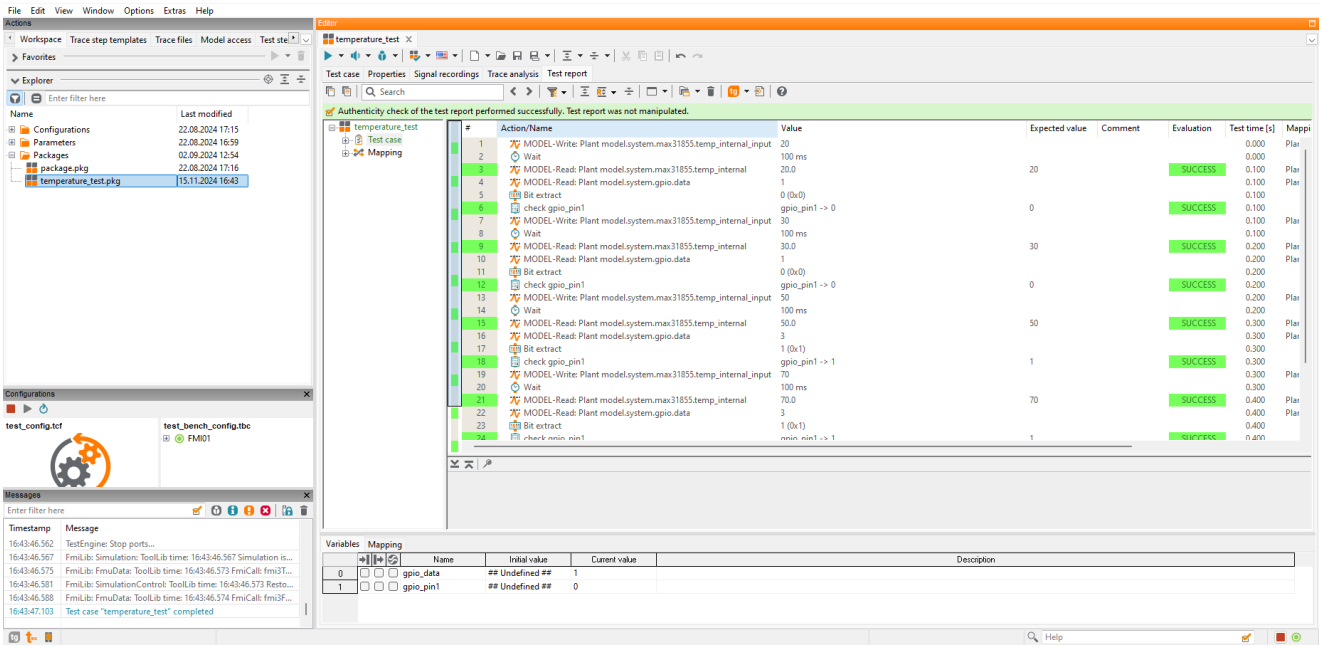
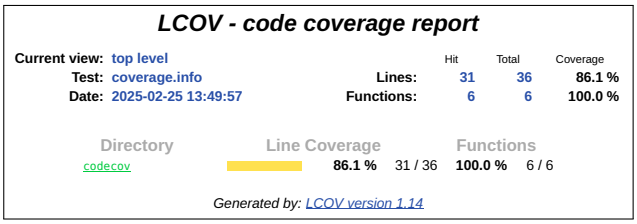
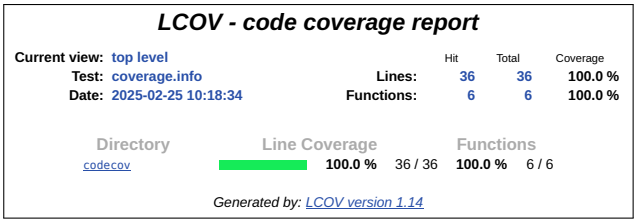


Figure 11. Successful test run in ecu.test.



(a) Constant temperature (standalone VP).



(b) Temperature controlled by ecu.test.

Figure 12. Coverage results for different configurations.

of the VP to generate execution traces and converted them to a *lcov*-based coverage report (Linux Test Project 2023).

Figure 12 visualizes coverage results for the presented Schmitt-trigger application. Figure 12a shows the report for the standalone VP without the FMI integration and ecu.test. Since the temperature value is constant in this case, the algorithm never executes the path that sets the GPIO pin (see Figure 8). This leads to a line coverage of 86.1 %. When ecu.test is connected and test scenarios are executed, the line coverage is increased to 100 %.

This simple example shows that the consideration of environmental influences is necessary for extensive testing. The effectiveness of the applied testing is directly reflected by the increased code coverage.

A more realistic industrial-grade application is, e.g., the simulation of an engine control unit in a VP. The various sensor inputs (e.g., position, temperature, and veclocity sensors) can be read from FMI. Output-signal characteristics, such as the duty cycle and frequency of Pulse Width Modulation (PWM) signals to control actuators like fuel injectors, ignition coils, and idle speed control valves, are forwarded to the importing tool for further processing.

5 Design-Flow Integration

Based on our integration, we propose a design flow that can be used for early target software verification and creation of tests. An overview is shown in Figure 13. At the beginning of the design process, a VP can be created according to the specification of the system. The target software can be developed and tested using this VP. Once the target software development is done, our FMI integration can be used to integrate the VP into a co-simulation and develop test scenarios. Edge-cases can be tested to find bugs in the target software early. The extracted coverage metrics guide verification and test engineers to cover all parts of the target software.

To fulfill standards like ISO 26262 (ISO/TC 22/SC 32 2018), the satisfaction of the requirements needs to be tested. Our FMU can be used to design those tests using a VP. Once physical hardware is available, the hardware can directly be tested using the already developed tests.

In contrast to physical hardware, the virtual model can be scaled arbitrarily, distributed around the world within seconds, and easily integrated into Continuous Integration/Continuous Delivery (CI/CD) flows. CI/CD allows for continuous and automatic testing during the development process to reduce costs by finding bugs earlier.

Once the physical hardware is available, the same test can be executed on the real hardware by replacing the adapter FMU by an hardware-communication interface.

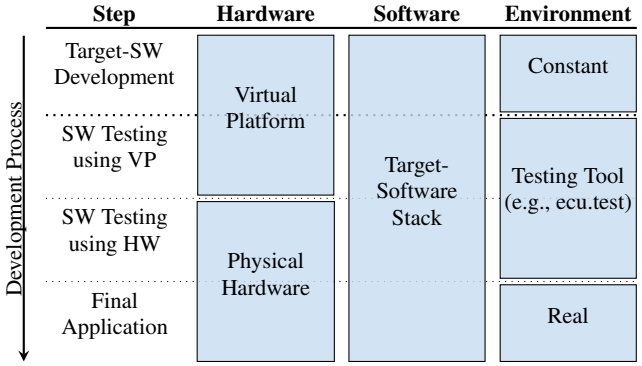


Figure 13. Incorporation of early target-software testing and validation in the design process.

Without our co-simulation integration, the test would have needed to be created directly for the physical hardware, making it possible only later in the development process.

When the target software stack passed all tests on the VP and the physical device, the application can be used in the real world application.

6 Conclusion and Future Work

In this paper, we have presented a novel approach to interface SystemC-based VPs with the FMI standard for co-simulation. By developing an adapter FMU that communicates with the VP via a TCP/IP-based connection, we enable the VP to exchange data with external tools in a co-simulation environment. Our method allows unmodified target software to run on the VP while interacting with realistic environmental inputs provided by other tools.

We demonstrated the effectiveness of our approach through a case study involving a Schmitt-trigger application running on the open-source AVP64. By connecting the VP to the FMI-compatible ecu.test tool, we supplied dynamic temperature inputs and observed the corresponding behavior of the software, achieving comprehensive testing and improved code coverage. This integration facilitates thorough software testing and verification by simulating environmental effects on the system, which is crucial for developing complex embedded systems.

While ecu.test is the leading testing tool in the automotive domain, and thereby a key candidate for evaluating our FMI integration, FMI opens the door to connect various other tools to VPs using our integration. Users have the possibility co-simulate VPs with models created by other tools, e.g., to add realistic physical environmental models to sensors of the VP. Tool providers that support FMI can directly benefit from our integration without the need of extending their tool. Moreover, VP developers can adopt a co-simulation approach to introduce physical environmental models when necessary, while retaining the flexibility to revert to a VP-only simulation if those models are not required.

Since our method only relies on VCML's VSP to establish the TCP/IP connection, it is not limited to a specific VP. There are no changes required for the VP to work with FMI, so other VPs can directly be integrated into an FMI co-simulation.

We showcased how the approach can be integrated into the target-software design process to enable early test development and testing. This speeds up the design process and reduces the number of errors in the product.

For future work, we plan to extend our approach to support more advanced communication between the VP and other FMUs by adding support for the new FMI Layered Standard for Network Communication (FMI-LS-BUS). This addition will allow connecting FMUs via virtual network interfaces like CAN or Ethernet.

References

Blochwitz, Torsten et al. (2011-06). "The Functional Mockup Interface for Tool independent Exchange of Simulation Models". en. In: pp. 105–114. DOI: 10.3384/ecp11063105. URL: https://ep.liu.se/en/conference-article.aspx?series=ecp&issue=63&Article_No=13.

Boehm, Barry W. (1976-12). "Software Engineering". In: *IEEE Transactions on Computers* C-25.12, pp. 1226–1241. ISSN: 1557-9956. DOI: 10.1109/TC.1976.1674590. URL: https://ieeexplore.ieee.org/document/1674590.

Bosbach, Nils et al. (2024-01). "NQC²: A Non-Intrusive QEMU Code Coverage Plugin". en. In: *Proceedings of the 16th Workshop on Rapid Simulation and Performance Evaluation for Design*. Munich Germany: ACM, pp. 16–21. ISBN: 9798400717918. DOI: 10.1145/3642921.3642924. URL: https://dl.acm.org/doi/10.1145/3642921.3642924.

Buildroot (2024). *Making Embedded Linux Easy*. URL: https://buildroot.org/.

Centomo, Stefano, Julien Deantoni, and Robert De Simone (2016-08). "Using SystemC Cyber Models in an FMI Co-Simulation Environment: Results and Proposed FMI Enhancements". In: *2016 Euromicro Conference on Digital System Design (DSD)*, pp. 318–325. DOI: 10.1109/DSD.2016.86. URL: https://ieeexplore.ieee.org/document/7723569.

Free Software Foundation, Inc. (2024). *Remote Protocol (Debugging with GDB)*. URL: https://sourceware.org/gdb/current/onlinedocs/gdb.html/Remote-Protocol.html.

IEEE Standards Association and others (2023-09). "IEEE Standard for Standard SystemC Language Reference Manual". In: *IEEE Std 1666-2023 (Revision of IEEE Std 1666-2011)*, pp. 1–618. DOI: 10.1109/IEEESTD.2023.10246125.

ISO/TC 22/SC 32 (2018). *Road vehicles — Functional safety*. Geneva, Switzerland. URL: https://www.iso.org/standard/68383.html.

Ivanković, Marko et al. (2019-08). "Code coverage at Google". In: *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2019. New York, NY, USA: Association for Computing Machinery, pp. 955–963. ISBN: 978-1-4503-5572-8. DOI: 10.1145/3338906.3340459. URL: https://dl.acm.org/doi/10.1145/3338906.3340459.

Jünger, Lukas (2023-05). *An ARMv8 Virtual Platform (AVP64)*. URL: https://github.com/aut0/avp64.

Jünger, Lukas et al. (2019-01). "Fast SystemC Processor Models with Unicorn". In: *Proceedings of the Rapid Simulation and Performance Evaluation: Methods and Tools*. RAPIDO '19. New York, NY, USA: Association for Computing Machinery,

pp. 1–6. ISBN: 978-1-4503-6260-3. DOI: 10.1145/3300189.3300191.

Kelly J. Hayhurst et al. (2001). *A Practical Tutorial on Modified Condition/Decision Coverage*. en. DIANE Publishing. ISBN: 978-1-4289-9599-4.

Krammer, Martin et al. (2015-09). "Standard compliant co-simulation models for verification of automotive embedded systems". In: *2015 Forum on Specification and Design Languages (FDL)*, pp. 1–8. URL: https://ieeexplore.ieee.org/document/7306083.

Linux Test Project (2023-10). *LTP GCOV extension (LCOV)*. URL: https://github.com/linux-test-project/lcov.

MachineWare (2024-03). *machineware-gmbh/vcml*. URL: https://github.com/machineware-gmbh/vcml.

Maxim Integrated (2015). *MAX31855 Datasheet*. URL: https://www.analog.com/media/en/technical-documentation/data-sheets/max31855.pdf.

Miller, Joan C. and Clifford J. Maloney (1963-02). "Systematic mistake analysis of digital computer programs". In: *Commun. ACM* 6.2, pp. 58–63. ISSN: 0001-0782. DOI: 10.1145/366246.366248. URL: https://dl.acm.org/doi/10.1145/366246.366248.

Modelica Association (2010a-10). *Functional Mock-up Interface for Co-Simulation*. URL: https://fmi-standard.org/assets/releases/FMI_for_CoSimulation_v1.0.pdf.

Modelica Association (2010b-01). *Functional Mock-up Interface for Model Exchange*. URL: https://fmi-standard.org/assets/releases/FMI_for_ModelExchange_v1.0.pdf.

Modelica Association (2024-11). *FMI 3.0.2*. en. URL: https://github.com/modelica/fmi-standard/releases.

Modelica Association and MODELISAR Consortium (2024-11). *Functional Mock-up Interface Specification 3.0.2*. URL: https://fmi-standard.org/docs/3.0.2/.

Ondrej Burkacky et al. (2018-02). "Rethinking car software and electronics architecture". en. In: URL: https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/rethinking-car-software-and-electronics-architecture#/.

Safar, Mona et al. (2018-05). "Virtual Electronic Control Unit as a Functional Mockup Unit for Heterogeneous Systems". In: *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5. DOI: 10.1109/ISCAS.2018.8351357. URL: https://ieeexplore.ieee.org/document/8351357.

Saidi, Salah Eddine et al. (2019). "Fast Virtual Prototyping of Cyber-Physical Systems using SystemC and FMI: ADAS Use Case". In: *Proceedings of the 30th International Workshop on Rapid System Prototyping (RSP'19)*. RSP '19. New York, NY, USA: Association for Computing Machinery, pp. 43–49. ISBN: 978-1-4503-6847-6. DOI: 10.1145/3339985.3358488. URL: https://dl.acm.org/doi/10.1145/3339985.3358488.

tracetronic GmbH (2024). *ecu.test*. URL: https://www.tracetronic.com/products/ecu-test/.

DOI
10.3384/ecp218545

Hybrid Simulation Models for Embedded Applications: A Modelica and eFMI approach

Tobias Kamp[1] Christoff Bürger[2] Johannes Rein[1] Jonathan Brembeck[1]

[1]Institute of Vehicle Concepts, German Aerospace Center, {tobias.kamp, johannes.rein, jonathan.brembeck}@dlr.de
[2]Dassault Systèmes AB, Sweden, christoff.buerger@3ds.com

Abstract

Hybrid simulation models combine physics equations with trainable components to improve simulation results and performance. Physics-enhanced neural ordinary differential equations (PeN-ODE) are a promising type of hybrid models that combine artificial neural networks (NN) with the differential equations of a dynamic system. Dynamic simulation models are often part of embedded control algorithms of cyber-physical systems (CPS); compliance with the safety and real-time requirements of such embedded environments is, however, challenging.

In this work, we propose a workflow to incorporate trained NNs in Modelica models to form hybrid simulation models that are PeN-ODEs. We thereby focus on the transformation steps from equation-based trained PeN-ODEs in Modelica towards causal solutions suited for the embedded domain – up to and including MISRA C:2023 compliance checks and final software-in-the-loop (SiL) tests of generated production code in the modeling environment – for which we leverage eFMI standard compliant tools (Dymola and Software Production Engineering). It is of particular interest how the trained NNs of the hybrid model are implemented. We present two approaches: (1) generation of C code using existing Open Neural Network Exchange (ONNX) tooling and (2) pure Modelica code with the tensor-flow represented as multi-dimensional equations. Both approaches are discussed, highlighting why (2) is, in the long run, a better option given the eFMI technology space.

Keywords: Hybrid modeling, neural network, machine learning, embedded system, Modelica, eFMI, Physics-enhanced Neural ODE, recalibration

1 Introduction

Two important objectives for system engineers that try to improve their simulation models are (1) minimizing the simulation-to-reality gap and (2) optimizing the model with respect to the simulation performance. In addition to traditional methods, machine learning (ML) methods offer varying data-driven approaches (Rai and Sahu 2020). Addressing the first objective in this context typically means using ML methods to learn (missing) dynamics from real-world measurements, thereby enhancing the predictive accuracy. To approach the second objective, surrogate models can be trained to replace computationally expensive components, using these very components to generate the necessary training data.

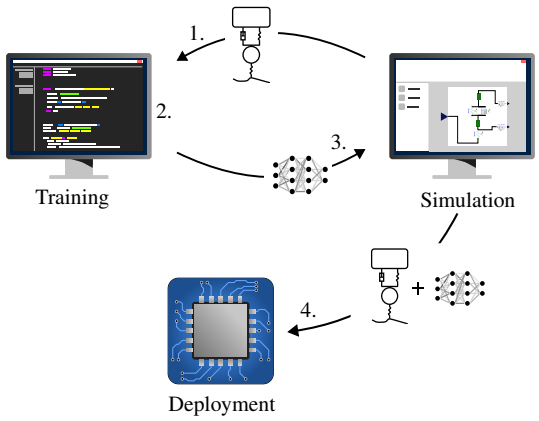


Figure 1. Development workflow of hybrid simulation models, including final deployment on the embedded target of a cyber-physical system.

It has been shown that dynamical systems can be thoroughly represented by recurrent neural networks (RNN) (Funahashi and Nakamura 1993), physics-informed neural networks (PINN) (Raissi, Perdikaris, and Karniadakis 2019; Cuomo et al. 2022), or neural ordinary differential equations (NODE) (Chen et al. 2018). Hybrid models (Willard et al. 2022) that directly combine trainable components with physics equations usually yield better stability, data efficiency and interpretability. In this study, we focus on a type of hybrid model known as physics-enhanced neural ordinary differential equation (PeN-ODE). PeN-ODEs combine ordinary differential equations (ODE) and artificial neural networks (NN) in a meaningful way (Kamp, Ultsch, and Brembeck 2023). The NNs are trained within the model-equations to capture missing non-linear effects or quantities to improve the predictive quality of the model. The implementation of PeN-ODEs is relatively straightforward, as it requires only the extension of the right-hand side of the ODE-system $\dot{x}(t) = f(x,u,t)$ with NNs. For the training and simulation of PeN-ODEs, well-established numerical integrator algorithms can be used.

Figure 1 sketches the general development workflow of hybrid models for system simulation, eventually deployed in a cyber-physical system (CPS). Assuming an equation-based model in a simulation environment, such as a Modelica[1] model in Dymola[2], is already available, the process is covered by Steps 1-4:

Step 1 (Export): The physics equations of the model are exported into an ML training environment (e.g., PyTorch[3] in Python[4] or Julia[5]).

Step 2 (Extension & training): The physics equations are extended with NNs to form a hybrid model (the PeN-ODE) that is typically trained using gradient-based optimization methods.

Step 3 (Re-import and simulation): The trained PeN-ODE or the NNs are re-imported into the simulation environment, enabling simulative validation and combination with other, non-hybrid model parts and/or control algorithms of the CPS.

Step 4 (Embedded application): The PeN-ODE is transformed into an implementation suited for deployment on an embedded target of the CPS.

In this study, we assume that the PeN-ODE is already trained – i.e., Steps 1-2 have been accomplished – for example using techniques presented by Thummerer et al. (2022). Thus, we present solutions for Steps 3-4, i.e., the incorporation of trained NNs in Modelica models with the ultimate goal to apply the whole PeN-ODE on an embedded system. The latter objective (Step 4) typically comprises compliance with complex, non-functional embedded domain requirements such as:

4a: MISRA:C 2023 compliance (The MISRA Consortium 2023).

4b: Restricted dependencies on libraries and frameworks.

4c: Worst execution-time and memory-consumption guarantees.

4d: Self-dependent implementation of the PeN-ODE, with its ODEs inline integrated to a level where only linear solver calls are required; such extensive inline integration is typically required to achieve (4b) and (4c), since non-linear solvers jeopardize (4b) and likely violate (4c).

4e: Strict error-handling concepts, especially in case of unexpected Positive or Negative Infinity and Not-a-Number (NaN) results of floating-point operations (IEEE 2019).

4f: Software-, processor- and hardware-in-the-loop (SiL, PiL & HiL) tests which are ideally derived from model-in-the-loop (MiL) tests defined in the simulation environment.

To comply with these requirements is of uttermost importance for CPS applications and especially relevant in control engineering, where Step 4 often is the ultimate objective for hybrid models. A typical use-case is to obtain a reduced order/surrogate model for unknown physics or to achieve acceptable performance in an embedded environment where computational resources are scarce.

An important question with respect to Step 4 is how the trained hybrid model – i.e., the PeN-ODE as the source for the embedded solution – looks like. The training and re-import of Steps 2-3 do not necessarily yield a model with the same abstraction level as the original equation-based model. For example, the approach of Thummerer et al. (2022) relies on the Functional Mock-up Interface[6] (FMI) for training and re-import. However, a Functional Mock-up Unit (FMU) is binary code or C source code. This means that the original physics equations are causalized and thus no longer explicitly available in the PeN-ODE. If only the NN parts of the PeN-ODE are re-imported as FMUs, manual integration with the original physics of the model is required. In any case, the tensor-flows of the NNs are hidden inside the FMU.

Thus, starting from an equation-based Modelica model, Steps 2-3 may yield a model with lower abstraction levels than the acausal equations. This holds also for approaches that use the Open Neural Network Exchange[7] (ONNX) format for trained NNs, or any low-level code implementation derived from that. Although re-importing such an implementation into an equation-based Modelica model for the sole purpose of simulation usually poses no issue, e.g., via FMI or the approach presented in Section 3, generating embedded code for the *whole* hybrid model is troublesome because the non-equation parts hamper compliance with requirements 4b-e. In particular, the ODE inline integration for the PeN-ODE (4d) is challenging, but also fulfilling requirement 4e is difficult without equation knowledge. If, on the other hand, Steps 2-3 yield a trained hybrid model whose physics and NNs are modeled as equations or ODEs, the simulation tool can leverage its existing numeric and symbolic manipulations and optimizations to find an algorithmic, causalized inline integration with only linear solver calls.

In the following, we investigate how the tensor-flow of trained PeN-ODEs can be preserved as equations in Step 3, such that Step 4 can leverage existing code generation facilities of the equation-based simulation tooling, which yields a toolchain from acausal physics equations with trained NNs down to safety-critical, hard real-time capable embedded code satisfying 4a-f. The key enabler for Step 4 is the Functional Mock-up Interface for em-

[1] https://modelica.org/language
[2] https://www.dymola.com
[3] https://pytorch.org
[4] https://www.python.org
[5] https://julialang.org
[6] https://fmi-standard.org
[7] https://onnx.ai

DOI 10.3384/ecp218555

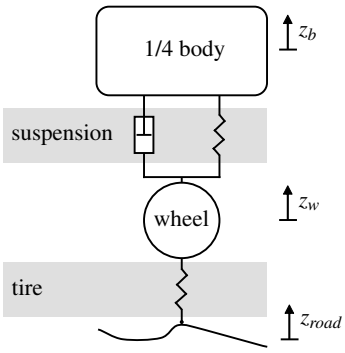


Figure 2. Scheme of the simple quarter vehicle model (QVM) comprising the body and wheel masses, a spring to represent the tire and a spring-damper pair for the suspension. The model input is the road height z_{road}; the vertical wheel and body positions z_w and z_b are the model outputs.

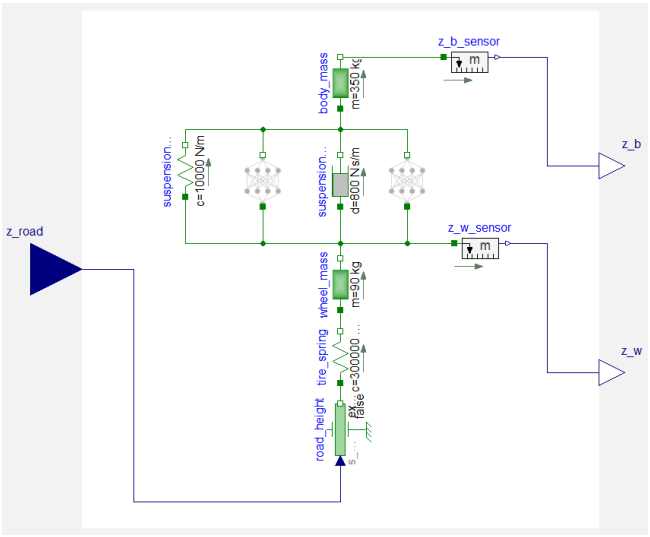


Figure 3. Modelica model of the neural QVM, i.e., the PeN-ODE. The two trained NNs capture non-linear effects of the suspension's spring and damper and are connected in parallel to the linear physics components. The NNs can be imported using generated C code from an ONNX representation (cf. Section 3) or as native Modelica equations obtained through a custom Python to Modelica generator (Section 4).

bedded systems[8] (eFMI). The final toolchain relies on the Dymola Modelica tool for equation-based physics modeling and simulation (Step 1), PyTorch for PeN-ODE training (Step 2), and the eFMI support of Dymola and Software Production Engineering[9] for embedded code generation (Step 4). In order to re-import the tensor-flows of trained PeN-ODEs as equations in Dymola (Step 3), we developed a simple Modelica code generator leveraging the open source NeuralNetwork[10] Modelica library.

The rest of the paper is organized as follows: Section 2 presents a motivating case-study which requires the deployment of a trained PeN-ODE on an embedded target and recalibration of NN parameters during runtime. Section 3 presents the C code generation from ONNX models via onnx2c[11] as a first approach for Steps 3-4. Sections 4 and 5 investigate our final solution for Steps 3-4, i.e., a Python to Modelica generator and eFMI with Dymola and Software Production Engineering. Section 6 presents future work, most importantly avoidance of scalarization of tensor-flows in embedded code, and Section 7 finally summarizes the related work regarding hybrid models and embedded code generation in Modelica.

2 Quarter vehicle model case-study

We showcase the suitability of the proposed toolchain for embedded code generation of PeN-ODEs (Step 4, a-f) by conducting a case-study for a quarter vehicle model (QVM) that represents the vertical driving dynamics of a road vehicle. QVMs are commonly used in the domain of vehicle dynamics to represent the dynamics of the suspension for controller synthesis and as prediction models (Fleps-Dezasse and Brembeck 2013; Ultsch, Ruggaber, et al. 2021). Typical applications of QVMs are fault detection of the suspension and the wheels or serving as part of a virtual sensor or as prediction model of controlled, semi-active suspensions that improve driving comfort or road-holding properties (Ultsch, Pfeiffer, et al. 2024). A QVM is limited to the one-dimensional (vertical) dynamics of one wheel and one quarter of the car-body, as depicted in Figure 2. Capturing the vertical dynamics of a road vehicle correctly is challenging, which is due to the influence of non-linear effects such as friction and elasticities (i.e., rubber bushings). To capture such effects, we use a PeN-ODE approach and integrate NNs into the physics equations to learn unknown non-linearities from measurement data, cf. (Kamp, Ultsch, and Brembeck 2023).

The starting point of the PeN-ODE development is a plain physics-based Modelica model which implements only the linear suspension dynamics. This linear QVM is exported as an FMU and imported into the training environment PyTorch. After extending the differential equations with two NNs – one amending the linear equations of the damper with its non-linear behavior and likewise another for the spring – a gradient-based optimization is conducted. After the training, the trained NNs have to be integrated into the (still linear) Modelica model (Step 3) to obtain a Modelica implementation of the PeN-ODE (cf. Figure 3). In Sections 3 and 4 we present two approaches to accomplish the transfer from the training environment back to the simulation environment. The neural QVM (nQVM) is then used for simulative validation, e.g., using Dymola, and shall further be deployed on an real-time target in the vehicle. There it serves as a prediction model

[8] https://www.efmi-standard.org

[9] https://my.3dexperience.3ds.com/welcome/compass-world/3dexperience-industries/transportation-and-mobility/smart-safe-and-connected/embedded-software-engineering/systems-software-production-engineer

[10] https://github.com/AMIT-HSBI/NeuralNetwork

[11] https://github.com/kraiskil/onnx2c

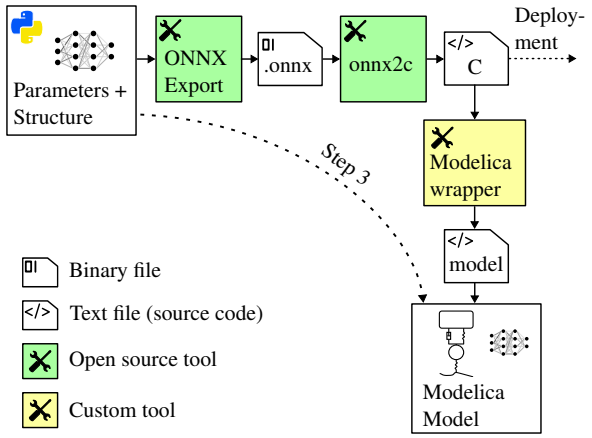


Figure 4. Scheme of the export toolchain using onnx2c to generate C code for ONNX models. Since many ML-frameworks already support ONNX export, we mark the ONNX export as openly available. Although the code onnx2c generates is suited for embedded application, there is no "ready-to-use" solution for embedded code generation of the *whole* PeN-ODE including its physics equations. The generated C code is imported into Modelica by using a Modelica external C function wrapped into a MSL MIMO block. Listings 1-2 show the wrapper code as generated by our custom tool.

for a control algorithm of the semi-active suspension. In addition, we want to enable the recalibration of the trained NNs during runtime on the embedded system, i.e., without recompilation. This is especially useful when multiple variants of the PeN-ODE are trained to consider changing circumstances, e.g., the current road type or suspension adjustments, as it is common practice in scheduled control algorithms. The application on the embedded target leverages the eFMI standard with tunable NN parameters, which is described in Section 5.

This work captures intermediate results of ongoing work, thus the scope of our case-study is up to the SiL validation of the embedded solution, including the recalibration. The SiL tests allow the comparison with the (continuous) simulation results and constitute the foundation for future HiL and driving tests under real world conditions.

3 NNs as external C code

In this section, we present a method to leverage ONNX as a model exchange standard and an open source ONNX to C compiler to incorporate trained NNs in Modelica models. The presented workflow can be automated to a high degree which we underline with generated code samples. However, the dependency on external C code hinders the application of the obtained PeN-ODE on embedded systems through eFMI, since integration with the embedded code generation for the physics part of the PeN-ODE is not obvious. Nevertheless, the method yields plain C code for the NNs that is generally suited for embedded application and is therefore relevant for cases without additional Modelica physics. We will outline how this approach can be used to deploy trained NNs in a CPS context, even if it is not the favorable approach to our use-case.

3.1 The ONNX format

"ONNX is an open format built to represent machine learning models. ONNX defines a common set of operators – the building blocks of machine learning and deep learning models ..."[7]. Once the model is defined using these operators, ONNX uses the (binary) Protocol Buffers format for serialization. Such ONNX files serve as exchange containers and can be compared to FMUs. The ONNX format is already widely established as an exchange standard for ML models and is supported e.g., by the Keras[12] and PyTorch frameworks, MATLAB®[13] and many more. The ONNX Runtime enables direct inference using an ONNX model.

3.2 ONNX to C compiler

In some applications, relying on the ONNX Runtime is not a valid option, especially when the model should be deployed on an embedded system. The open source onnx2c[11] compiler offers a remedy for users that can make use of plain C code. It is optimized for TinyML[14], i.e., running on microcontrollers. Provided with an ONNX model, onnx2c generates a single C file with all required functions that correspond to the atomic ONNX operators. The parameter arrays that hold the trainable parameters are likewise contained in the generated code. It has to be mentioned that onnx2c does not support all available operators and not all model typologies and is a privately maintained project. The C code contains a function `entry(const float x[1][M], float y[1][N])` with the input dimension M and output dimension N. This function is the only function that needs to be called in order to perform a forward pass through the model.

3.3 Integration in Modelica

Once the NN is exported as an ONNX model and compiled into C code, it can be used in Modelica using its external C interface (cf. Figure 4). Listing 1 presents an example implementation of the entry-function call that can be used as a template for generation. Further, it can be convenient to wrap the function into a Modelica Standard Library (MSL) MIMO block as shown in Listing 2.

Listing 1. Modelica function that calls the entry function from the generated C file (called nn.c). The dimensions M and N correspond with the number of in- and outputs of the NN (to be replaced with size_t values).

```
function call_entry "call_entry"
  input Real u[M];
  output Real y[N];
  external "C" call_nn(u, y)
    annotation(
    IncludeDirectory="<<nn.c dir>>",
```

[12] https://keras.io
[13] https://www.mathworks.com/products/matlab.html
[14] https://github.com/mit-han-lab/tinyml

DOI
10.3384/ecp218555

```
    Include="
#include \"nn.c\"
void call_nn(const double* u, double* y)
{
  float u_buf[1][M], y_buf[1][N];
  size_t i;

  for (i = 0; i < M; i++)
  {
    u_buf[0][i] = (float) u[i];
  }

  entry(u_buf, y_buf);

  for (i = 0; i < N; i++)
  {
    y[i] = (double) y_buf[0][i];
  }
}");
end call_entry;
```

Listing 2. MIMO wrapper for C-function call (cf. Listing 1); The dimensions M and N correspond with the number of in- and outputs of the NN (to be replaced with Integer values).

```
model MyNN
  extends Modelica.Blocks.Interfaces.MIMO(
    nin = M,
    nout = N);
equation
  y = call_entry(u);
end MyNN;
```

If a PeN-ODE contains multiple NNs, one must assure that variable- and function names in the C code are unique. When using onnx2c, this is generally not the case, since the C files are generated independently and at least the entry-function always has the same name. As of now, this can only be solved by editing the C code after its generation. Issues can also occur, when the ONNX model is composed in a way that the operator-nodes have ambiguous or repeating names. This can be avoided by giving a unique name to each node during the ONNX export of the ML model. The node names of an existing ONNX model can still be adapted, e.g., using the ONNX Python API.

3.4 Discussion

The presented approach uses the ONNX open standard and an open source ONNX to C compiler. The incorporation of the generated C code in Modelica (Step 3) is straightforward and can be effectively automated by generating corresponding Modelica wrappers. Furthermore, ONNX is a well established exchange standard for ML models and many ML frameworks already offer the export to ONNX. The onnx2c compiler is stable, maintained, and supports many of the existing ONNX operators. It generates C code which is suitable for the application on microcontrollers, which is important for CPSs. In comparison to solutions that use the ONNX Runtime, it has the advantage to be self-contained without any third party library or framework dependencies, which facilitates exchange and portability. Modelica tools can, for example, export PeN-ODE models as binary code FMUs without further dependencies.

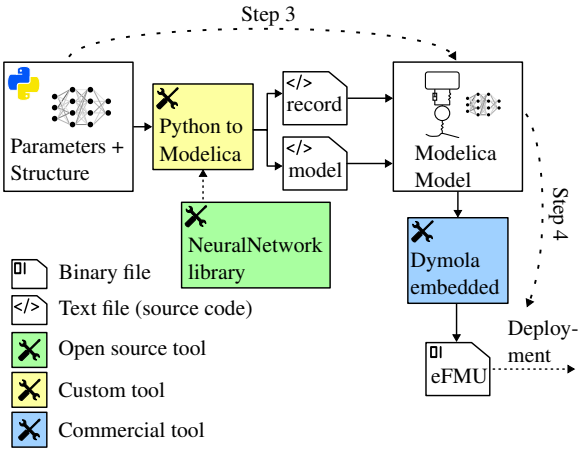


Figure 5. Scheme of the export toolchain employing the NeuralNetwork library. The Python to Modelica generator is a custom tool that allows direct Modelica code generation from Python. This approach leverages Dymola's existing symbolic facilities to find algorithmic solutions suited for embedded application and enable embedded code generation for the *whole* PeN-ODE using the open eFMI standard.

However, the onnx2c approach yields embedded code only for the NN parts of the PeN-ODE. Embedded code generation for the whole PeN-ODE (Step 4) is not straightforward because the production code for the physics equations must be generated and connected with the NNs. This is not a trivial system integration task, since knowledge about causalization, equation system properties and connectivity of the *whole* PeN-ODE must be properly incorporated. There might be, for example, algebraic loops between physics and NNs, or the physics equations and NNs form a mixed system of equations where Boolean conditions depending on NN outputs switch physics behavior fed into the NN. In such cases, the NN code has to be properly embedded into the iterations of the inline integrated ODE solver when generating embedded code for the physics parts of the Modelica model.

These embedded code generation issues for PeN-ODEs can be bypassed by handing the challenges over to the already existing, advanced symbolic facilities of Modelica compilers like Dymola. Section 4 presents an approach for Step 3 where the NN parts of the PeN-ODE are re-imported as pure Modelica equations such that the Modelica compiler can take care of proper integration.

4 NNs as native Modelica

In this section, we present an approach that implements trained NNs as Modelica models utilizing the NeuralNetwork Modelica library. This method allows seamless integration of the NNs with the physics equations and enables eFMI-based embedded code generation for the whole PeN-ODE. Figure 5 summarizes the tooling of this Modelica-centric workflow, where the trained NNs are

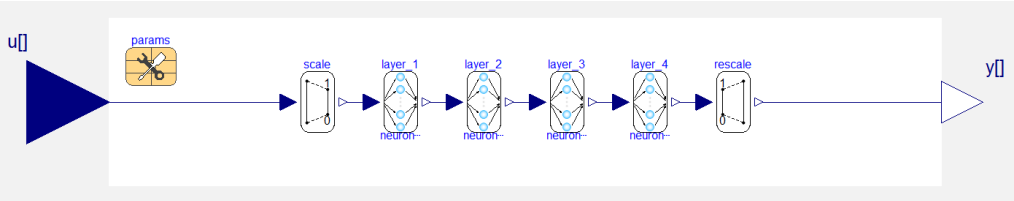


Figure 6. Generated, equation-based Modelica model of one of the trained NNs of the nQVM, showing its diagrammatic tensorflow using NeuralNetwork library components. The NN parameters (weights and biases) for each of the four dense layers (16×1 – 16×16 – 16×16 – 1×16) are contained in the parameter record that can be marked for exposure as tunable eFMI parameters.

transformed back to Modelica equations.

4.1 The NeuralNetwork library

The NeuralNetwork Modelica library was first published in 2006 (Codecà and Casella 2006). Since 2023, it is further developed by the University of Applied Sciences and Arts Bielefeld (HSBI) under open source 3-Clause BSD Licence[10]. As of version 2.1, it offers implementations to compose multi-layer perceptrons (MLP) including dense layers, scaling, standardizing, and principal component analysis (PCA) layers, and the activation functions rectified linear unit (ReLU), sigmoid, hyperbolic tangent (tanh), softplus and unit step.

Considering the vast variety of ML architectures, this is only a basic set. For PeN-ODEs however, simple MLPs suffice in many cases (Thummerer, Stoljar, and Mikelsons 2022; Kamp, Ultsch, and Brembeck 2023).

4.2 Generation of Modelica NNs

We implemented a custom Python to Modelica code generator to automate the implementation of trained NNs using the definitions of the NeuralNetwork library. Listing 3 presents an example of a simple MLP that was generated in this manner. We deem it best to generate a separate Modelica record alongside the NN that holds its parameters, i.e., NN weights and biases (cf. Listing 4). This record can be used to redeclare the parameters of the NN block and thus enables a fast exchange. This is useful when multiple variants of the NN are to be validated in simulation, given that the NN architecture (number of layers, number of neurons, and activation functions) is maintained. Figure 6 shows the composition of different layers of a typical MLP with the corresponding parameter record.

Listing 3. Sample of a generated Modelica NN. The NeuralNetwork library components are wrapped into a MIMO block comparable to the external C approach (cf. Listing 2).

```
block MyNN
  extends Modelica.Blocks.Interfaces.MIMO;
  import NN = NeuralNetwork;
  // NN parameter record
  parameter MyParametrization params;
  // NN layers
  NN.Layer.Dense layer_1(
    weights = params.layer_1_weights,
    bias = params.layer_1_bias,
    redeclare function f =
      NN.ActivationFunctions.ReLu);
  NN.Layer.Dense layer_2(
    weights = params.layer_2_weights,
    bias = params.layer_2_bias,
    redeclare function f =
      NN.ActivationFunctions.Id);

equation
  // NN composition
  connect(u, layer_1.u);
  connect(layer_1.y, layer_2.u);
  connect(layer_2.y, y);
end MyNN;
```

Listing 4. Generated parameter record defining default weights and biases for the NN of Listing 3, exposed as tunable eFMI parameters for the embedded code generation of Section 5.

```
record MyParametrization
  extends Modelica.Icons.Record;
  // Trained NN parameters
  parameter Real [16, 1] layer_1_weights =
    {{2.15}, {-0.713}, ...};
  parameter Real [16] layer_1_bias =
    {0.174, -0.0312, ...};
  parameter Real [1, 16] layer_2_weights =
    {{0.215, 1.342, ...}};
  parameter Real [1] layer_2_bias =
    {0.230};
  // Expose as tunable eFMI parameters
  annotation(
    __Dymola_eFMI_ExposeTunableParameters =
      true);
end MyParametrization;
```

4.3 Discussion

The pure Modelica approach offers a seamless integration of trained NNs with the physics part of the PeN-ODE in Modelica. The NN parameters can be encapsulated in records separated from the NN architecture, facilitating variation and training validation. A drawback compared to the usage of ONNX via onnx2c is the limited number of available architectures of the NeuralNetwork library. Further, a complete Python to Modelica generator leveraging the NeuralNetwork library is not (yet) freely available. Our implementation is, as of now, just a functional prototype and not published. The most important advantage of the native Modelica approach is the possibility to export the *whole* PeN-ODE via eFMI for embedded application as motivated and described in the following Section 5.

5 Embedded code via eFMI

In this section, we motivate the use of the eFMI standard for Modelica equation-based hybrid models in embedded applications. Our solution relies on the import of trained NNs as Modelica equations as presented in Section 4, meets requirements 4a-f of Section 1, and enables online recalibration of NN parameters.

5.1 The eFMI Standard

The Functional Mock-up Interface for embedded systems (eFMI) is "an open standard for the ... model-transformation-based development of advanced control functions suited for safety-critical and real time targets ... [, defining a] container architecture ... from high-level modeling and simulation – e.g., a-causal, equation-based physics in Modelica – down to actual embedded code"[8].

The key interface between the physics simulation and embedded application domains is the Guarded Algorithmic Language for Embedded Control (GALEC). To deploy an equation-based simulation model in a safety-critical, hard real-time environment, the simulation tool has to transform the model into a GALEC program (Algorithm Code container) and hand it over to any eFMI-aware production code generator for final embedded code generation (Production and Binary Code container). The idea is, that each tooling can be a domain expert on its abstraction-level without bothering about the other domains. A Modelica tool like Dymola knows a lot about discretization, symbolic optimization and inline integration – i.e., how to find a sampled algorithmic solution expressible in GALEC – whereas production code generators like Software Production Engineering are knowledgeable about satisfying embedded domain coding requirements but have no concept of modeling physics for equation-based simulation.

GALEC has some language characteristics making it a convenient intermediate representation between modeling and embedded software (Lenord et al. 2021). It is target independent, supports multi-dimensional arithmetic, and a rich set of built-in functions to abstract common math operations including interpolation and solving systems of linear equations. Most importantly, it is computational-safe, which means that a powerful static evaluation concept is used to guarantee all indexing is within bounds, all loops have an upper-bound, and all potential runtime errors like NaNs, singular linear equation systems or domain errors of built-in functions (like poles of trigonometric functions) are handled or explicitly exposed to the runtime environment. GALEC programs satisfy 4b-e of Step 4 by definition. Once a simulation tool finds a sampled solution for the simulation problem – which is not always possible for general equation systems – and expresses it as GALEC program, further tooling towards embedded C code "only" has to preserve these characteristics and can focus on code optimization (e.g., loop and multi-dimensional expression unrolling) and embedded target environment compliance (e.g., compliance with style guidelines, code analysers, interfaces or system integration requirements).

Besides Algorithm, Production and Binary Code containers, eFMI also defines Behavioral Model containers to define test-scenarios. Typically, Behavioral Models are derived from continuous MiL experiments in a physics simulation environment and shared for later SiL and HiL tests of production code.

5.2 eFMI support in Dymola

Dassault Systèmes provides several tools supporting eFMI: Dymola for generating eFMI Algorithm Code containers for synchronous Modelica models, Software Production Engineering for generating eFMI Production, and in turn, Binary Code containers and AUTOSAR Builder[15] to extend Production Code containers to become AUTOSAR[16] components.

It is noteworthy, that Software Production Engineering is also used as embedded code generation backend for No Magic Cameo Systems Modeler[17], a development environment for model-driven engineering, system architecture, system design, and software engineering based on SysML[18]. Generated production code therefore is of industry-level quality, for example MISRA C:2023 compliant. The integration with Dymola is seamless. Dymola provides facilities to configure 32 bit and 64 bit floating-point precision for production code generation, import production code for SiL simulation via generated Modelica wrappers, derive SiL tests from existing MiL experiments, check MISRA C:2023 compliance with Cppcheck Premium[19], import production code in MATLAB®/Simulink®[20] as C Function blocks, and export eFMUs with deployment-ready production code as FMUs.

5.3 QVM case-study with Dymola

Based on Dymola's eFMI support presented in the previous section, generating embedded code for the nQVM of Section 2, with its damper and spring NNs represented as NeuronalNetwork library models according to Section 4, has been straightforward. The two major challenges were, first, how to conveniently expose the NN weights and biases as tunable eFMI parameters, and second, how to model the respective recalibration SiL tests in Modelica.

For the first issue, we extended Dymola with a new annotation for Modelica records. Adding `__Dymola_eFMI_ExposeTunableParameters = true` to a record *class* marks the independent parameters of *instances* for exposure as tunable eFMI parameters, regardless how deeply nested within models they are. The Python to Modelica generator of Section 4 can hence automatically add the annotation to the NN parametrization record (cf. Listing 4).

[15] https://www.3ds.com/products/catia/autosar-builder
[16] https://www.autosar.org
[17] https://www.3ds.com/products/catia/no-magic
[18] https://www.omgsysml.org
[19] https://www.cppcheck.com
[20] https://www.mathworks.com/products/simulink.html

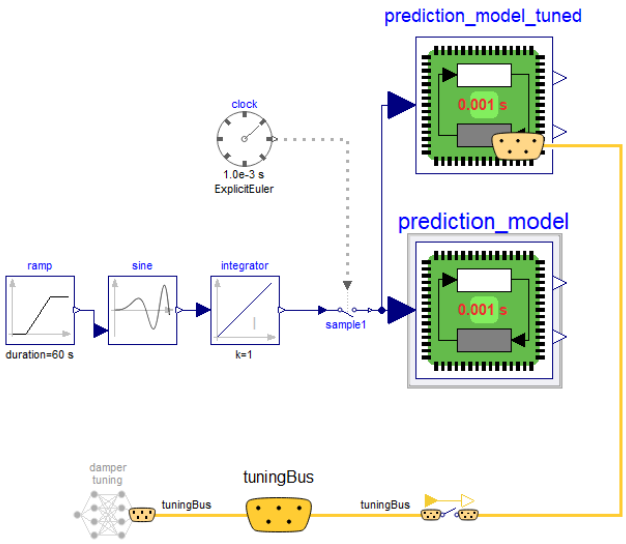


Figure 7. Recalibration SiL test in Dymola: The nQVM PeN-ODE is instantiated twice – once untuned and once tuned – via a generated Modelica wrapper backed by the eFMUs production code. The damper tuning component is manually implemented. It triggers recalibration at 2 s and applies updated NN parameters for the learned friction model on the generated tuning bus.

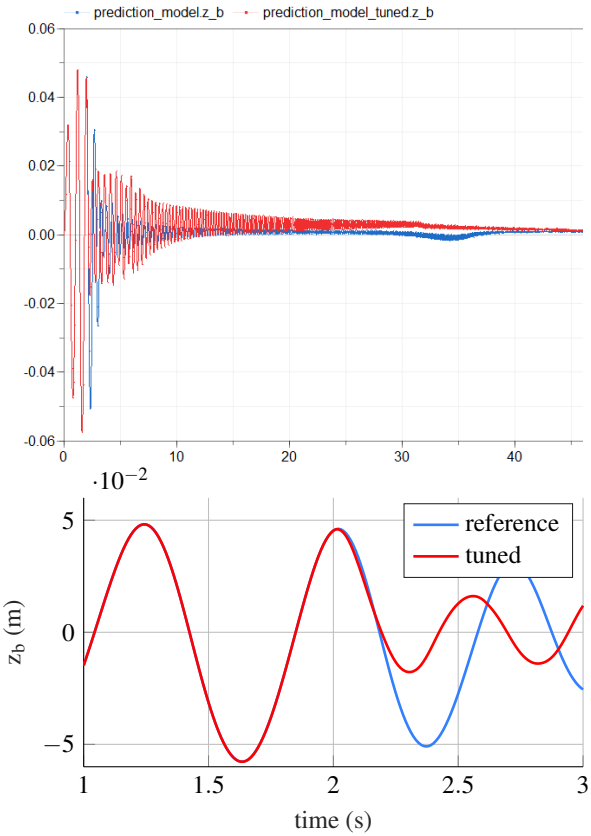


Figure 8. SiL simulation results of the body height z_b for the untuned (blue) and tuned (red) nQVM. At 2 s simulation time, the recalibration is triggered and leads to a new system behavior.

The second issue, how to model recalibration SiL tests, exposes a general Modelica shortcoming. From the eFMI perspective, recalibration test scenarios are well-supported in Behavioral Model containers; and Dymola supports the derivation of Behavioral Models from existing Modelica experiments by searching them for instances of the model that is subject to eFMI code generation. Each of those instances becomes a SiL test scenario, where the instance inputs are the stimuli and the outputs the reference trajectories. This concept works well to automatically derive SiL test suites from ordinary Modelica experiments. However, Modelica has no online recalibration concept. In Modelica, parameters can be only modified before, but not during the simulation. Hence, online recalibration must be modeled by runtime values encoding "parameters". Since this implies extensive model changes on the models to recalibrate, recalibration support for ordinary Modelica models is pointless. In our case however, we can provide recalibration facilities in the Modelica wrappers Dymola generates for SiL simulation of production code generated by Software Production Engineering. To this end, the wrappers provide an optional tuning bus, which can be enabled whenever recalibration shall be tested. Actual calibrations are simply connected to the bus, requiring manual modeling of a continuous time-based source provisioning the parameter sets to apply at each time point.

Figures 7 shows the recalibration scenario we used to validate the nQVM. In this experiment, we recalibrate the damper-NN with a slightly altered (learned) friction model after 2 s. The PeN-ODE is simulated with a chirp signal (sine-sweep) as input. In Figure 8, the simulation result of the body height z_b of the tuned and untuned PeN-ODE is plotted. Although the tuning clearly alters the dynamics of the system, the state of the PeN-ODE is not changed when recalibrating such that there are no discontinuities in the trajectory. If recalibration would take several sampling steps, which can be expected when using actual real-time hardware, the current state would be held – i.e., no samplings are conducted – until the recalibraion is terminated. Nevertheless, it is likely that observers or control algorithms using the PeN-ODE converge much faster when continuing sampling from the hold state compared to a full state reset. Although the nQVM is stable in the recalibration scenario of Figures 7 and 8, no such guarantee can be given for general equation systems; if, and under which constraints, an independent parameter indeed is suited for online tuning, is a domain specific characteristic.

6 Future work

This paper presents intermediate results of the ongoing research in the PeN-ODE-related work package (WP4) of the ITEA 4 Project OpenSCALING[21] (cf. Acknowledgements). The work scheduled for the second half of the project ending in October 2026 can be separated into three tasks, ranked from highest to lowest priority: (1) preserva-

[21] https://itea4.org/project/openscaling.html

DOI 10.3384/ecp218555

tion of the multi-dimensional arithmetic of the tensor-flow of NNs in production code, (2) actual system integration of the eFMU of the case-study presented in Section 2 with HiL and driving tests under real-world conditions including recalibration of NN parameters and (3) support of the System Structure & Parametrization[22] (SSP) standard for NN parametrization and recalibration.

6.1 Multi-dimensional embedded tensor-flows

Most Modelica tools, including Dymola, transform models into an index 1 differential algebraic equation (DAE) system, where each element of multi-dimensions is scalarized to individual variables; doing so, multi-dimensional arithmetic operations are flattened to a set of individual scalar operations (scalarization). Although this representation is essential for many advanced numeric and symbolic manipulations and optimizations, it also significantly increases code size when scalarizing the tensor-flows of the equation-based NNs presented in Section 4. For embedded applications, code size is critical. Hence, it is important to avoid the scalarization of tensor-flows. However, for the physics-parts of PeN-ODEs it still is beneficial to find symbolic solutions to avoid linear system solver calls.

In our case-study, the actual physics equations of the nQVM only result in ~4 KB of eFMI GALEC code including two symbolically solved size-1 linear equation systems, whereas the scalarized tensor-flows of the spring and damper NNs – each holding 593 parameters in the input layer (16×1, ReLu), two hidden layers (16×16, ReLu), and the output layer (1×16, Identity), each with bias (cf. Figure 6) – produce ~117 KB of code. This is a lot, considering that the tensor-flows of the NNs could be written as single-line multi-dimensional GALEC expressions. The ratios for derived C production and binary code are similar. The onnx2c solution of Section 3, which preserves the tensor-flows as for-loops, only yields ~5.5 KB of C code. All numbers are excluding the initialization of NN weights and biases, which contributes significantly to code size but cannot be avoided.

The prevention of scalarization is subject of work package 3 (WP3), large scale system (LSS) modeling, of the OpenSCALING project. Although the development focuses on physics based LSSs – like electric power grids with homogeneous components of huge quantities modeled as arrays of Modelica components – the planned solutions are likely also applicable in our use-case. If anything, tensor-flows are even simpler to handle because they are clean – i.e., not mixed with exceptions or zero crossing logic – arithmetic. Foundational research on how to preserve multi-dimensional operations for index 1 DAEs without sacrificing symbolic optimizations is already available (Otter and Elmqvist 2017; Abdelhak, Casella, and Bachmann 2023).

[22] https://ssp-standard.org

6.2 eFMU system integration and CPS tests

The presented toolchain has been validated up to, and including, SiL tests of the generated eFMU in Dymola. The actual system integration on a dedicated embedded target – for our case-study this involves the deployment of the nQVM as prediction model on the embedded control system of our experimental research platform (Ruggaber et al. 2023) – still must be conducted. We are confident, that our eFMI approach is well-suited for that eFMUs generated by Dymola and Software Production Engineering have been successfully system-integrated in the past, including final CPS tests under real-world conditions (Ultsch, Ruggaber, et al. 2021). A new challenge is the online recalibration for tuning the weights of the embedded NNs, which we deem non-critical, considering the successful SiL experiment of the recalibration facilities of the generated eFMU.

6.3 SSP standard support

Online recalibration of PeN-ODE NN-weights is related to ongoing SSP standardization efforts. SSP is supposed to be a universal standard for defining varying parameter sets for interconnected FMUs (Hällqvist et al. 2021). Modelica support, in particular SSP import and export in Dymola, is available (Brück 2023). Our tunable parameter record approach of Sections 4 and 5 for modeling NN-weights can be mapped directly using Dymola's existing SSP facilities. In FMI and eFMI however, recalibration of tunable parameters can be seen as a *timed sequence* of parameter changes; however, SSP has no notion of timing. Likewise, Modelica lacks the means to conveniently model online recalibration. In Modelica, parameters can only be changed by means of static modifications before simulation starts, which is why the SiL tests presented in Section 5 had to implement new calibrations via ordinary Modelica variables instead of parameters. This is troublesome and requires the definition of event-points for recalibration throughout the continuous simulation. To align SSP, Modelica, FMI and eFMI such that recalibration test-scenarios can be modeled as timed SSP parametrizations is an open issue, very much in the scope of OpenSCALING and its objective to harmonize the Modelica Association's[23] open standards ecosystem.

7 Related work

Our previous work provided an extended introduction to PeN-ODEs, including how to define and train them (Kamp, Ultsch, and Brembeck 2023). Requirements 4a-f, to enable embedded and CPS applications, were not considered however.

Thummerer et al. (2022) proposed a solution for export, training, and re-import of PeN-ODEs in simulation environments (Steps 1-3) using the FMI standard. Their approach is feasible even for complex systems, and especially in industrial applications where discretion is of importance. Thanks to the ubiquitous availability of

[23] https://modelica.org/association

FMI in the simulation domain, and increasing support also in training environments (e.g., FMPy[24] for Python or FMI.jl[25] for Julia), this approach is widely applicable. However, FMUs are implementation black-boxes, hiding the involved equations such that Step 4, preparing PeN-ODEs for embedded application, is not straightforward. The approach does not consider the embedded domain requirements 4a-f.

The open source SMArtInt library[26] offers a way to use ONNX models and TensorFlow[27] models exported as Lite Runtime[28] (LiteRT) in Modelica, supporting a huge range of NN types and architectures. However, its dependency on a plethora of heterogeneous third party frameworks and technologies severely impedes embedded application when compliance with 4b-e is required.

Hübel et al. (2022) trained a surrogate model that is benchmarked inside a Modelica model. Comparable to our approach, they used the NeuralNetwork Modelica library to transfer the trained NN to Modelica. They also mentioned the possibility to use generated C code as a future research topic, but did not elaborate on embedded applications.

In the ITEA EMPHYSIS project (EMPHYSIS International Consortium 2021), which bore the eFMI standard, multiple demonstrators used eFMUs to incorporate NNs into predictive model control applications targeting embedded systems. The starting point of embedded code generation were trained NNs however, not hybrid models. Hence, although 4a-c and 4e-f are supported by representing NNs as eFMI GALEC programs and leveraging the eFMI tooling as we do, the combination with physics equations – and therefore an ODE inline integration to a level where only linear solver calls are required (4d) – has not been investigated.

The work of Ultsch et al. (2021) within the EMPHYSIS project can be regarded as predecessor to our study. Similar to the QVM case-study of Section 2, they transformed a nonlinear prediction model in Modelica via eFMI into an embedded solution suited for the model-based control of the semi-active dampers of a car. An extended Kalman filter was used for the state estimation based on the prediction model. To this end, the eFMU of the prediction model was integrated into a dedicated Kalman filter library implemented in C. The state estimation algorithm of the Kalman filter thereby perturbates discretized continuous states of the prediction model by explicitly setting them before performing an integration step with the inline integrated solver. The eFMU was deployed on the ECU of a series-produced car and validated in driving tests under real-world conditions.

Kurzbach et al. (2023) proposed a low-level equation language for Modelica, in which the higher level object-oriented concepts like inheritance, modifications and nested components are flattened to simple equations. The motivation for such a "base Modelica" is to facilitate the integration of third party technology spaces with Modelica tooling. If such a representation becomes part of the Modelica standard and would be widely adopted, it could significantly ease bridging the simulation and ML domains (Step 1 and 3 of our approach); in particular, if it could be used as an actual *model* exchange format in FMI. Although the definition of a basic-equation language for Modelica has been a long time vision of the Modelica Association[23], it is much too early to judge if the proposal will be standardized and sufficiently widely adopted.

8 Conclusions

We presented a toolchain for the embedded application of PeN-ODE hybrid models derived from Modelica physics equations. In order to meet the non-functional requirements on embedded software, we relied on the Functional Mock-up Interface for embedded systems (eFMI) and its support in the Modelica tool Dymola. Dymola's existing symbolic optimization and inline-integration routines are prolific in finding causalized, algorithmic solutions for complex equation systems, in particular to avoid nonlinear solver calls as required in embedded applications. The export of algorithmic solutions as eFMI GALEC programs in turn enables further eFMI support to eventually derive high-quality production code.

A challenge of this approach is the correct integration of the trained NN-components of PeN-ODEs with the physics part, such that the integrated solvers of embedded solutions can properly handle the interactions between NNs and physics-equations, like event-handling and zero-crossings in mixed systems of equations, smoothness requirements, sampling and discretization, etc. We avoid this open research question by representing the tensor flow as such – i.e., as *multi-dimensional expression* – eventually yielding an equations-only representation of the whole PeN-ODE which can be processed by Dymola's existing symbolic facilities. For the required re-import of the trained NNs into the original physics-equations, we developed a Python to Modelica code generator targeting the NeuralNetwork Modelica library.

We validated our approach with the help of a neural quarter vehicle model (nQVM) case-study. SiL tests of the eFMI solution demonstrate its applicability, including recalibration tests for online changes of NN parameters. Left as future work is to extend Dymola's symbolic facilities such that they avoid scalarization of the tensor-flows of NNs and final system integration of the nQVM with HiL and driving tests under real world conditions.

Acknowledgements

This work has been supported by the Swedish Agency for Innovation Systems (Vinnova[29], grant number 2023-

[24] https://github.com/CATIA-Systems/FMPy
[25] https://github.com/ThummeTo/FMI.jl
[26] https://github.com/xrg-simulation/SMArtIInt
[27] https://www.tensorflow.org
[28] Formerly known as TensorFlow Lite (TFLite).
[29] https://www.vinnova.se

00969) and the German Federal Ministry of Education and Research (BMBF[30], grant number FKZ 01IS23062A) within the ITEA 4 Project Open standards for SCALable virtual engineerING and operation (OpenSCALING[21], ITEA Project 22013).

References

Abdelhak, Karim, Francesco Casella, and Bernhard Bachmann (2023-10). "Pseudo Array Causalization". In: *Proceedings of the 15th International Modelica Conference*. Vol. 204. Linköping Electronic Conference Proceedings. Modelica Association and Linköping University Electronic Press, pp. 177–188. DOI: 10.3384/ecp204177.

Brück, Dag (2023-10). "SSP in a Modelica Environment". In: *Proceedings of the 15th International Modelica Conference*. Vol. 204. Linköping Electronic Conference Proceedings. Modelica Association and Linköping University Electronic Press, pp. 711–715. DOI: 10.3384/ecp204711.

Chen, Ricky T. Q. et al. (2018-12). "Neural Ordinary Differential Equations". In: *Proceedings of The Thirty-Second Annual Conference on Advances in Neural Information Processing Systems*. Curran Associates, Inc., pp. 6571–6583. DOI: 10.48550/arXiv.1806.07366.

Codecà, Fabio and Francesco Casella (2006-09). "Neural Network Library in Modelica". In: *Proceedings of the 5th International Modelica Conference – Volume 2*. Modelica Association, pp. 549–557.

Cuomo, Salvatore et al. (2022-07). "Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next". In: *Journal of Scientific Computing* 92.3. DOI: 10.1007/s10915-022-01939-z.

EMPHYSIS International Consortium (2021-09). *EMPHYSIS – D7.9 eFMI for physics-based ECU controllers – Public report*. Tech. rep. ITEA 3 project 15016. ITEA. URL: https://www.efmi-standard.org/media/resources/emphysis-public-demonstrator-summary.pdf.

Fleps-Dezasse, Michael and Jonathan Brembeck (2013). "Model based vertical dynamics estimation with Modelica and FMI". In: *IFAC Proceedings Volumes* 46.21, pp. 341–346. DOI: 10.3182/20130904-4-jp-2042.00086.

Funahashi, Ken-ichi and Yuichi Nakamura (1993-01). "Approximation of dynamical systems by continuous time recurrent neural networks". In: *Neural Networks* 6.6, pp. 801–806. DOI: 10.1016/s0893-6080(05)80125-x.

Hällqvist, Robert et al. (2021-09). "Engineering Domain Interoperability Using the System Structure and Parameterization (SSP) Standard". In: *Proceedings of 14th Modelica Conference 2021*. Vol. 181. Linköping Electronic Conference Proceedings. Modelica Association and Linköping University Electronic Press, pp. 37–48. DOI: 10.3384/ecp2118137.

Hübel, Moritz et al. (2022-11). "Hybrid physical-AI based system modeling and simulation approach demonstrated on an automotive fuel cell". In: *Proceedings of Asian Modelica Conference 2022*. Vol. 193. Linköping Electronic Conference Proceedings. Modelica Association and Linköping University Electronic Press, pp. 157–163. DOI: 10.3384/ecp193157.

IEEE (2019-07). *IEEE Standard for Floating-Point Arithmetic*. IEEE Std 754-2019 (Revision of IEEE 754-2008). Institute of Electrical and Electronics Engineers. ISBN: 978-1-5044-5924-2. DOI: 10.1109/IEEESTD.2019.8766229.

Kamp, Tobias, Johannes Ultsch, and Jonathan Brembeck (2023). "Closing the Sim-to-Real Gap with Physics-Enhanced Neural ODEs". In: *Proceedings of the 20th International Conference on Informatics in Control, Automation and Robotics*. SCITEPRESS - Science and Technology Publications. DOI: 10.5220/0012160100003543.

Kurzbach, Gerd et al. (2023-10). "Design proposal of a standardized Base Modelica language". In: *Proceedings of the 15th International Modelica Conference*. Vol. 204. Linköping Electronic Conference Proceedings. Modelica Association and Linköping University Electronic Press, pp. 469–477. DOI: 10.3384/ecp204469.

Lenord, Oliver et al. (2021-09). "eFMI: An open standard for physical models in embedded software". In: *Proceedings of 14th Modelica Conference 2021*. Vol. 181. Linköping Electronic Conference Proceedings. Modelica Association and Linköping University Electronic Press, pp. 57–71. DOI: 10.3384/ecp2118157.

Otter, Martin and Hilding Elmqvist (2017-05). "Transformation of Differential Algebraic Array Equations to Index One Form". In: *Proceedings of the 12th International Modelica Conference*. Vol. 132. Linköping Electronic Conference Proceedings. Modelica Association and Linköping University Electronic Press, pp. 565–579. DOI: 10.3384/ecp17132565.

Rai, Rahul and Chandan K. Sahu (2020). "Driven by Data or Derived Through Physics? A Review of Hybrid Physics Guided Machine Learning Techniques With Cyber-Physical System (CPS) Focus". In: *IEEE Access* 8, pp. 71050–71073. DOI: 10.1109/access.2020.2987324.

Raissi, M., P. Perdikaris, and G.E. Karniadakis (2019-02). "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* 378, pp. 686–707. DOI: 10.1016/j.jcp.2018.10.045.

Ruggaber, Julian et al. (2023-02). "AI-For-Mobility—A New Research Platform for AI-Based Control Methods". In: *Applied Sciences* 13.5. DOI: 10.3390/app13052879.

The MISRA Consortium (2023-04). *MISRA C:2023 – Guidelines for the use of the C language in critical systems*. Third edition, Second revision. The MISRA Consortium Limited. ISBN: 978-1-911700-09-8.

Thummerer, Tobias, Johannes Stoljar, and Lars Mikelsons (2022-10). "NeuralFMU: Presenting a Workflow for Integrating Hybrid NeuralODEs into Real-World Applications". In: *Electronics* 11.19, p. 3202. DOI: 10.3390/electronics11193202.

Ultsch, Johannes, Andreas Pfeiffer, et al. (2024-08). "Reinforcement Learning for Semi-Active Vertical Dynamics Control with Real-World Tests". In: *Applied Sciences* 14.16. DOI: 10.3390/app14167066.

Ultsch, Johannes, Julian Ruggaber, et al. (2021-11). "Advanced Controller Development Based on eFMI with Applications to Automotive Vertical Dynamics Control". In: *Actuators* 10.11. DOI: 10.3390/act10110301.

Willard, Jared et al. (2022-11). "Integrating Scientific Knowledge with Machine Learning for Engineering and Environmental Systems". In: *ACM Computing Survey* 55.4. Revised and extended version of "Integrating Physics-Based Modeling With Machine Learning: A Survey". DOI: 10.1145/3514228.

[30] https://www.bmbf.de

DOI
10.3384/ecp218555

A Study on Vehicle Suspension Loads Prediction Method Based on Hybrid Road Simulation using Modelica Library and FMI

Minsu Hyun[1]

[1]Hyundai Motor Company, South Korea, mshyun@hyundai.com

Abstract

This study presents a method for predicting vehicle suspension component loads at the early design stage. A hybrid road simulation combines road load data acquired from a reference vehicle with the Time Waveform Replication (TWR) technique to generate virtual equivalent road profiles. The TWR was implemented in Python, and a multibody dynamics vehicle model developed using Modelon's Vehicle Dynamics Library was used to simulate chassis response. Integration and iterative simulation between the TWR system and the vehicle model were conducted via Functional Mock-up Units using the Python FMI library, FMPy. These virtual inputs were applied to a virtual test rig. In this study, road load data from a reference vehicle were used to derive the input signals, which were then applied to simulate the suspension loads of a target vehicle. Simulation results were validated against measurement data to confirm the effectiveness of the proposed method.

Keywords: Suspension Component Loads, Hybrid Road Simulation, TWR(Time Waveform Replication), Four-Post Test Rig, Virtual Testing Lab, Virtual Proving Ground, FMI(Functional Mock-up Interface), FMPy, Modelica Library

1 Introduction

In the early stages of vehicle development, where physical test vehicles are not available, it is essential to predict the loads input into suspension components to evaluate the durability performance of the suspension system. The predicted suspension component loads are used not only for evaluating the severity and life expectancy of each suspension part, but also for the development of durability test modes at the vehicle level, as well as test methods at the system, module, and component levels. For this reason, accurate prediction of suspension component loads is critical, and technologies for such predictions have been actively researched for decades.

As shown in Figure 1, the technologies used to predict suspension component loads can be broadly categorized into two types.

The first is the Virtual Testing Lab (VTL) approach, where measured wheel loads are directly applied to the wheel center of a fixed body model to predict component loads. This method has the advantages of simple analysis and the ability to use realistic drive signals, but it cannot consider changes in wheel loads due to design modifications or interactions between front and rear suspensions.

In contrast, the Virtual Proving Ground (VPG) method predicts component loads without any measured data by utilizing highly accurate road, tire, driver, and vehicle models. Although it can be applied to any type of vehicle, its accuracy heavily depends on the fidelity of the tire, road, and driver models, and it is difficult to accurately replicate the driving path, speed, and steering of a real test driver, making the method complex to operate.

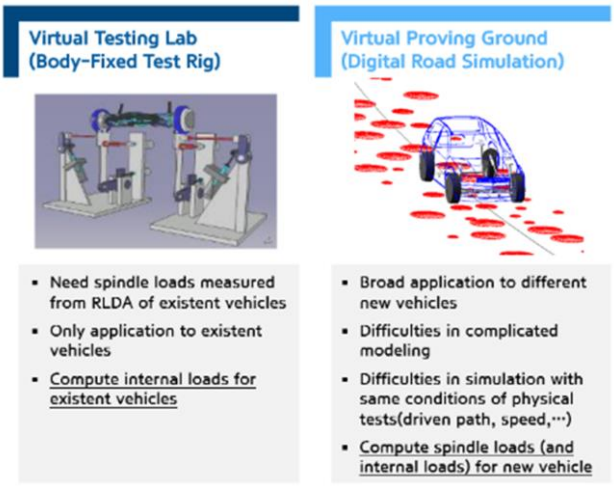


Figure 1. Types of component load prediction simulations.

This study proposes a hybrid road simulation-based suspension load prediction method that overcomes the limitations and leverages the strengths of both VTL and VPG. Section 2.1 introduces the fundamental concept of hybrid road simulation using FMI and Modelica Library. Section 2.2 explains the Time Waveform Replication (TWR) algorithm required for hybrid road simulation. Section 2.3 presents the virtual test rig and simulation method. Sections 2.4 and 2.5 present case studies using

RLDA (Road Load Data Acquisition) data from the reference vehicle and the target vehicle to validate the proposed methodology.

2 Methodology

2.1 Overview of Hybrid Road Simulation

Hybrid road simulation is a technique that utilizes both RLDA-measured data and a vehicle model. As shown in Figure 2, the core process of hybrid road simulation involves generating equivalent virtual road profiles or drive signal on a virtual test rig to replicate target signals from RLDA data using the simulation model of the reference vehicle. The inputs generated are then applied to the simulation model of the target vehicle to predict component loads. This approach should be applied to similar platforms to ensure accuracy, but it offers the advantages of simulating realistic responses without requiring separate road or driver models, thus reducing model dependency.

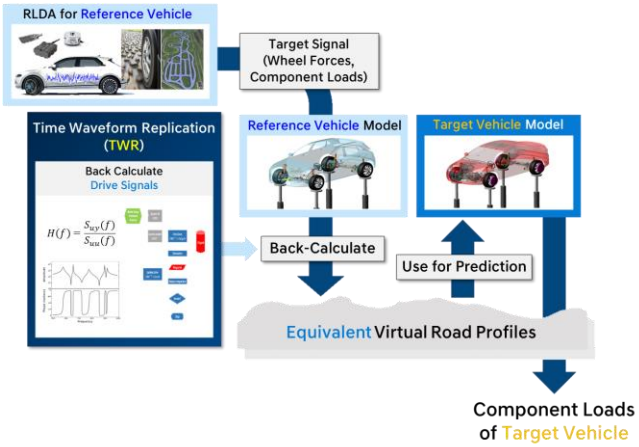


Figure 2. Hybrid road simulation process.

To perform hybrid road simulation, a TWR algorithm is required. The TWR methods include an iteration-based technique using Frequency Response Functions (FRF) and a control-based approach using PID or Iterative Learning Control (ILC). The FRF-based method requires more time due to its iterative nature but is easier to use and offers good reproducibility. The control-based method eliminates the need for iterations but requires optimization of the controller to improve tracking performance.

2.2 Time Waveform Replication (TWR)

In this study, the FRF-based iterative method among TWR techniques was employed to extract drive signals for the hybrid road simulation. Figure 3 illustrates the TWR process, which consists of four main steps: (1) FRF generation using white-pink noise, (2) initial drive signal creation, (3) iterative drive signal update, and (4) convergence check using RMSE thresholds. Each step is described in detail in the following subsections.

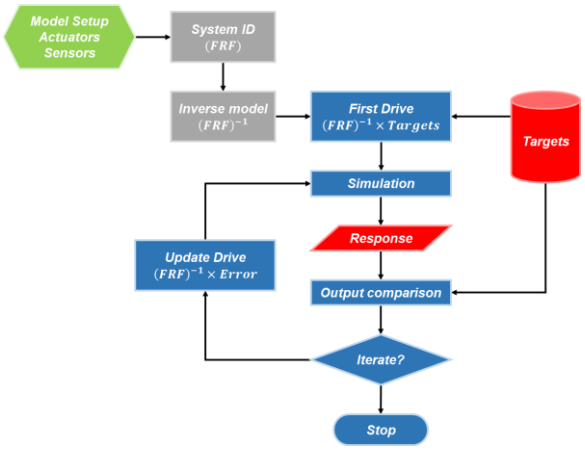


Figure 3. Time waveform replication (TWR) process.

2.2.1 FRF Generation Using White-Pink Noise

In the first step, white-pink noise is applied to the input channels of the vehicle model to extract responses from the output channels and generate an FRF model. The white-pink noise $WPN(f)$ has a constant magnitude from the start frequency (f_{st}) to the border frequency (f_b) and gradually decreases from f_b to the end frequency (f_{en}), as shown in Figure 4. Equation (1) defines WPN(f) across these segments, where the attenuation after f_b depends on the exponential parameter p.

$$WPN(f) = \begin{cases} \left(\frac{1}{f_b}\right)^{2p}, & f_{st} \le f < f_b \\ \left(\frac{1}{f}\right)^{2p}, & f \ge f_b \end{cases} \quad (1)$$

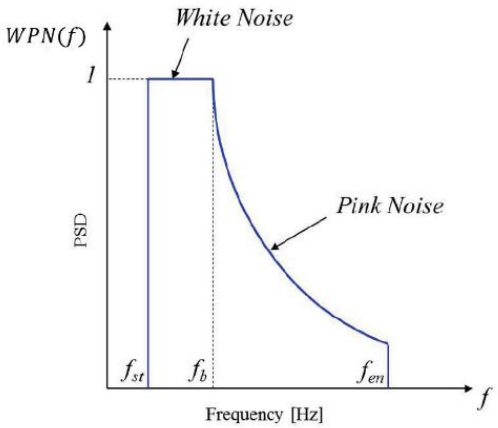


Figure 4. White-Pink Noise.

The process for FRF generation is shown in Figure 5.

DOI 10.3384/ecp218567

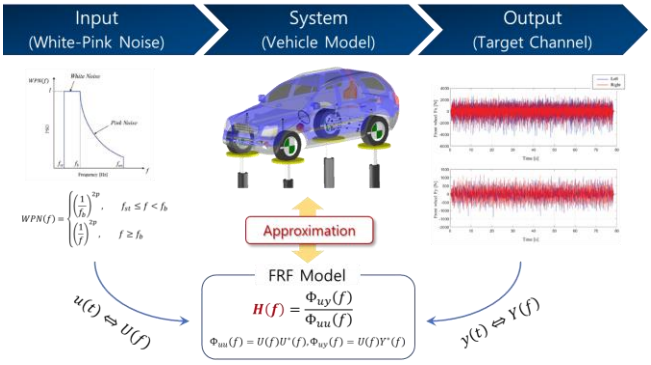


Figure 5. System identification (FRF generation) process.

In the FRF generation process, the auto-spectrum represents the frequency-domain power distribution of a signal, typically used to analyze the input characteristics. The cross-spectrum, on the other hand, quantifies the frequency-domain relationship between the input and output signals. These spectral components are used to compute the Frequency Response Function (FRF) by dividing the cross-spectrum by the auto-spectrum, enabling the identification of system dynamics. The cross-spectrum and auto-spectrum are obtained as shown in Equations (2) and (3):

$$\Phi_{uy}(f) = U(f)Y^*(f) \tag{2}$$

$$\Phi_{uu}(f) = U(f)U^*(f) \tag{3}$$

Here, $U(f)$ and $Y^*(f)$ are the Fourier transforms of $u(t)$ and $y(t)$, respectively, and * denotes the complex conjugate. The FRF $H(f)$ is then derived as:

$$H(f) = \frac{\Phi_{uy}(f)}{\Phi_{uu}(f)} \tag{4}$$

2.2.2 Initial Drive Signal Generation

The initial drive signal is generated using the inverse of the FRF and the inverse Fourier transform as shown in Equation (5):

$$u(t) = d \cdot \mathcal{F}^{-1}\left(H(f)^{-1}\mathcal{F}\left(S \cdot w(t)\right)\right) \tag{5}$$

Here, $H(f)^{-1}$ is the inverse FRF, $\mathcal{F}$ is the Fourier transform, $w(t)$ is the target signal, S and d are weights for the target and drive signals, respectively. The signal is valid within linear system regions, but nonlinear behavior introduces error, necessitating additional correction.

2.2.3 Drive Signal Update via Iteration

To correct errors between target and actual responses, an updated drive signal is computed iteratively. Equation (6) represents the iterative update of the drive signal in the TWR process. The initial drive signal is computed by applying the inverse FRF to the target signal. In subsequent iterations, the error between the simulated response and the target signal is calculated and passed through the inverse FRF to generate a correction signal. This correction is then added to the previous drive signal, progressively improving the match between the simulated and target responses. This approach is grounded in frequency-domain system identification and iterative learning control principles and has been widely used in road load replication studies for its convergence reliability and simplicity. The $k+1$-th drive signal is calculated as in Equation (6):

$$u_{k+1}(t) = u_k(t) + d \cdot \mathcal{F}^{-1}\left(H(f)^{-1}\mathcal{F}\left(E \cdot \left(y(t) - S \cdot w(t)\right)\right)\right) \tag{6}$$

Here, E is the error weighting parameter, and $y(t)$ is the output of the k-th iteration. The process continues until the RMSE (Root Mean Square Error) of each output channel is below the threshold.

2.3 Virtual Test Rig for Hybrid Road Simulation

The virtual test rig used in this study was constructed using a full vehicle model based on Modelon's Vehicle Dynamics Library (VDL), which supports high-fidelity multibody dynamics (MBD) simulation. This MBD-based model captures realistic vehicle responses and suspension loads by simulating the actual kinematics and compliance characteristics of suspension and chassis systems. The suspension architecture of the reference and target vehicles consists of a McPherson strut configuration at the front and a multi-link suspension at the rear, as illustrated in Figures 6 and 7. This combination was selected to reflect a typical modern passenger vehicle setup, where the front provides packaging efficiency and steering simplicity, while the rear enables better ride comfort and lateral stability.



Figure 6. Front suspension model (McPherson strut) in VDL.

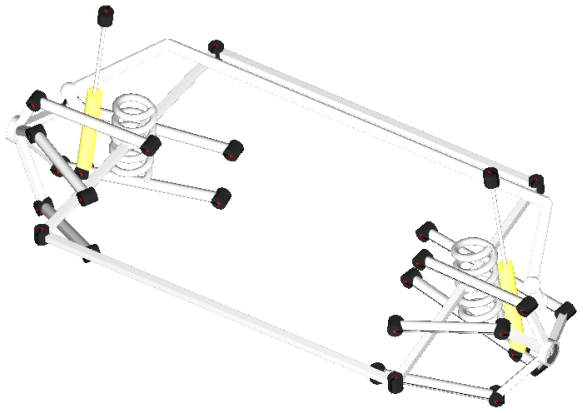


Figure 7. Rear suspension model (multi-link) in VDL.

While developing the suspension model, it was identified that the default bump stop element in Modelon's VDL lacks support for nonlinear force–displacement characteristics, which are critical for accurately capturing suspension component loads during durability simulations. To address this limitation, a custom bump stop model was implemented using Modelica. This component incorporates a user-defined nonlinear force–displacement curve using a lookup table and becomes active when the damper compression exceeds a predefined bump clearance. This modeling approach enables accurate replication of the sharp load increase that occurs during bottoming events, which significantly influences suspension component fatigue and durability assessment. To support this, two dedicated submodels were constructed and are illustrated in Figures 8 and 9.

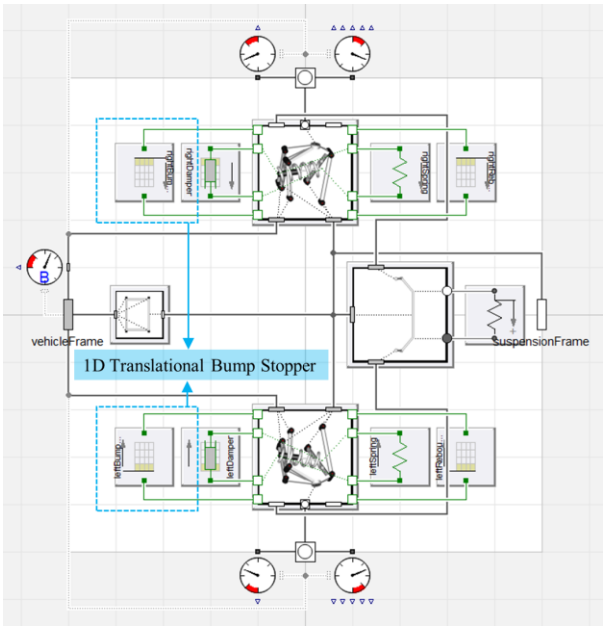


Figure 8. 1D translational bump stopper model.

Figure 8 shows the custom 1D translational bump stopper model used in the rear suspension system. The element accepts the force–displacement curve and is activated only when bump clearance is exceeded.

Figure 9 presents the kinematic model in the rear suspension linkage model that computes the bump stop compression displacement. In this configuration, the bump stop is mounted not on the shock absorber upper bushing but directly on the vehicle body. As such, the damper stroke alone does not reflect bump stop engagement. Therefore, an additional kinematic model was developed to calculate the relative displacement between the lower control arm and the chassis hardpoint. This value serves as the effective compression input to the bump stop force model.

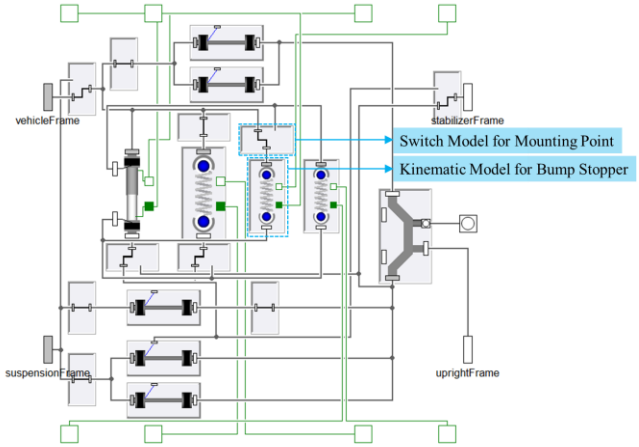


Figure 9. Kinematic model for bump stopper.

This approach ensures that the bump stop responds only under physically accurate conditions, enhancing the realism of load prediction and its influence on suspension durability performance.

The virtual test rig is configured as a 4-post setup, capable of applying vertical displacements at the tire contact patches through equivalent virtual road profiles. Additionally, actuators are installed at each wheel center to apply longitudinal and lateral forces, as well as moments. This configuration enables comprehensive reproduction of real-world wheel center load conditions and supports the simulation of both road-induced and control-induced excitations. Figure 10 shows the layout of this virtual durability test rig within the simulation environment.

DOI
10.3384/ecp218567

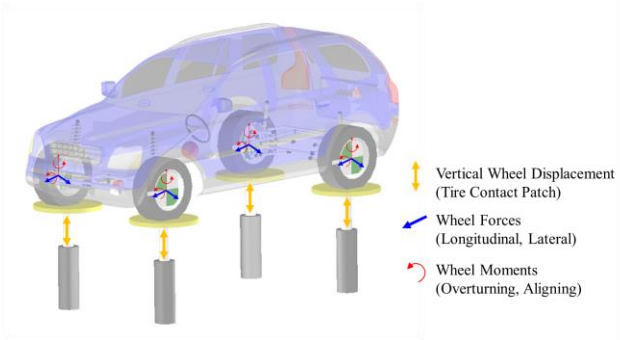


Figure 10. Configuration of virtual test rig.

To enable hybrid road simulation, the entire vehicle model and test rig were exported as Functional Mock-up Units (FMUs) using the FMI standard. These FMUs were integrated with a Python-based TWR algorithm, implemented using the FMPy library. This setup allows for automated iteration of drive signals to match target RLDA responses. The overall simulation framework and data flow between Python and the FMU-based vehicle model are depicted in Figure 11.

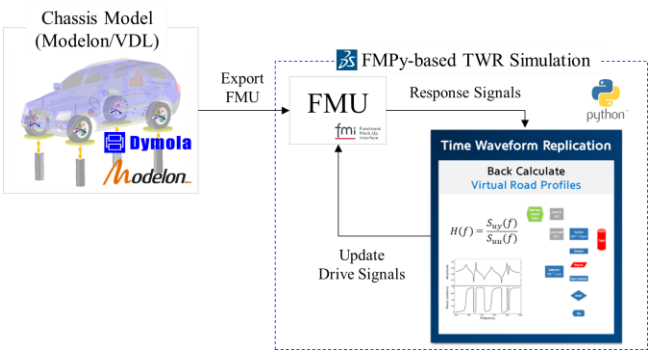


Figure 11. TWR simulation based on FMPy.

The TWR algorithm operates by comparing the actual response of the virtual vehicle to the target RLDA signals. It continuously updates the drive signals by feeding new input signals, stored in *.mat format, into the FMU simulation. These updates are computed using an FRF-based iterative optimization approach, which enhances tracking fidelity for complex time-domain signals. During each TWR iteration, the Python controller:

Step 1. Executes a simulation with the current drive signal.

Step 2. Evaluates the error between target and simulated output (e.g., wheel center forces).

Step 3. Updates the drive signal using the inverse FRF and re-simulates until convergence.

This tightly integrated Python-FMI workflow provides robust co-simulation performance while ensuring that realistic suspension dynamics are captured through the MBD-based virtual rig.

2.4 Application to Hybrid Road Simulation

2.4.1 Drive Signal Update via Iteration

For the reference vehicle, equivalent virtual road profiles and drive signals capable of reproducing 20 channels of wheel forces and moments were generated based on Belgian road RLDA data. Predicted and measured component and wheel loads were compared. Figures 12 and 13 show the improvement in front and rear vertical wheel force prediction accuracy across iterations. Figure 14 illustrates the RMSE convergence behavior across 10 iterations for 20 wheel load channels. All channels showed a consistent downward trend in RMSE, indicating stable convergence. While convergence patterns were similar overall, the final RMSE values for vertical loads remained slightly higher, stabilizing below 12%. This level of accuracy is considered reasonable for durability simulation purposes. In contrast, wheel center longitudinal and lateral loads, as well as aligning and overturning moments, achieved lower residual RMSE values, typically below 6%. These results confirm the robustness of the FRF-based iterative TWR algorithm in minimizing signal error across diverse load types.

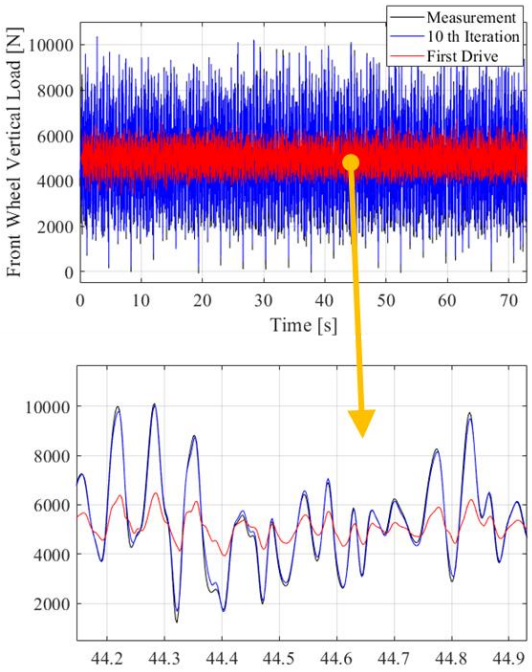


Figure 12. Predicted results of front wheel center vertical load.

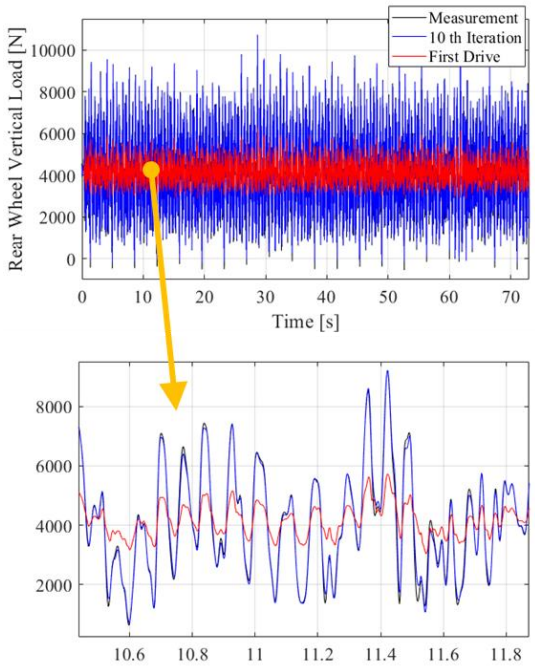


Figure 13. Predicted results of rear wheel center vertical load.

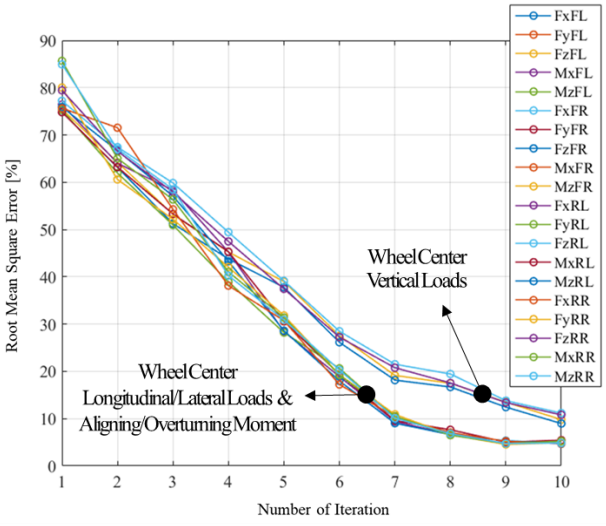


Figure 14. RMSE history of wheel loads and moments.

In this study, component damage was calculated as a normalized pseudo damage metric using rainflow cycle counting and Miner's rule. The damage values represent relative fatigue severity based on the simulated load time histories, allowing comparison between measured and predicted results without requiring material-specific S-N curves. Table 1 summarizes RMS and normalized pseudo damage values of measured and simulated component loads, confirming the high accuracy of the generated drive signal and road profile.

Table 1. TWR simulation results of reference vehicle.

Channel	*Index*	*Simulation*	*Target*	*Error*
Front Wheel Center Vertical Load	RMS	1729.62	1691.39	2.26 %
	Pseudo Damage	1.34	1.24	8.06 %
Rear Wheel Center Vertical Load	RMS	2074.84	2048.02	1.31 %
	Pseudo Damage	1.81	1.7	6.47 %
Front Lower Arm Boll Joint Fx Load	RMS	875.29	866.18	1.05 %
	Pseudo Damage	0.33	0.32	3.13 %
Real Assist Arm Axial Load	RMS	1634.37	1615.21	1.19 %
	Pseudo Damage	2.88	2.80	2.86 %
Rear Lower Arm Axial Load	RMS	1548.75	1571.01	1.42 %
	Pseudo Damage	2.80	2.96	5.40 %

2.4.2 Simulation for Target Vehicle

The extracted drive signals and road profiles were applied to the target vehicle model to predict component and wheel loads. Figures 15 to 19 present range pair graphs showing load distribution and cumulative damage. While vertical loads and front lower arm ball joint loads closely matched measured data, some discrepancies were observed in rear component loads, primarily due to sensitivity of damage metrics and reliance on reference vehicle measurements for lateral/longitudinal forces.

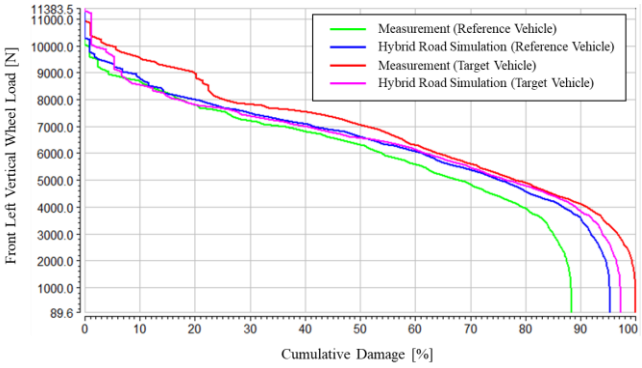


Figure 15. Damage of front wheel center vertical load.

DOI
10.3384/ecp218567

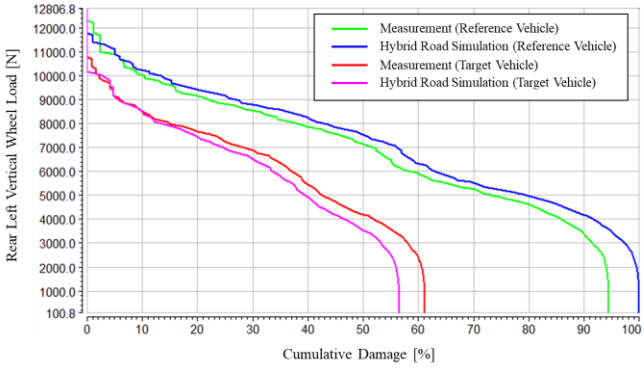


Figure 16. Damage of rear wheel center vertical load.

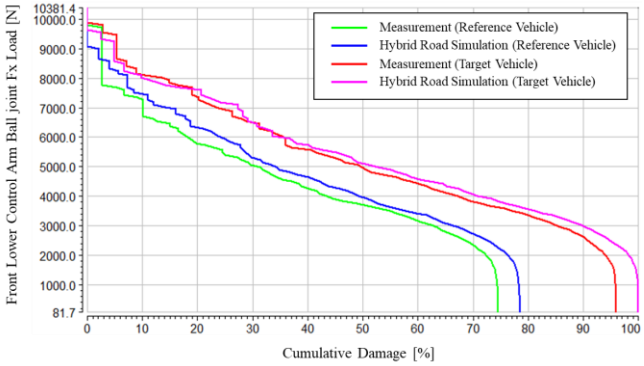


Figure 17. Damage of front lower arm ball joint load.

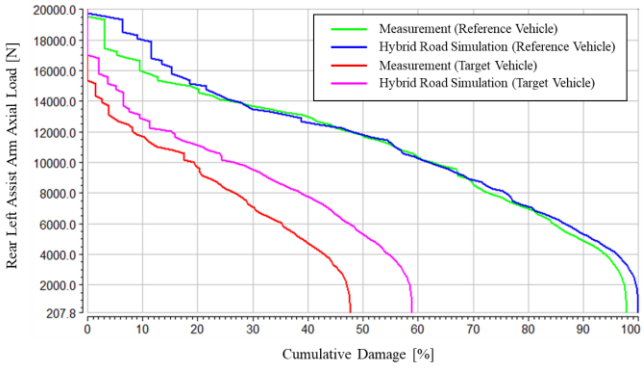


Figure 18. Damage of rear assist arm axial load.

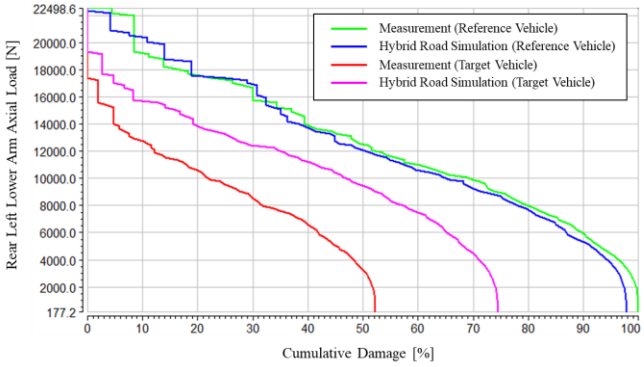


Figure 19. Damage of rear lower arm axial load.

Table 2 presents the RMS and damage values for major suspension components, showing that trends in damage and RMS variation were reproduced reasonably well even for the rear suspension.

Table 2. TWR simulation results of target vehicle.

Channel	*Index*	*Simulation*	*Target*	*Error*
Front Wheel Center Vertical Load	RMS	1726.78	1842.046	6.26 %
	Pseudo Damage	1.36	1.4	2.86 %
Rear Wheel Center Vertical Load	RMS	1785.68	1819.82	1.87 %
	Pseudo Damage	1.02	1.1	7.27 %
Front Lower Arm Boll Joint Fx Load	RMS	875.63	858.28	2.02 %
	Pseudo Damage	0.424	0.372	13.98 %
Real Assist Arm Axial Load	RMS	1291.52	1251.66	3.18 %
	Pseudo Damage	1.69	1.37	23.36 %
Rear Lower Arm Axial Load	RMS	1419.09	1345.80	5.45 %
	Pseudo Damage	2.40	1.75	37.14 %

3 Conclusion

This study presented a hybrid road simulation method for predicting vehicle suspension component loads by integrating road load data and multibody vehicle models through FMI-based co-simulation. By employing an FRF-based iterative TWR algorithm, equivalent virtual road profiles were generated using RLDA data from a reference vehicle and applied to a target vehicle model exported as an FMU.

The proposed methodology offers several advantages relevant to the Modelica and FMI community:

- **FMI-based modular integration**: The approach demonstrates a practical application of the FMI standard using FMPy to couple a Python-based TWR algorithm with a Modelica-based full vehicle model, ensuring seamless interoperability in a simulation workflow.

- **Efficient virtual durability testing**: By leveraging existing RLDA data and applying it through a virtual test rig configuration, the method enables early-stage durability analysis without the need for physical prototypes or detailed driver models.

- **System-level fidelity with manageable complexity**: Compared to traditional Virtual Proving Ground (VPG) approaches, the hybrid method maintains accuracy in load prediction while reducing the dependency on high-fidelity subsystems like road or tire models.

Future work will focus on extending this FMI-based hybrid simulation framework to cover broader driving scenarios and enhancing prediction robustness, especially for design-dependent variables such as lateral and longitudinal load components. This research illustrates how the use of standardized model exchange formats like FMI can accelerate the development and validation of vehicle durability simulation methods in industrial applications.

References

Schultz, R., & Carter, S. (2023). "A MIMO time waveform replication control implementation." Society for Experimental Mechanics Annual Conference and Exposition. Cham: Springer Nature Switzerland.

Bäcker, M., Langthaler, T., Olbrich, M., Oppermann, H. (2005). "The Hybrid Road Approach for Durability Loads Prediction". SAE Technical Paper. No. 2005-01-0628

Modelica Association (2013-07). *Modelica – A Unified ObjectOriented Language for Systems Modeling. Language Specification Version 3.2 Revision 2*. Tech. rep. Linköping: Modelica Association. URL: https://www.modelica.org/documents/ModelicaSpec32Revision2.pdf.

Kugu, O., Zhou, S., Nowak, R., Müller, G., Reiterer, S. H., Meierhofer, A., ... & Grafinger, M. (2023). An fmi-and ssp-based model integration methodology for a digital twin platform of a holistic railway infrastructure system. In Modelica Conferences (pp. 717-726).

Hatledal, L. I., Zhang, H., Styve, A., & Hovland, G. (2018). Fmi4j: A software package for working with functional mock-up units on the java virtual machine. In *The 59th Conference on Simulation and Modelling (SIMS 59)*. Linköping University Electronic Press, Linköpings universitet.

Hyun, M., & Heo, S.-J. (2022). "Modular Architecture Design of Chassis System Using Database." International Journal of Automotive Technology, 23, 225–232. Cham: Springer Nature Switzerland. URL: https://link.springer.com/article/10.1007/s12239-022-0019-9

Park, S., Jeon, Y., Kang, D.-o., Hyun, M., & Heo, S.-J. (2019). "Predicting the Vehicle Performance at an Early Stage of Development Process via Suspension Bushing Design Tool." Modelica Conference 2019. Linköping University Electronic Press. URL: https://ep.liu.se/en/conference-article.aspx?series=157&issue=157&Article_No=86

Ko, K.-c., Park, S., Kim, J., Jeong, Y., Hyun, M., & Heo, S.-J. (2017). "Development of Hierarchal Commercial Vehicle Model for Target Cascading Suspension Design Process." Modelica Conference 2017. Linköping University Electronic Press. URL: https://ep.liu.se/en/conference-article.aspx?issue=132&Article_No=49

Br(e)aking the Boundaries of Physical Simulation Models: Neural Functional Mock-up Units for Modeling the Automotive Braking System

Tobias Thummerer[1] Fabian Jarmolowitz[2] Daniel Sommer[2] Lars Mikelsons[1]

[1]Chair of Mechatronics, University of Augsburg, Germany, {tobias.thummerer,lars.mikelsons}@uni-a.de
[2]Robert Bosch GmbH, Germany, {fabian.jarmolowitz,daniel.sommer4}@de.bosch.com

Abstract

Testing real hardware and simulation models in combination in a software- or hardware-in-the-loop set-up is challenging. One of the key factors is the high demand for accuracy in the simulation model. If classical modeling based on physical principles is not sufficient to reach the desired level of accuracy, *hybrid modeling*, the combination of physical simulation models and machine learning can be applied. In this publication, we train a hybrid model for a controlled electric motor within the electro-hydraulic braking system of a car under the conditions and restrictions of a real engineering application in the field. We apply state-of-the-art modeling patterns for this, and further extend them with application specific methodological optimizations. Finally, we investigate and show the quantitative and qualitative advantages of the proposed approach for this specific application, resulting in a gain in accuracy by multiple factors.

Hybrid Modeling, Graybox Modeling, Scientific Machine Learning, Functional Mock-Up Unit, Functional Mock-up Interface, Braking System, Automotive

1 Introduction

Most simulation models in the automotive environment are built based on physical equations. This has the advantage that the model builder, as well as other engineers, have an intuitive understanding of the model and its parameters. As these physical equations also have very broad validity, most physical models can also generate correct results outside the required working range (extrapolation), making them robust in various application scenarios. When creating models, equations are chosen on the basis of assumptions, i.e. simplifying assumptions are made. This introduces a modeling error that leads to deviations between the simulation model and the real system. On the other hand, there are machine learning approaches that can theoretically generate arbitrarily accurate results depending on the selected model and model dimension if a sufficient amount of data is given. This flexibility comes at the cost of poor interpretability due to the black-box nature of most machine learning models like arifical neural networks (ANNs). The field of *hybrid modeling*, which is part of *scientific machine learning*, attempts to combine the advantages of both modeling and simulation approaches, e.g., to create more accurate, yet interpretable models.

1.1 Use Case: Bosch Integrated Power Brake

The Integrated Power Brake (IPB) is a vacuum-independent and electro-hydraulic brake control system that combines brake boosting and brake control functions in one piece of hardware. By applying the brake pedal, the control unit calculates the driver's brake request based on internal sensors. This brake request is implemented by activating a pressure build-up unit consisting of an electric motor, a gearbox, and a hydraulic piston. Finally, brake pressure is transferred to the wheel brakes via brake pipes to decelerate the vehicle. For more information on the system, see Robert Bosch GmbH (2025) and Robert Bosch GmbH (2024).

The developed control software is tested using software-in-the-loop simulation. This requires a model of the hardware that generates sufficient quality of results. A very detailed physical model only makes sense to a certain extent, as otherwise the required simulation performance cannot be achieved. Therefore, strict requirements are given for both simulation accuracy and performance. A hybrid modeling approach should be able to fulfill these requirements.

The model used for the use case contains the electric motor, including the mechanical parts of the pressure build-up unit. The physical model calculates the position, speed, torque, and current of the rotor. The motor is controlled by a motor controller that runs software that calculates control signals based on the position of the rotor. This motor controller is implemented in software for the use case considered.

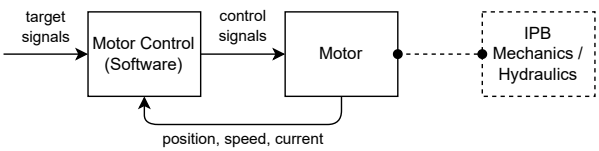


Figure 1. IPB motor control.

The model itself poses various challenges:

- The system is very sensitive with respect to the initial state of the rotor position, as the control signals are calculated based on this. Even small deviations make the motor run *out-of-sync*, as can be observed for the physical model during experiments (s. Sec. 4).
- The control signals are sampled with a frequency of $10^4\,Hz$, triggering time events at this frequency.
- The nonlinear physical model is marginally stable with real zero eigenvalues independent of the system state, so for the entire operation range of the motor.
- Special model conditions (such as reaching end-stop) cause additional state events.

The system consists of two mechanical and two electrical states. The mechanical states for the electric motor are the angular position and the angular speed of the motor rotor. In addition, the inputs to the electric motor are pulse-width modulation (PWM) signals for each motor phase, the motor supply voltage, and the time-varying mechanical load caused by the hydraulic subsystem.

The simulation model is given as functional mock-up unit (FMU), which is introduced in detail in the next section. The use of a FMU for the considered use case not only allows one to simulate the motor model but also to later extend it by a machine learning model without actually sharing the model equations and parameters. This is because the FMU contains only compiled machine code. In fact, the use case shown here was processed without knowing the physical model or its parameters. This demonstrates a significant advantage of the method, as models can be improved by hybrid modeling without disclosing the corresponding intellectual property.

1.2 Functional Mock-up Interface (FMI)

The Functional Mock-up Interface (FMI) is an open, tool-independent standard (container format) for exchanging and co-simulating dynamic models, usually in the form of ordinary differential equations (ODEs), across different simulation systems and tools. Developed initially in 2008-2011, FMI has become the de facto industry standard for model exchange and co-simulation, supported by more than 230 different simulation tools (Modelica Association 2020; Bertsch 2022; Modelica Association 2023). FMI supports two main types of FMUs: Co-simulation (CS) and Model-exchange (ME). The ME-FMU offers an interface similar to the right-hand side of a system of ODEs, for example providing the state derivative for a given state. This allows for the complementation of the ODE and its underlying physics with data-driven components from machine learning to improve, for example, the accuracy (Thummerer, Mikelsons, and Kircher 2021).
FMI 3.0 has introduced several features that enables one to integrate AI models in simulation, e.g. by updating parameters much more efficiently than in previous versions. For example, with the added support of adjoint derivatives, FMUs can be integrated in automatic differentiation frameworks, as demonstrated in *FMISensitivity.jl* [1].

1.3 Neural ODEs

In machine learning, using an ANN to learn for the right-hand side of an ODE is referred to as *neural ODE* (R. T. Q. Chen et al. 2018), compare Fig. 2. The gradient required for training neural ODEs is typically determined by back-propagation through the numerical solver (reverse mode automatic differentiation) or by applying adjoint methods. This concept was later extended to support event-ODEs as well (R. T. Chen, Amos, and Nickel 2020). Numerous publications show that neural (ordinary) differential equations are an excellent tool to learn dynamical system behavior (Ramadhan et al. 2023; Tac, Sahli Costabal, and Tepole 2022; Xie, Parlikad, and Puri 2019).

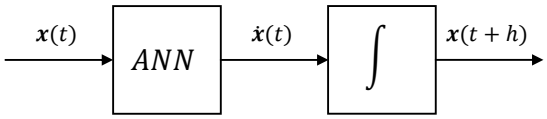


Figure 2. Within neural ODEs, the right-hand side of an ODE is expressed by an ANN and solved by a numerical integration algorithm (ODE solver). Based on a given state $\boldsymbol{x}(t)$, the ANN computes a state derivative $\dot{\boldsymbol{x}}(t)$, which is integrated to the next system state $\boldsymbol{x}(t+h)$ by performing a step of h. Figure adapted from (Thummerer, Kolesnikov, et al. 2023).

1.4 UODE / PeN-ODE

Compared to the concept of neural ODEs, one can conclude that, besides ANNs, other universal approximators are possible as well to approximate the right-hand side of an ODE. Further, the right-hand side can contain a priori knowledge in the form of equations. This concept was introduced as universal ordinary differential equation (UODE) (Rackauckas et al. 2021) together with a software framework. This framework serves as a software foundation for *FMIFlux.jl*[2], which is used within this application. Within the *OpenSCALING* research project (s. Sec. Funding), a special kind of UODE is introduced, combining explicitly ANNs and physical simulation models to be solved by an ODE solver, referred to as physics-enhanced neural ODE (PeN-ODE). Similar and related research was performed in Sorourifar et al. (2023) and Thummerer, Kolesnikov, et al. (2023), using the same terminology. A detailed discussion on PeN-ODEs can also be found in (Hofmann and Mikelsons 2025) and the model investigated in this paper can be interpreted as PeN-ODE.

[1] `https://github.com/ThummeTo/FMISensitivity.jl`
[2] `https://github.com/ThummeTo/FMIFlux.jl`

DOI 10.3384/ecp218575

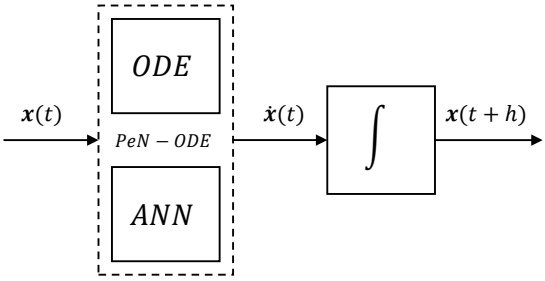


Figure 3. A physics-enhanced neural (PeN)-ODE, which consists of a ODE, an ANN and the ODE solver. Based on a given state $\boldsymbol{x}(t)$, the PeN-ODE computes a state derivative $\dot{\boldsymbol{x}}(t)$, which is integrated to the next state of the system $\boldsymbol{x}(t+h)$. There are no restrictions on how ODE is connected to the ANN.

1.5 Preliminary Work

ME-FMUs offer a similar interface compared to ODEs and can therefore be intuitively replaced with each other on a conceptual level. However, on the technical level, multiple adjustments are required. In Thummerer, Mikelsons, and Kircher (2021), *neural FMUs* were introduced as the combination of FMUs and ANNs. From the point of view of PeN-ODEs, a neural FMU can be constructed by replacing the ODE by a ME-FMU. In addition, neural FMUs can also be constructed based on CS-FMUs (Thummerer, Mikelsons, and Kircher 2021), but are not further investigated in this contribution.

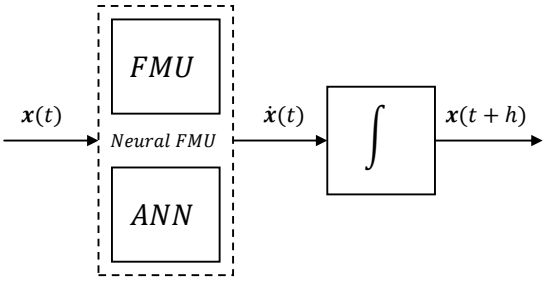


Figure 4. A neural FMU (ME), that consists of a ME-FMU, an ANN and the ODE solver. Based on a given state $\boldsymbol{x}(t)$, the neural FMU computes a state derivative $\dot{\boldsymbol{x}}(t)$, which is integrated into the next state of the system $\boldsymbol{x}(t+h)$. No assumptions on how the FMU is connected to the ANN are made. In addition, multiple FMUs and ANNs can be involved.

In Thummerer, Stoljar, and Mikelsons (2022), multiple training strategies for neural FMUs are investigated. In the following, we briefly explain the methodological foundation that we use as a starting point for the presented extensions. The following decisions are made:

Gates As a starting point for building a hybrid model architecture, the gates topology is used (Thummerer, Stoljar, and Mikelsons 2022), based on a parallel-serial layout (Thummerer and Mikelsons 2025). Here, two gates are applied that control how much of the (original) FMU motor dynamics and how much of the (newly) learned ANN motor dynamics contribute to the resulting hybrid model motor dynamics. The gates are parameterizable and are trained together with other optimization parameters, choosing proper initial parameters is part of hyperparameter optimization.

Pre- and Post-Processing To normalize the values between machine learning and the domain of the physical model, pre- and post-processing is applied online (Thummerer, Stoljar, and Mikelsons 2022). Here, pre- and post-processing corresponds to the introduction of two additional linear operations that shift and scale signals at the domain boundaries (between the physical and machine learning model and vice versa) in order to prevent suboptimal operation ranges for the nonlinearities inside the ANN. For example, if hyperbolic tangent activation is applied (as we used in the later experiment), small values close to zero lead to operation in (almost) linear ranges of this function and therefore neglection of the intended nonlinearity, whereas large values drastically saturate to 1 or -1, leading to a vanishing gradient. Both issues are major obstacles when training (nonlinear) hybrid models.

Interfaces In Thummerer and Mikelsons (2025), it is investigated how models can be combined, and learnable connections between physical and machine learning models are proposed. However, it is shown that this is not always efficient if only a limited amount of data is available. This motivates a clever selection of interface signals for the ANN. Because only little is known about the not modeled physical effect, it is decided to use the following signals as inputs for the ANN:

- The **system state** is used, to allow the ANN to learn its own representation of the right-hand side of the ODE. This corresponds to a parallel-like model architecture.
- The **system inputs**, to allow the learned right-hand side to further depend on the current input.
- The **state derivative** calculated by the FMU, which allows to *correct* the dynamical representation of the system calculated by the physical model. This corresponds to a serial-like model architecture.
- The time is explicitly not used as input to the ANN, because a time-invariant system is assumed, so without explicit time dependency.

The presented decisions result in the architecture visualized in Fig. 5, which is used as the foundation for the considered application.

2 Method

Although the architecture introduced in Fig. 5 was able to produce good results in a very similar form for different application scenarios from mechanical engineering (Thummerer, Stoljar, and Mikelsons 2022; Thummerer,

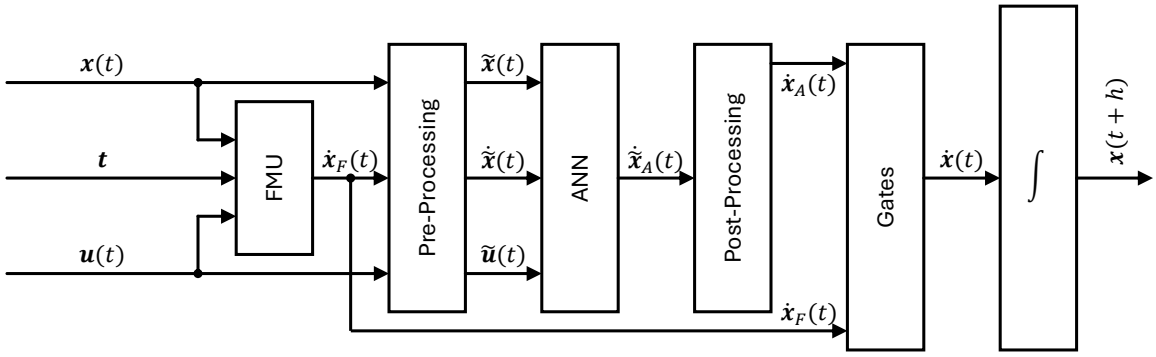


Figure 5. The starting point for derivation of the neural FMU (ME) used in this paper. It features pre- and post-processing layers as well as gates. First, the state $\boldsymbol{x}$, time t, and inputs $\boldsymbol{u}$ are fed into the FMU. The FMU computes a state derivative $\dot{\boldsymbol{x}}_F$, which is pre-processed together with the original state and input values. After pre-processing, the considered values are passed to an ANN, that computes its own state derivative vector $\tilde{\dot{\boldsymbol{x}}}_A$, which is further post-processed to $\dot{\boldsymbol{x}}_A$. The gates layer then controls how much of the ANN dynamics and the FMU dynamics contribute to the overall dynamics of the hybrid model $\dot{\boldsymbol{x}}$, which is finally integrated to the next system state $\boldsymbol{x}(t+h)$ by performing a step with size h in time.

Kolesnikov, et al. 2023) and medical technology (Thummerer, Tintenherr, and Mikelsons 2021), further methodological adjustments must be made for the particularly challenging application case at hand. These are introduced and discussed below.

2.1 Architecture: Learning Input Delay

The measurement data used is recorded at high frequency ($10^4\,Hz$). It is worth mentioning that this sampling frequency is not an excessive choice and is required to capture significant information, for example PWM signals to drive the electric motor. Due to the combination of several sensor signals in the data logger or errors in time synchronization between these sensors, time offsets may occur between the measurement signal of different sensors (see Fig. 6). In general, for small deviations, it is difficult to distinguish these temporal deviations from system dynamics. If the cause cannot be clearly identified (and explained), appropriate correction options can be provided in the architecture for both possible causes. This is achieved by integrating a block for introducing a constant-time offset for the input signal in addition to a neural network for correcting the system dynamics. Mathematically, this is defined simply as follows:

$$\boldsymbol{u}_{out}(\boldsymbol{u}(t),t) := \boldsymbol{u}(t+\Delta t), \tag{1}$$

where Δt is a trainable parameter and $\boldsymbol{u}_{out}$ is the output signal of the delay correction block. Since the time-offset block is parameterized and its parameter Δt can be differentiated, this parameter can be optimized together with the parameters of the neural network in a gradient-based training.

2.2 Architecture: Bypass Dynamics

For the presented application, we decide to only manipulate the *mechanical* part of the system state vector, so the mechanical subsystem, respectively. This is justified

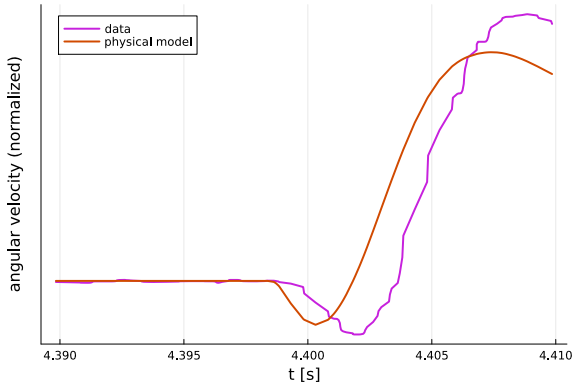


Figure 6. The plot shows the angular velocity data, as well as the output of the simulation model. There is a time shift in between that could be compensated by adding an offset in time to the input signals.

by the fact that the modeling and parameterization of the electrical subsystem are assumed to be accurate, while the mechanical subsystem has a higher level of uncertainty. As a consequence, the electrical part of the state derivative is forwarded to the ODE solver without any manipulation, which corresponds to leaving the right-hand side of the electrical subsystem untouched. To further preserve the second-order ODE structure for the mechanical subsystem, we only allow manipulation of the rotor acceleration (and not the rotor speed), which we achieve by forwarding the unmodified rotor speed to the ODE integrator as well.

Based on the preliminary work summarized in Fig. 5, the proposed input offset correction block and signal forwarding, a model architecture as in Fig. 7 can be deployed to address the presented use case.

2.3 Data Handling

As shown in Fig. 8, real measurement data is often very heterogeneous in terms of information content. For a

DOI
10.3384/ecp218575

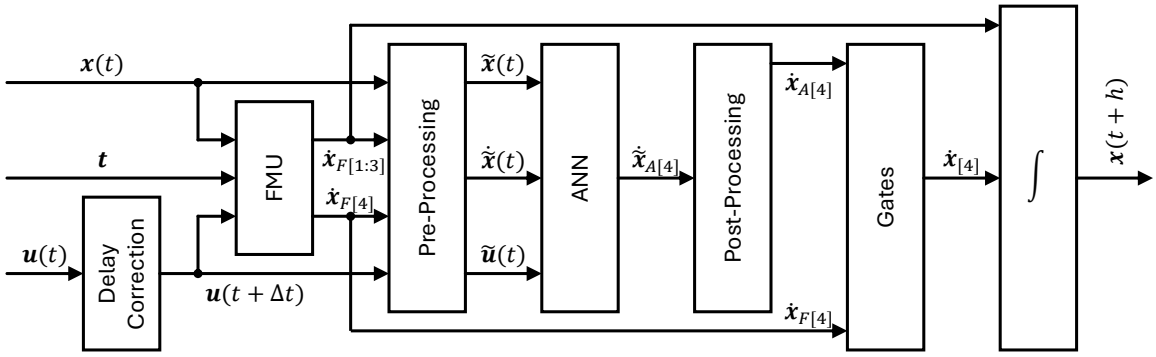


Figure 7. The architecture designed for the presented use case, based on the introduced architecture in Fig. 5 and extended by input delay correction. Not all state derivatives $\dot{x}$ are computed by the ANN and post-processing, only $\dot{x}_{A[4]}$ (the motor rotor acceleration). The others $\dot{x}_{F[1:3]}$ are cached and forwarded directly to the ODE solver.

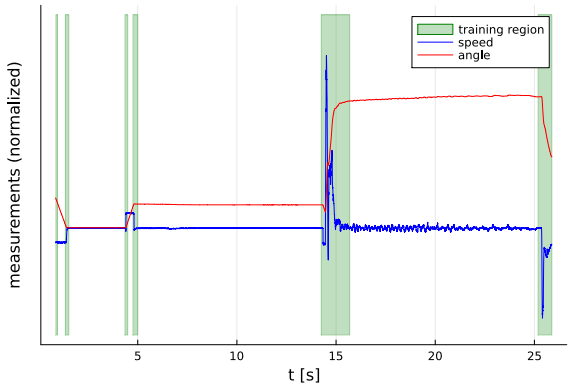


Figure 8. The training data. As data is *heterogneous* (varying information content), only the green regions (that offer information *variance*) are used for training.

large part of the time axis, the system is at rest (or almost at rest). This is due to a delay between the start and end of the recording and the execution of the actual test. In this application, we have decided to thin out the data manually, but we are working in parallel on robust approaches to automate this step. It is important to note that this step is more complex than, for example, filtering only for derivatives and has a significant influence on the quality of the results with such a small amount of training data.

3 Experiment

The presented method is validated in the challenging use case of the IPB introduced in Sec. 1.1. The experiment involves the training process, followed by the comparison of results against measurements of the real physical system (ground-truth data, baseline). To determine good hyperparameters, a hyperparameter optimization is performed.

3.1 Data

For intellectual property reasons, all data plots and error measures within this publication are normalized. The data used for training, validation, and testing was generated under typical development conditions:

- All measurements were performed on the real IPB hardware (prototype) and
- an ECU software measurement tool was used to measure the sensor and actor control quantities, so no additional sensor are required/used.

To show the ability of the presented method to provide good results on a very small data set under conditions of real development in the field, only three measurements with durations of $\approx 12-25\,s$ are used. One trajectory is used for training (gradient determination for optimization), one for validation (evaluating the success of hyperparameter optimization), and one for testing (completely unknown to the model and optimization processes).

3.2 Training

Because the experiment includes a real industrial simulation model that is part of active development, this model cannot be shared. However, the experiment setup is described in the following, to allow for reproduction within similar use cases:

Hyperparameter Optimization For hyperparameter optimization, the optimizer *Hyperband* (Li et al. 2018) is applied with parameters $R = 81$ and $\eta = 3$. The optimizer is paired with a simple random sampler as the inner sampling algorithm[3]. As resource for *Hyperband*, a time span is used for the processed training data, which is limited to a maximum of $100\,s$.

Optimizer For the actual training, the optimizer *Adam* (Kingma 2014) is applied, with step size $\eta \in [10^{-6}, 10^{-4}]$ (best: 3^{-6}), $\beta_1 \in \{0.99, 0.9\}$ (best: 0.99) and $\beta_2 \in \{0.9999, 0.999, 0.99\}$ (best: 0.9999). Stochastic mini-batching is applied as training strategy, with a batch element length of $[0.01\,s, 0.1\,s]$ (best: $0.02\,s$).

[3]Implementation available: `https://github.com/ThummeTo/DistributedHyperOpt.jl`.

Loss function The loss function consists of two separated errors, one for the motor angle and one for the speed. For error determination, mean squared error (MSE) as well as mean absolute error (MAE) (best) are used. To obtain a scalar loss, the loss terms are weighted by a percentage between 0 % and 100 % (best: 70 % speed, 30 % angle) and added together. This allows for loss functions that involve only the speed, only the angle, or a mixture of both.

ANN layout For the ANN, different widths $\{8, 16, 32\}$ (best: 8) and depths $\{2, 3, 4\}$ (best: 4) are tested. Only the *tanh* activation function is applied for all layers.

Gates The gates for the influence of the ANN are trained together with the ANN parameters, but feature different initial openings between $[0\,\%, 50\,\%]$ (best: 30 %).

4 Results & Discussion

After training, we find the following results that we want to discuss in detail in this section. It is important to note that training, validation, and testing are performed *open-loop*. This means that control signals are recorded from the closed-loop (controlled) system, but for training and validation, the motor is operated based on the pre-recorded control signals without involving the controller (remove feedback). This allows us to evaluate the actual accuracy of the hybrid model without controller influences positively distorting the results.

Training, validation, and test data vary w.r.t. different aspects, for example the maximum applied load or the maximum rotor velocity vary by at least 10 % between the individual datasets.

We deliberately decided against baselining against a pure ML model because under the given circumstances (very little data, very few training resources) no presentable ML model can be trained, even if more advanced approaches such as neural ODE are used for this purpose. Please note that training data only captures a part of the state space of the system, and therefore a pure ML approach is by design not able to recover the system behavior beyond the limited training data.

4.1 Validation

The best hyperparameter run values can be found in Sec. 3.2, leads to the following results for the motor angle (s. Fig. 9). The determined value for the time offset (input delay) is $\approx -1.995 \times 10^{-5} s$. Because the motor is very sensitive with respect to the inputs, this small offset has a significant positive influence on the simulation results.

Qualitatively, it can be stated that the neural FMU trajectory does not have any significant, obvious deviation compared to the data trajectory. This does not apply to the original FMU that has a large deviation, especially near the end of the trajectory. This can be seen in Fig. 10 starting at around $18.3\,s$, which shows the deviation in more detail.

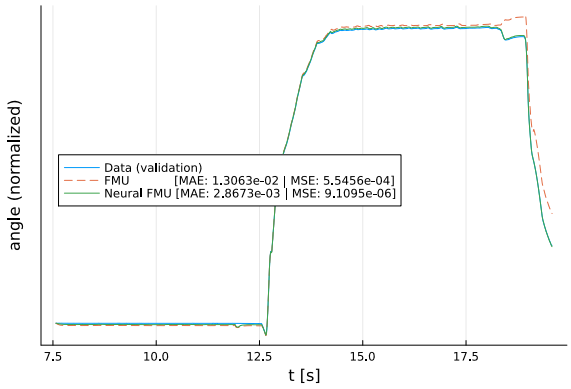


Figure 9. The motor angle on the validation data on the full trajectory. Loss values are given in square brackets.

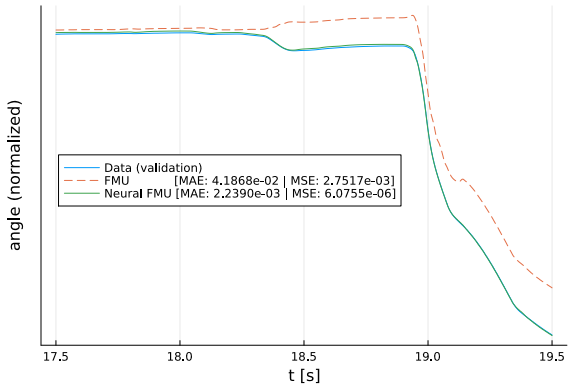


Figure 10. The motor angle on the validation data, zoomed in to the time interval from $17.5\,s$ to $19.5\,s$. The loss values for this interval are given in square brackets.

On a quantitative level, the deviation for the motor angle for the neural FMU is ≈ 4.6 times smaller compared to the FMU on MAE, and even around 61 times on MSE. At this point, we need to highlight that these factors are very impressive from an engineering point of view and deliver massive added value for the (hybrid) simulation model.

As expected from the good results for the motor angle, the neural FMU also provides a qualitatively good fit on the motor speed, see Fig. 11.

Again, we investigate a part of the speed trajectory in more detail. What is particularly striking are the deviating spikes of the FMU, which can be investigated in more detail in Fig. 12. The spikes result from the fact that the model is very sensitive with respect to the control signals and small deviations between the control signal and the position of the rotor lead to unstable behavior in the open-loop control system. Although the original simulation model (FMU) consistently shows these oscillations that even lead to overshoots in the event of rapid changes in speed (acceleration), the neural FMU is able to compensate for the unintended oscillations and provides a nice

DOI
10.3384/ecp218575

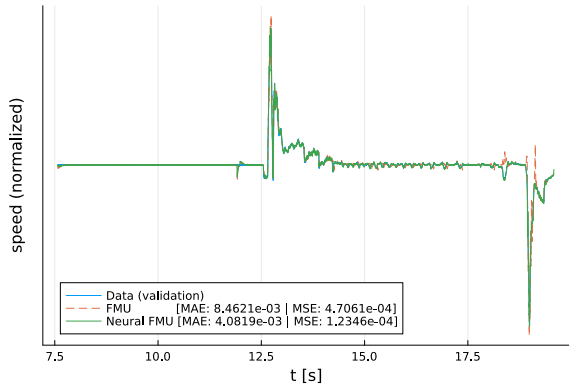


Figure 11. The motor speed on the validation data on the full trajectory. Loss values are given in square brackets.

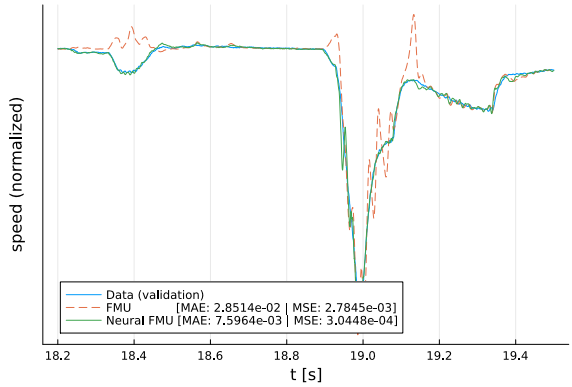


Figure 12. The motor speed on validation data, highlighting the interval between 18.2 s and 19.5 s. Loss values are given in square brackets.

and smooth fit without significant deviations.

4.2 Testing

For proper validation of the hybrid model, the neural FMU is further investigated under testing data, that is completely unknown from training as well as hyperparameter optimization. As for validation data, the neural FMU qualitatively performs very well on testing data (s. Fig. 13), and is even able to provide a little better increase in precision of the predicted rotor angle of factor ≈ 5.4 on MAE or factor ≈ 74 on MSE, respectively.

In contrast to the validation, we want to inspect a segment with raising edge for the motor angle in more detail, see Fig. 14. Within this segment, the original model (FMU) already performs very well, and we want to investigate if the neural FMU is able to retain the parts of the original model that already perform well. Even if the original model has high accuracy in the investigated time span, the neural FMU is able to outperform this performance by a factor of around 2.2 (MAE).

Finally, we also investigate the motor speed on testing data in the following. When investigating the rotor speed on testing data (s. 15), we find quantitative improvements similar to the validation scenario of factor > 2 (MAE).

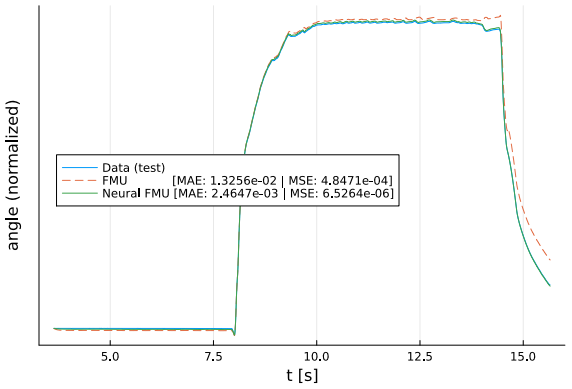


Figure 13. The motor angle on test data on the full trajectory.

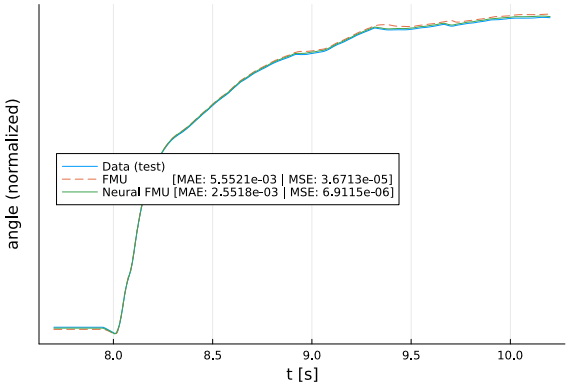


Figure 14. The motor angle on the test data, between 7.8 s and 10.2 s. Loss values are given in square brackets.

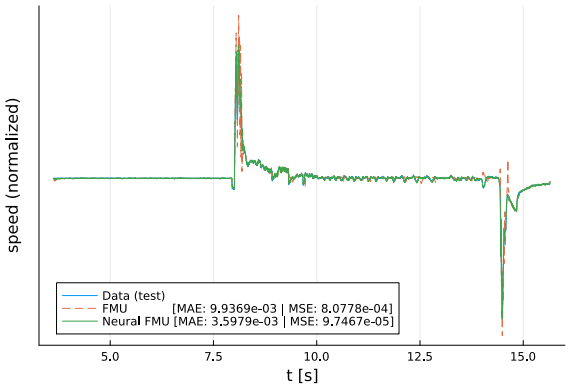


Figure 15. The motor speed on the test data on the full trajectory. Loss values are given in square brackets.

Again, more significantly, the qualitative fit has drastically improved in terms of reduced (wrong) oscillations in the open-loop simulated motor, see Fig. 16.

5 Conclusion

We can summarize that hybrid modeling has the potential to be used in challenging real industrial applications. In this particular use case of the automotive braking system, we were able to increase the accuracy by multiple factors, even though the physical model was based on a

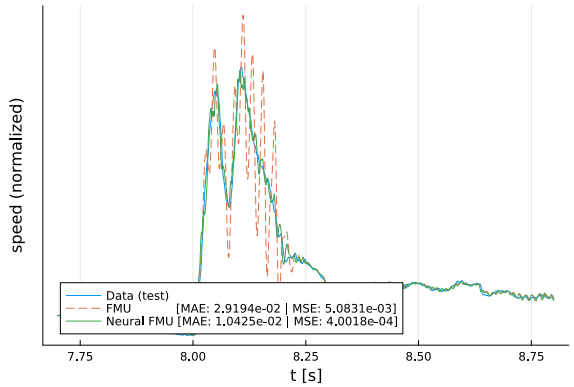


Figure 16. The motor speed on test data, highlighting the interval between 7.75 *s* and 8.75 *s*. Loss values are given in square brackets.

simulation model from real state-of-the-art engineering, built and parameterized by experts. Nevertheless, the successful results presented are still based on well-founded decisions by the development engineer, e.g. the choice of a model architecture that meets the requirements of the use case. We also see this as the task of the modern simulation engineer in the future. Although such decisions can in principle also be optimized by ML approaches, the approach presented here impresses with its very low computing power requirements compared to large-scale optimizations. In this use case, even the in general expensive hyperparameter optimization was executed on a standard desktop CPU with a runtime of only around 48 hours.

For the considered use case, we used data from the (partly) available real prototype to improve an existing simulation model. This approach is of course not applicable, e.g. during early phases of the development, when a real demonstrator is not available. However, the proposed method can also be applied based on synthetic data from another simulation model, e.g. to create a hybrid surrogate model with improved properties (e.g. computational performance or memory requirements) compared to the original model.

Future work will focus, for example, on training and testing based on a larger data set. In this application, in particular, a relatively small data set (one maneuver each for training, testing, and validation) was used to demonstrate the applicability of the method even with only small amounts of data, which is common for many industrial application fields — especially during protoyping. Further, we look at ways to further automate parts of development (e.g. training and batching strategies) to further reduce the development time of such hybrid models.

Abbreviations

ANN arifical neural network

CS co-simulation

FMI Functional Mock-up Interface

FMU functional mock-up unit

IPB Integrated Power Brake

MAE mean absolute error

ME model-exchange

MSE mean squared error

ODE ordinary differential equation

PeN physics-enhanced neural

PWM pulse-width modulation

UODE universal ordinary differential equation

Funding

This research was partially funded by the following sources:

- ITEA3-Project UPSIM (Unleash Potentials in Simulation) No. 19006. For more information, see: `https://www.upsim-project.eu/` (accessed on 12.12.2024).
- ITEA4-Project OpenSCALING (Open standards for SCALable virtual engineerING and operation) No. 22013. For more information, see: `https://openscaling.org/` (accessed on 12.12.2024).

The authors thank all those involved for their support in making our research possible.

Availability of Software, Data and Model

The software used *FMIFlux.jl*[4] is open source on GitHub. However, the simulation model and the data presented are not available to the public due to intellectual property. All data shown was normalized. However, *FMIFlux.jl* offers demo applications that are similar to the workflow presented, and application-specific features, such as learning of time shifts, are available as part of the open source repository.

Author Contributions

Conceptualization, T.T., J.F., S.D., M.L.; methodology, T.T., M.L.; software, T.T.; validation, T.T., S.D.; first-principle model: S.D.; writing, T.T., S.D., J.F., M.L.; visualization, T.T., S.D.; All authors have read and agreed to the published version of the manuscript.

[4] `https://github.com/ThummeTo/FMIFlux.jl`

References

Bertsch, Christian (2022). *FMI 3.0 – The next generation exchange format for system simulation beyond tool borders*. Tech. rep. Robert Bosch GmbH. URL: https://www.bosch.com/stories/fmi-3-0-the-next-generation-exchange-format-for-system-simulation-beyond-tool-borders/.

Chen, Ricky T. Q. et al. (2018). "Neural Ordinary Differential Equations". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf.

Chen, Ricky TQ, Brandon Amos, and Maximilian Nickel (2020). "Learning neural event functions for ordinary differential equations". In: *arXiv preprint arXiv:2011.03902*.

Hofmann, Andreas and Lars Mikelsons (2025). "Towards Integration of PeN-ODEs in a Modelica-based workflow". In: *16th International Modelica & FMI Conference, Lucerne, Switzerland, Sep 8-10, 2025*. (submitted).

Kingma, Diederik P (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.

Li, Lisha et al. (2018). "Hyperband: A novel bandit-based approach to hyperparameter optimization". In: *Journal of Machine Learning Research* 18.185, pp. 1–52.

Modelica Association (2020-12). *Functional Mock-up Interface for Model Exchange and Co-Simulation. Document version: 2.0.2*. Tech. rep. Linköping: Modelica Association. URL: https://github.com/modelica/fmi-standard/releases/download/v2.0.2/FMI-Specification-2.0.2.pdf.

Modelica Association (2023-07). *Functional Mock-up Interface Specification. Version 3.0.1*. Tech. rep. Modelica Association. URL: https://fmi-standard.org/docs/3.0.1/.

Rackauckas, Christopher et al. (2021). *Universal Differential Equations for Scientific Machine Learning*. arXiv: 2001.04385 [cs.LG].

Ramadhan, Ali et al. (2023). *Capturing missing physics in climate model parameterizations using neural differential equations*. arXiv: 2010.12559 [physics.ao-ph]. URL: https://arxiv.org/abs/2010.12559.

Robert Bosch GmbH, ed. (2024). *Kraftfahrtechnisches Taschenbuch*. Wiesbaden: Springer Vieweg. ISBN: 978-3-658-44233-0. DOI: 10.1007/978-3-658-44233-0. URL: https://link.springer.com/book/10.1007/978-3-658-44233-0.

Robert Bosch GmbH (2025). *Bosch Integrated Power Brake*. https://www.bosch-mobility.com/en/solutions/driving-safety/integrated-power-brake/ [Accessed: 04-03-2025].

Sorourifar, Farshud et al. (2023). "Physics-enhanced neural ordinary differential equations: application to industrial chemical reaction systems". In: *Industrial & Engineering Chemistry Research* 62.38, pp. 15563–15577.

Tac, Vahidullah, Francisco Sahli Costabal, and Adrian B. Tepole (2022). "Data-driven tissue mechanics with polyconvex neural ordinary differential equations". In: *Computer Methods in Applied Mechanics and Engineering* 398, p. 115248. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2022.115248. URL: https://www.sciencedirect.com/science/article/pii/S0045782522003838.

Thummerer, Tobias, Artem Kolesnikov, et al. (2023). "Paving the way for hybrid twins using neural functional mock-up units". In: *Modelica Conferences*, pp. 141–150.

Thummerer, Tobias and Lars Mikelsons (2025). *Learnable & Interpretable Model Combination in Dynamical Systems Modeling*. arXiv: 2406.08093 [cs.LG]. URL: https://arxiv.org/abs/2406.08093.

Thummerer, Tobias, Lars Mikelsons, and Josef Kircher (2021). "NeuralFMU: towards structural integration of FMUs into neural networks". In: *Proceedings of 14th Modelica Conference 2021, Linköping, Sweden, September 20-24, 2021*. Ed. by Martin Sjölund et al. ISBN: 978-91-7929-027-6. DOI: 10.3384/ecp21181297.

Thummerer, Tobias, Johannes Stoljar, and Lars Mikelsons (2022). "NeuralFMU: Presenting a Workflow for Integrating Hybrid NeuralODEs into Real-World Applications". In: *Electronics* 11.19. ISSN: 2079-9292. DOI: 10.3390/electronics11193202. URL: https://www.mdpi.com/2079-9292/11/19/3202.

Thummerer, Tobias, Johannes Tintenherr, and Lars Mikelsons (2021-11). "Hybrid modeling of the human cardiovascular system using NeuralFMUs". In: *Journal of Physics: Conference Series* 2090.1, p. 012155. DOI: 10.1088/1742-6596/2090/1/012155. URL: https://dx.doi.org/10.1088/1742-6596/2090/1/012155.

Xie, Xiang, Ajith Kumar Parlikad, and Ramprakash Srinivasan Puri (2019). "A Neural Ordinary Differential Equations Based Approach for Demand Forecasting within Power Grid Digital Twins". In: *2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pp. 1–6. DOI: 10.1109/SmartGridComm.2019.8909789.

DOI
10.3384/ecp218575

VDCWorkbench: A Vehicle Dynamics Control Test & Evaluation Library for Model and AI-based Control Approaches

Jonathan Brembeck[1] Ricardo Pinto de Castro[2]
Johannes Ultsch[1] Jakub Tobolar[1]
Christoph Winter[1] Kenan Ahmic[1]

[1]Department of Vehicle System Dynamics and Control, DLR Institute of Vehicle Concepts, Germany, {first.last}@dlr.de

[2] Dept. of Mechanical Engineering, University of California Merced, USA , rpintodecastro@ucmerced.edu

Abstract

The ability to systematically compare and evaluate diverse control strategies is essential for the development of effective control algorithms in autonomous driving. This contribution presents the VDCWorkbench Modelica Library, a unified platform designed to support the development, testing, validation and verification of vehicle dynamics controllers and energy management strategies. The presented library is an extension of the IEEE VTS Motor Vehicle Challenge 2023 models and offers multi-physical component modeling, including a hybrid energy storage system (fuel cell & hydrogen tank and battery with aging model), as well as vehicle dynamics control for autonomous driving research projects. Two path-following approaches are featured: an open-loop lateral controller with a static inversion of a single-track model, and a closed-loop state-dependent geometric path-following controller with static control allocation. The library may also serve as the foundation for development of vehicle control methods, such as two-degree-of-freedom control approaches concepts. One example is given for this combination of a feedforward controller with residual reinforcement learning, where a learned agent improves the performance of the open loop controller. The entire library will be released as open source on GitHub in September 2025.

Keywords: Vehicle Dynamics Control, Lateral Control, Trajectory Following, Reinforcement Learning, Inverse Model, Energy Management, Vehicle Dynamics Simulation, Open Source, Powertrain Simulation, Electric Vehicle, Functional Mockup Interface

1 Introduction

To accelerate the development of control algorithms in the field of vehicle dynamics, efficient modeling tool chains are required to capture key physical effects with reduced computational complexity. In this work, we present the Vehicle Dynamics Control Workbench Library (*VDCWorkbench*), a Modelica-based library that can be used as a workbench for the design, testing and validation of vehicle dynamics controllers and energy management algorithms of electric vehicles. This open-source library gives students, researchers and engineers the opportunity to familiarize themselves with vehicle dynamics modeling and control. The toolbox can be used, for example, for autonomous vehicle projects and assessment of different vehicle configurations.

Other Modelica-based simulation libraries for electric and autonomous vehicles are typically commercial libraries such as Modelon's Vehicle Dynamics, DLRs Powertrain and Alternative Vehicle Library and Claytex's VeSyMA offerings. Outside Modelica, open-source platforms like Gazebo (robotics simulation, can be adapted for vehicle control), CARLA (open-source simulator focusing on automated driving scenarios), Bullet Physics (general rigid and soft body physics, can be used for vehicle dynamics) provide multibody physics, sensor modeling, and scenario generation for vehicle dynamics and control [1]. Most of these previous simulation packages lack support for comprehensive multi-physics modeling of the entire vehicle system and do not allow exporting Functional Mock-up Units (FMUs) for training AI-based controllers—capabilities that are uniquely provided by our *VDCWorkbench* Library. Moreover, tools such as CARLA emphasize high-fidelity scenario visualization for generating synthetic perception data, rather than integrated vehicle-level control and energy management modeling, which are supported by the *VDCWorkbench* library. Note that advanced visualization capabilities can also be provided by the *VDCWorkbench* via the DLR Visualization Library.

A preliminary version of the *VDCWorkbench* library was used as the simulation engine for the IEEE VTS Motor Vehicle Challenge 2023 [2], a research competition for energy management strategies of hybrid electric vehicles.

This library has not yet been published and was only made available to the participants of the challenge as a black box FMU for the development of their energy management algorithms. The entire *VDCWorkbench* library is released as open-source software in the GITHUB repository of the Vehicle System Dynamics and Control department: DLR-VSDC/VDCWorkbench. The library is tested with Dymola 2025x Refresh 1 (64-bit Windows) and also works in OpenModelica v1.25.0 (64-bit Windows) to increase the potential user community for academic projects.

This publication is structured as follows. First, we will discuss the structure of the library with its control modules and the multi-physical modeling of the components. We then describe two concepts for the integration of vehicle dynamics control following a precomputed time-independent parametric path.

2 VDC Workbench Structure

This chapter presents the structure of the *VDCWorkbench* library. Figure 1 shows the schematic structure of the library which was derived and extended from the IEEE VTS Motor Vehicle Challenge 2023 [2]. The library has four types of components: i) vehicle dynamic and actuator models; ii) energy storage models; iii) vehicle control algorithms and iv) energy management algorithms (EMA).

The reference vehicle for the library is an adapted version of the DLR ROboMObil ([3] [4] [5] [6]), which is configured with an simplified drivetrain and a hybrid energy storage system (fuel cell & hydrogen tank and battery). The vehicle has two in-wheel electric motors installed in the rear axle (τ_{RL}, τ_{RR}), one central front motor (τ_F), and a front steer-by-wire actuation (δ). Within the vehicle control block different vehicle dynamics control algorithms are implemented as descripted later in Chapter 3. Some of these vehicle control algorithms can enhance energy efficiency of the vehicle by enabling energy recuperation in both front and rear axle [7] and also decreasing tire slip losses [8]. The in-wheel motors can improve motion control of the vehicle by extending the maximum lateral acceleration [9] and decreases response of inner (yaw-rate) control loops [2]. Additionally, the energy management algorithm (EMA), as developed in the IEEE VTS Motor Challange 2023 [2], sets the control variables for optimizing the energy consumption of the powertrain.

These normalized control variables of the EMA have the following meaning

- $\alpha_{AD} \in [0,1]$ the axle torque distribution ratio between front ($\alpha_{AD} = 0$) and rear axle ($\alpha_{AD} = 1$)
- $\alpha_{TV} \in [0,1]$ the torque vectoring ratio between rear left ($\alpha_{TV} = 0$) and rear right motors ($\alpha_{TV} = 1$)
- $\alpha_v \in [0,1]$ velocity derating factor,
- $\alpha_{FC} \in [0,1]$ the normalized fuel cell current,

For further details on the EMA, please refer to [2]. Several EMA strategies have been developed for this vehicle configuration, ranging from rule-based algorithms, machine learning and optimal energy management (see Ranking of the competition).

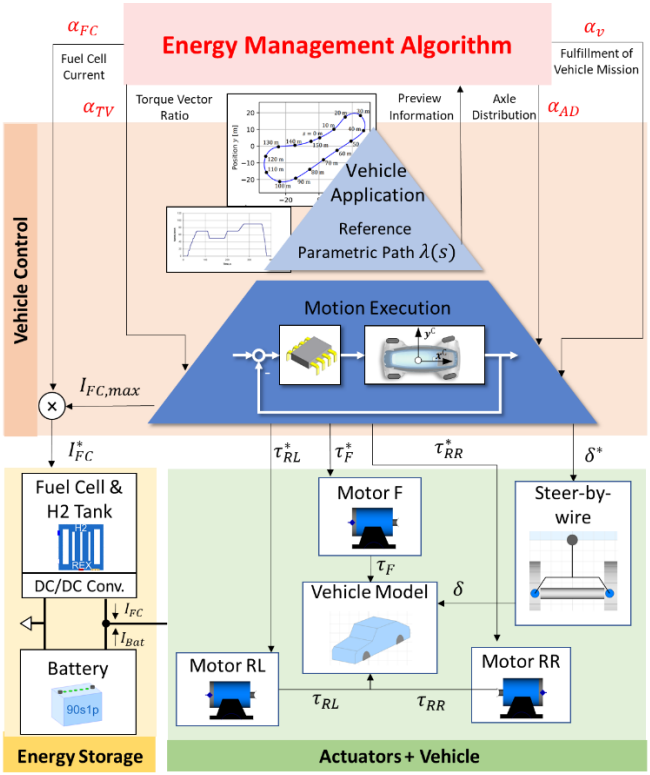


Figure 1. Schematic Structure of the VDCWorkbench Library

2.1 Overall Library Setup and Usage

Figure 2 provides an overview of the sub-packages available in the *VDCWorkbench*. In the `VehicleComponents` package, there is a sub-package for the mechanical structure of the `chassis` (Vehicle Model & Steer-by-Wire Model in Figure 1) based on the PlanarMechanics Library [10] [11].

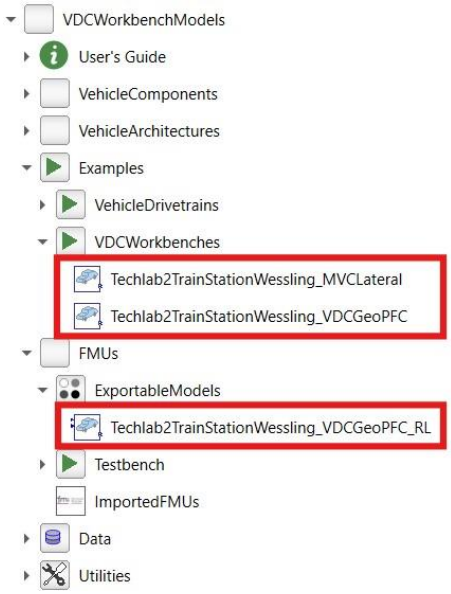


Figure 2. Architecture of the VDCWorkbench Library

The `Powertrain` sub-package contains firstly a model of a quasi-stationary Permanent Magnet Synchronous Machine (PMSM) with an approximated control in the d/q coordinate system. Secondly, a table-based first-order (RC) equivalent circuit battery model with a cell aging model that models the capacity loss over the cell lifetime as a function of the number of charging and discharging cycles, current load and temperature. The third component is a functional quasi-stationary model of a fuel cell with tank. In addition to the vehicle dynamic controllers, which are discussed in detail in Chapter 3, the last sub-package `Controllers` also contains a base class for energy management and the implementation of the baseline energy management strategy as described in Chapter IV.D in [2].

In the sub-package `Examples.VehicleDrivetrains` there are various test configurations to configure the component models with each other to create a multi-physical vehicle and thus build your own vehicle architecture. The examples contained here in `VariantsVehicleDrivetrains` range from a simple vehicle with only one centrally arranged PMSM with differential to a vehicle with four in-wheel hub motors. To test these vehicle configurations, various driving maneuvers are provided. The following two sub-packages `Examples.VDCWorkbenches` and `FMUs` contain complete vehicle models with the baseline energy management algorithm. In addition, there are driving dynamics controllers including test scenarios for experimental investigation or for export to AI training environments. The sub-package `Data` includes all the parameterizations used for the components and controllers of our simplified RObOMObil vehicle model [3]. This is intended to ensure that the models are consistently parameterized and represent a blueprint for your own vehicle configurations. In addition to the bus definitions, which are based on the VehicleInterfaces Library, the last sub-package `Utilities` contains various auxiliary models, such as a sensor model for determining the vehicle dynamics variables required for the controllers.

In order to use our library and simulate the examples, the user must clone the GIT repositories VehicleInterfaces (tested release V2.0.1) and PlanarMechanics (tested release V1.6.0) locally and load them in the Modelica simulator.

2.2 Vehicle Workbench Configuration

In this section, we provide more details about the main model of the *VDCWorkbench* library. The structure of the overall simulation model with the individual building blocks, as shown in Figure 1, is implemented in the model `VDCWorkbench2025_Model` of the sub-package `Examples.VDCWorkbenches` (see Figure 3). The color-coded boxes corresponding to the subcomponents in the schematic sketch. All data relevant for the simulation is stored in the record `data`. The controller block is implemented as replaceable, which we will instantiate, explain and examine in Chapter 3 with various vehicle controllers. The `vehicle`, `drivetrain`, `powerSource` and `baselineEMA` blocks are each taken from the parameterized models of the `VehicleComponent` sub-package.

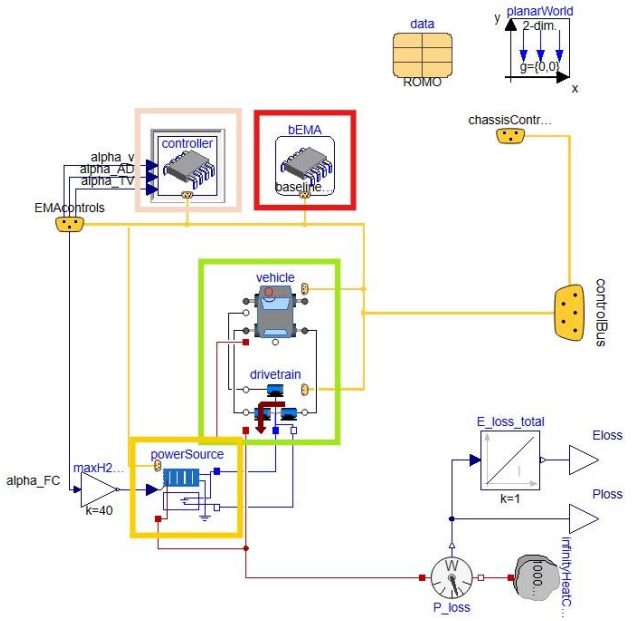


Figure 3. Structure of vehicle components and controllers

2.3 Test Scenario Generation

In order to be able to test our driving dynamics controllers in different scenarios, it is necessary to provide the `controller` block with a parametric path description of the test scenarios. This is defined as a time-independent vehicle movement request along a flat geometric path. For this purpose, we have developed an interactive path planning tool with which parametric paths, subject to constraints, can be generated. This is not planned to be part of the *VDCWorkbench* repository but we are open for inquiries, both for commercial and academic use. Based on an interface to Google Maps (expendable to other map systems such as Here MAPS or Open Street Maps), the reference vehicle path can be created via point-and-click and numerically optimized within the lane boundaries. An example of this is shown here for a section of a path around a roundabout on the route from the DLR site in Oberpfaffenhofen to the next railroad station in Wessling.

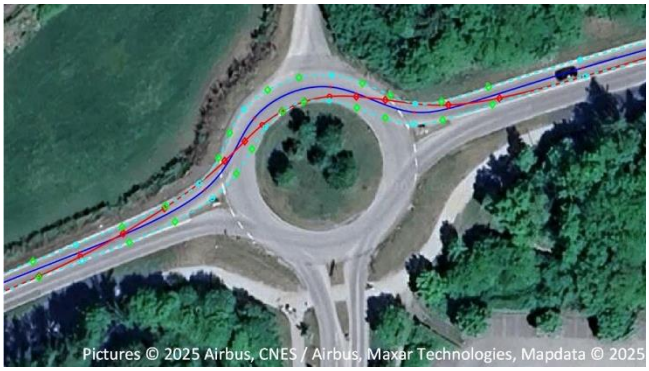


Figure 4. Interactive path planning GUI with boarders (green), unoptimized path (blue) and optimized path (red)

As can be seen in Figure 4, the geometric path is described as a two-dimensional polynomial (blue) with its interpolation points (small red dots), that are calculated under boundary conditions for the start and end points. Additionally, an optimization of the path (red) based on energy-optimal criteria with constraints, such as compliance with the road boundary (green), is given [12]. All variables are stored in the path description function $\boldsymbol{\lambda}(s) \in \mathbb{R}^5$, which depends on the path arc length s.

3 Vehicle Dynamics Control

The library also includes two baseline vehicle dynamics controllers designed to regulate both the lateral path and longitudinal velocity of a vehicle. These controllers serve as reference implementations that can be used out of the box or extended with more advanced control strategies, including machine learning methods.

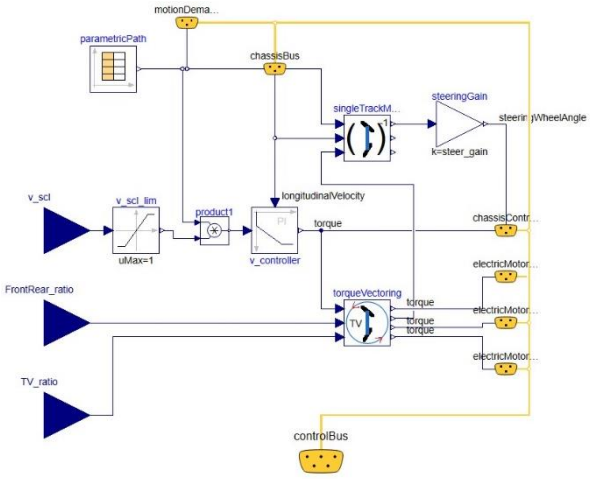


Figure 5. Baseline lateral and speed controller

The first controller is an open-loop lateral controller based on the static inversion of a single-track vehicle model (cf. Figure 5). It calculates the required steering angle to follow a predefined path using the kinematic properties of the vehicle. This is combined with a PI (Proportional-Integral) speed controller, referred to as `v_controller`, which regulates the longitudinal velocity of the vehicle. This modular structure allows the user to separately tune lateral tracking and velocity control loops. Further details can be found in [2].

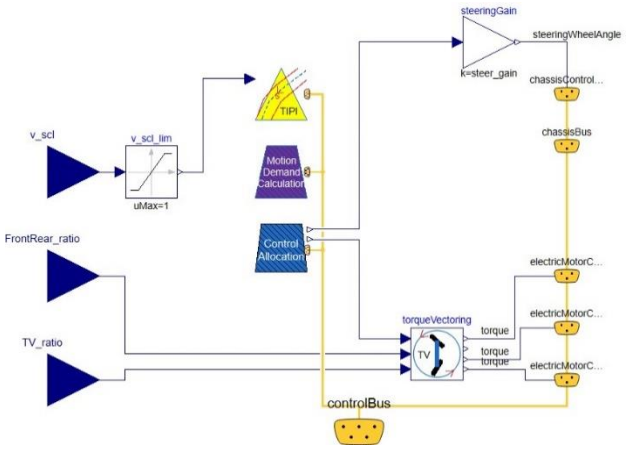


Figure 6. Geometric path following control

The second controller implements a closed-loop geometric path-following strategy (GeoPFC, cf. Figure 6). It relies on a state-dependent computation of the control input, which is then mapped to physical actuator commands through an optimization-based control allocation procedure. This approach allows the controller to adjust the vehicle's motion, depending on its current state with respect to a reference path (e.g., lateral deviation, heading error). For more information, refer to [5] [13].

Example: In Figure 7 we show the lateral and longitudinal deviation of the close-loop controller for our test scenario from Chapter 2.3. While the longitudinal error is minimized very well, particularly in dynamic maneuvers such as the roundabout in the section $s = [900 - 1200]$m there are larger deviations in the lateral error. These can be explained by the very simple control allocation approach, which does not take vehicle and actuator dynamics into account, and the pure front axle steering.

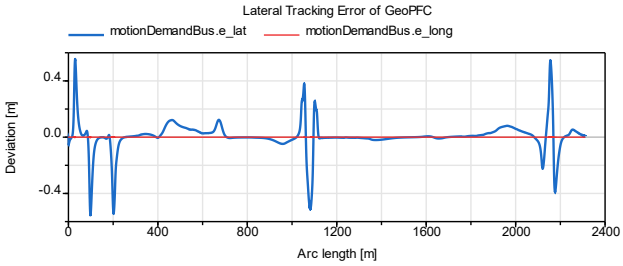


Figure 7. Deviation of controller tracking error

Figure 8 shows the results of the `Control Allocation` and `TorqueVectoring`, including the torque allocated to the traction motors and the steering angle.

DOI 10.3384/ecp218585

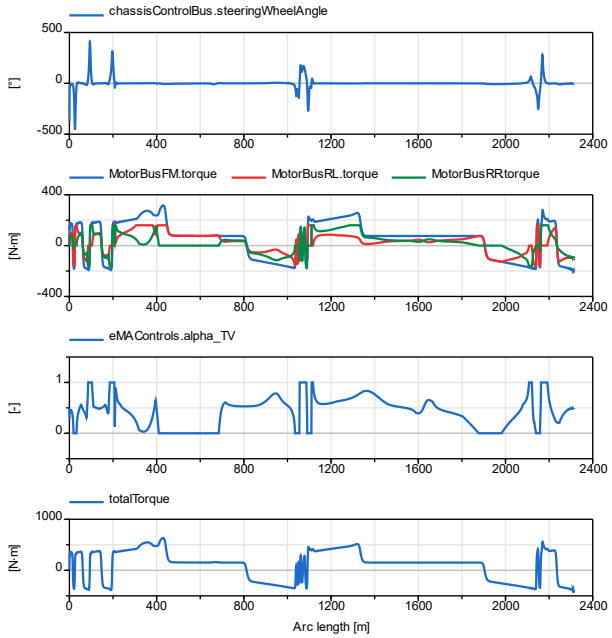


Figure 8. GeoPFC controller actuator demand values

These baseline controllers are designed to be compatible with residual reinforcement learning (rRL) frameworks. In this paradigm, the baseline controller is augmented with a learned agent that contributes an additional control input (red boxes in Figure 9). This residual term allows for performance improvement while preserving the interpretability and domain knowledge of the model-based controller [14] [15].

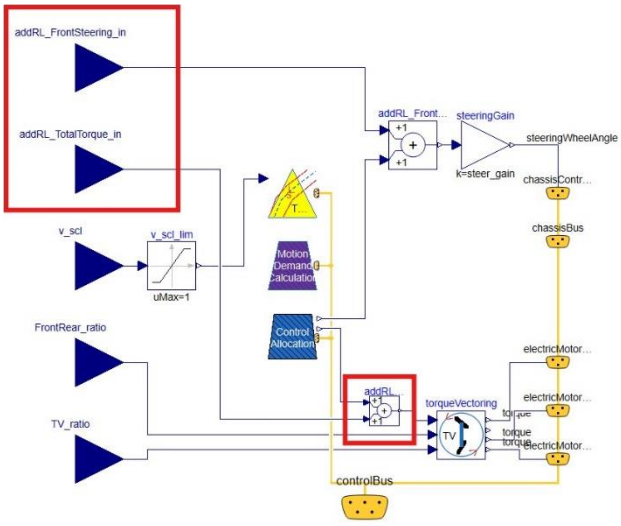


Figure 9. GeoPFC with additional RL agent inputs

4 Summary & Outlook

This work introduces a new Modelica library (*VDCWorkbench*) for simulation of vehicle dynamics and control. It supports developing, testing, and validating control algorithms in hybrid electric vehicles based on the DLR ROboMObil prototype vehicle. The library offers both open-loop and closed-loop vehicle dynamics controllers along a parametric path,. Energy management functions that make use of the multi-physical modeling approach can enhance the control strategy and optimize vehicle energy efficiency. A baseline policy EMA is included, and more sophisticated strategies can be added. *VDCWorkbench* library will be released open source in the DLR VSDC GitHub repository by September 2025, making it broadly accessible for all kind of research applications. It is planned to add further controllers, both for vehicle dynamics and energy management, as well as vehicle models to the library and thus provide the community with further application examples.

Acknowledgements

This research was supported by the DLR project Validation and Verification for the Next Generation Vehicle (V&V4NGC).

References

[1] Y. Li, W. Yuan, S. Zhang, W. Yan, Q. Shen, C. Wang and M. Yang, "Choose Your Simulator Wisely: A Review on Open-Source Simulators for Autonomous Driving," *IEEE Transactions on Intelligent Vehicles*, vol. 9, pp. 4861-4876, May 2024.

[2] J. Brembeck, R. De Castro, J. Tobolář and I. Ebrahimi, "IEEE VTS Motor Vehicles Challenge 2023: A Multi-physical Benchmark Problem for Next Generation Energy Management Algorithms," in *2022 IEEE Vehicle Power and Propulsion Conference (VPPC)*, 2022.

[3] J. Brembeck, L. M. Ho, A. Schaub, C. Satzger, J. Tobolar, J. Bals and G. Hirzinger, "ROMO - The Robotic Electric Vehicle," in *22nd IAVSD International Symposium on Dynamics of Vehicle on Roads and Tracks*, Manchester Metropolitan University, 2011.

[4] J. Brembeck and P. Ritzer, "Energy optimal control of an over actuated Robotic Electric Vehicle using enhanced control allocation approaches," in *Proceedings of the 2012 IEEE Intelligent Vehicles Symposium*, Alcala de Henares, Spain, 2012.

[5] J. Brembeck, "Model Based Energy Management and State Estimation for the Robotic Electric Vehicle ROboMObil," Ph.D. dissertation, Technical University of Munich, Munich, Germany, 2018.

[6] DLR Vehicle System Dynamics Blog, "10 years ROboMObil," 08 2022. [Online]. Available: https://vsdc.de/en/robomobil-timeline/.

[7] R. de Castro, M. Tanelli, R. E. Araújo and S. M. Savaresi, "Design of safety-oriented control allocation strategies for overactuated electric vehicles," *Vehicle System Dynamics*, vol. 52, pp. 1017-1046, 2014.

[8] J. Torinsson, M. Jonasson, D. Yang and B. Jacobson, "Energy reduction by power loss minimisation through wheel torque allocation in electric vehicles: a simulation-based approach," *Vehicle System Dynamics*, vol. 60, pp. 1488-1511, 2022.

[9] B. Lenzo, "Torque Vectoring Control for Enhancing Vehicle Safety and Energy Efficiency," in *Vehicle Dynamics: Fundamentals and Ultimate Trends*, B. Lenzo, Ed., Cham, Springer International Publishing, 2022, p. 193–233.

[10] D. Zimmer, "A Planar Mechanical Library for Teaching Modelica," in *Proceedings of the 9th International Modelica Conference*, Munich, Germany, 2012.

[11] D. Zimmer and M. Otter, "Real-time models for wheels and tyres in an object-oriented modelling," *Vehicle System Dynamics*, 2009.

[12] C. Winter, R. de Castro and T. Bünte, "Experimental Validation of Online Motion Planning for Semi-Autonomous Vehicles," in *2022 IEEE Vehicle Power and Propulsion Conference, VPPC 2022*, 2022.

[13] P. Ritzer, C. Winter and J. Brembeck, "Experimental Validation of Geometric Path Following Control with Demand Supervision on an Over-Actuated Robotic Vehicle," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, Gothenburg, Sweden, 2016.

[14] J. Ultsch, K. Ahmic and J. Brembeck, "train_fmu_gym: A Functional Mock-up Unit-based Framework to Train Reinforcement Learning Agents for Multi-Physical Systems," *Manuscript submitted for publication*, 2025.

[15] J. Ultsch, J. Brembeck and R. de Castro, "Learning-Based Path Following Control for an Over-Actuated Robotic Vehicle," in *VDI-AUTOREG*, Düsseldorf, 2019.

 DOI 10.3384/ecp218585

Simulation of Embodied Cyber Physical System Based on Modelica/MWORKS: A Case Study of Intelligent Unmanned Surface Vessel

Zhiguo Zhou[1,4,5]; Xuehua Zhou[1]; Lin DU[2]; Peiquan Ma[1]; Xiang Wang[1]; Ying chen[1]; Mingjia Liu[1]; Tengyue Wang[1]; Lixin Hui[3]; Cun Zeng[3]

[1]School of Integrated Circuits and Electronics, Beijing Institute of Technology, China, {zhiguozhou, xuehuazhou, 3220242141, 3220231862, 3120221304,3220242100}@bit.edu.cn

[2]Maritime and Transportation Department, Ningbo University, China, dulin1@nbu.edu.cn

[3]Suzhou Tongyuan Soft Control Information Technology Co., Ltd, China, {huilx,zengc}@tongyuan.cc

[4]Beijing Institute of Technology, Zhuhai 519088 No. 6 Jinfeng Road, Tangjiawan Town, Xiangzhou District, Zhuhai City, Guangdong Province, China

[5]Tangshan Research Institute of BIT, Tangshan 063000,China

Abstract

This paper proposes a new paradigm of the Embodied Cyber-Physical System (Embodied CPS, ECPS) to address the issues of the disconnection between physical laws and intelligent decision-making and the insufficient interaction with dynamic environments in the modeling and simulation of traditional CPS. ECPS achieves unified modeling of physical laws and autonomous decision-making through the "perception-decision-action" closed loop. To verify ECPS, an embodied space framework based on Modelica/MWorks is designed. Through three major technological innovations: constructing an embodied domain modeling specification and embedding the Navier-Stokes equations into the training of the policy network; expanding the syntax and semantics of Modelica, encapsulating physical constraint reinforcement learning components, and establishing a gradient interaction protocol between the Physics-Informed Neural Network (PINN) and Modelica equations; building a digital twin-hardware-in-the-loop co-simulation platform based on the FMI/SSP protocol to establish a collaborative verification link between high-precision physical simulation and real-time decision-making. Taking the Unmanned Surface Vehicle (USV) as the carrier, the full-process method from dynamic modeling, reinforcement learning strategy training to virtual-real environment co-simulation is demonstrated. Experiments verify the effectiveness of this framework in achieving the closed-loop coupling of physical simulation and intelligent decision-making under complex sea conditions, providing a methodological foundation for interpretable modeling and verifiable simulation in the development of embodied intelligence.

Keywords: embodied intelligence, embodied domain, embodied spatial, physical information neural network

1 Introduction

1.1 Background

With the increasing demand for the adaptability of autonomous systems in dynamic physical environments, Cyber-Physical Systems (CPS) [1] are gradually evolving towards Embodied Cyber-Physical Systems (Embodied CPS, ECPS). Its core feature lies in that intelligent agents need to conduct real-time interactions with the physical environment through the "perception-decision-action" closed loop [2]. In the traditional CPS framework, physical models and intelligent algorithms are often designed in a segmented way, resulting in bottlenecks such as the lack of compliance with physical laws and insufficient dynamic interaction modeling when intelligent agents are deployed in real environments. For example, in scenarios where dynamic physical constraints (such as the torque limitations of robotic arm joints and the dynamic boundaries of mobile robots) coexist with high-dimensional environmental uncertainties (such as the random distribution of obstacles and multimodal disturbances), existing methods find it difficult to achieve the deep integration of physical laws, environmental interactions, and autonomous decision-making. How to construct a simulation system with the ternary coupling of "physics-information-environment" [3], and organically unify the strong constraints of physical laws with the data-driven environmental adaptability has become the core challenge in the development of embodied intelligence systems.

1.2 Related Work

In terms of ECPS simulation methods, current research presents two typical paths: One is based on the Modelica multi-domain modeling tool (such as MWORKS), which accurately describes the constitutive relations of physical

systems through acausal equations. However, in terms of the deep integration of the intelligent decision-making layer and the deep learning framework, it is still limited by the difficult problem of collaborative solving of discrete-continuous systems. The other is to adopt the Physics-Informed Neural Network (PINN) [4]. By embedding physical laws such as the Navier-Stokes equations into the neural network training process through the residual constraints of differential equations, it demonstrates the advantage of following physical constraints in embodied intelligence scenarios such as fluid mechanics prediction and robotic motion control. It is worth noting that recent research attempts to integrate PINN with Modelica [5], and realizes the co-simulation of the neural network model and the multi-domain physical model through the Functional Mock-up Interface (FMI) standard. However, there is still a lack of standardized protocols in key technical aspects such as the gradient backpropagation mechanism and the alignment of spatio-temporal scales between the two. Overall, the development of current ECPS simulation tools is still in the exploratory stage. Existing research generally faces three major bottlenecks: the lack of a semantic-level integration mechanism between physical equations and data-driven models; the insufficient standardization of the protocol stack for real-time interaction between heterogeneous simulation platforms; and the contradiction between the efficiency and interpretability of the representation of the embodied space in a high-dimensional dynamic environment.

1.3 Challenges

The core challenges of ECPS simulation are as follows:
It is difficult to couple multi domain models: the traditional Modelica standard library lacks a specific agent interface, and cannot uniformly describe the real-time interaction between physical equations (such as ship dynamics) and intelligent algorithms (such as DRL) [6];
Physical constraints are separated from intelligent decision-making: data-driven methods are easy to generate actions that violate physical laws (such as exceeding the safety threshold of roll angle), while traditional control theory is difficult to deal with high-dimensional environmental uncertainty; Lack of virtual and real collaborative verification system: the existing tool chain (such as Gazebo+ROS) [7] has a gap between the physical simulation accuracy (microsecond step size) and the efficiency of intelligent algorithm (millisecond response), and lacks standardized interface protocol.

1.4 Proposed Approach

In response to the above challenges, this paper proposes an ECPS simulation framework based on the embodied space, and achieves methodological breakthroughs through a three-level technical path. With the modeling specification of the embodied domain as the technical foundation, this framework expands the Modelica language to define standardized interaction interfaces (such as sensors/actuators) and physical constraint description grammars, and supports the semantic-level integration of complex physical models such as the six-degree-of-freedom dynamics of ships and reinforcement learning strategies. Relying on the embodied space architecture, a collaborative simulation link of "physical domain (high-precision fluid mechanics equations) - information domain (DRL decision-making) - embodied domain (PINN gradient interaction)" is constructed. The constraints of the Navier-Stokes equations are embedded in the strategy training through the PINN-Modelica protocol to ensure that intelligent decisions strictly follow the physical boundaries. Based on the FMI/SSP protocol, multi-time scale scheduling (1ms-level physical simulation/10ms-level decision-making) is realized. In combination with ROS/Unity, the co-verification of virtual and real environments is achieved, providing a methodological paradigm for the engineering of ECPS in dynamic environments.

2 Methodology

2.1 Modelica Modeling Specification for the Embodied Domain

To meet the needs of intelligent design for modern autonomous systems and address the issues of insufficient intelligence in traditional cyber-physical integration systems and weak theoretical constraints of intelligent algorithms, this paper constructs a three-tier technical system for the Modelica modeling specification of the embodied domain: forming the embodied domain specification based on the extensible syntax and semantics of Modelica; constructing a verification protocol stack for virtual and real systems based on API interfaces, the FMI standard, and the SSP protocol; and building a physical-intelligence integration engine based on the PINN-Modelica gradient interaction protocol. This specification incorporates intelligent considerations on the basis of the physical domain and the information domain to achieve the unification of physical laws and autonomous decision-making. On the one hand, by utilizing the extensible syntax and semantics of Modelica, a general modeling specification and architecture for embodied entities are formed. The model components of different objects are standardized, and generalized models of general autonomous systems such as autonomous vehicles and unmanned surface vehicles are constructed. This provides theoretical constraint boundaries for the embodied data training of various unmanned systems and scenario working conditions, and is verified in real time through the physics-informed neural network, breaking through the collaborative bottleneck between semantic ambiguity and physical compliance. On the other hand, based on the PINN-DRL architecture, physical laws such as the Navier-Stokes equations are embedded in the neural network training.

DOI 10.3384/ecp218591

Through the regularization constraints of physical equations, the model not only fits the data but also follows physical laws, thereby improving generalization ability, data efficiency, and interpretability of time decision-making in dynamic environment.

2.2 Extension Specification for Modelica Embodied Domains

Traditional engineering often faces problems such as poor compatibility of heterogeneous interface protocols, discretization of physical constraint description, and difficulty in cross domain simulation coupling, resulting in low efficiency of agent modeling and limited simulation credibility. Although the Modelica standard library can solve the unified modeling and simulation problems in many fields, such as Electromechanical, hydraulic and thermal control, it lacks standardized support for perception action closed-loop, real-time physical constraints and intelligent strategy fusion, and cannot realize the deep coupling between the agent and the physical environment. It is difficult to cope with the rapid development of embodied intelligence in the fields of robots, unmanned systems and so on.

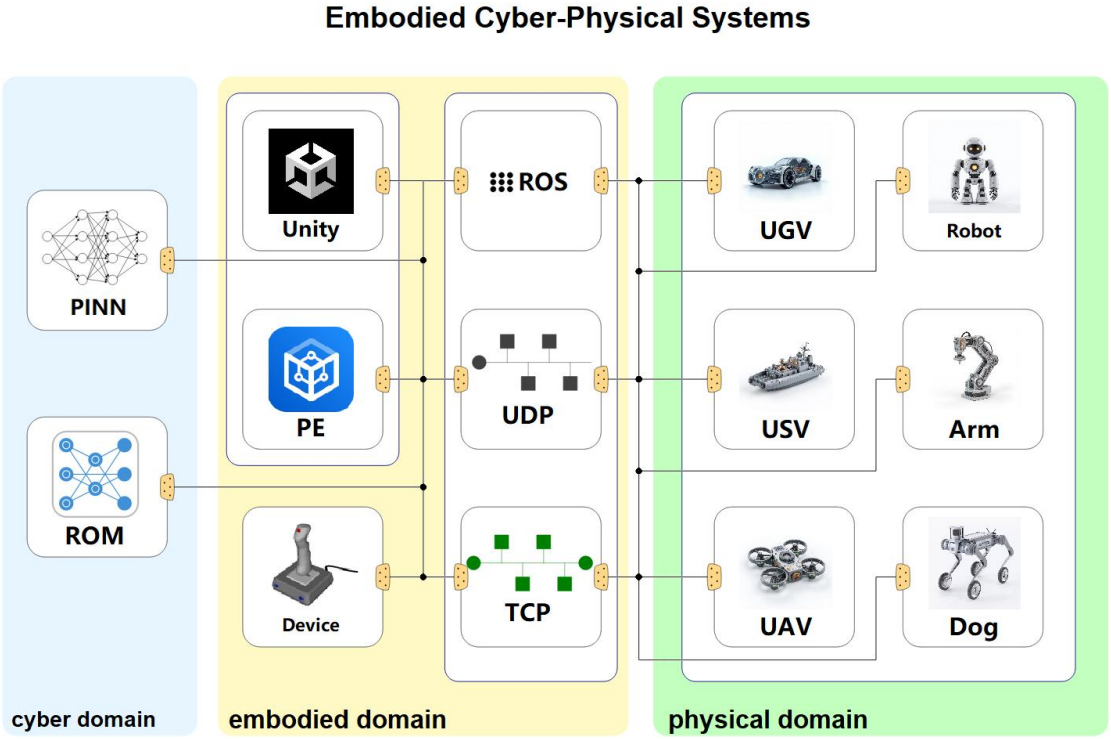


Figure 1. Embodied Cyber-Physical Systems.

The embodied cyber-physical system consists of three parts: the physical domain, the information domain, and the embodied domain. The physical domain includes various embodied intelligent agents, such as autonomous vehicles, unmanned ships, drones, robots, robotic arms, robotic dogs, and so on. Embodied intelligent agents satisfy the concept of Modelica replaceable classes and can switch autonomously according to demand. The information domain includes reinforcement learning strategies and neural network reductions, achieving closed-loop optimization of physical laws and autonomous decision-making. The embodied domain is responsible for the environmental interaction capabilities of the entire embodied space, including communication, 3D visual scenes, external devices, etc. The various parts are connected through the embodiment domain language extension, achieving a technical closed loop of "perception-decision-action".

Therefore, in order to support the formal modeling of embodied agents and improve the modeling ability of complex systems, it has three core functional modules: standardized interaction framework: defining the base class and open interface of embodied agents, unifying the sensor actuator communication standard and reducing the amount of hardware adaptation code. Integration of physical intelligence: innovative introduction of PINN hybrid modeling syntax and physical constraint reinforcement learning library to optimize AI strategy in strict accordance with dynamic laws. Multi domain collaborative support: define Modelica general standard interface based on API interface or communication protocol to realize the coupling simulation of control logic, physical model and environmental factors under a unified time base.

Specifically, this specification extends the following syntax for Modelica language:

- add the embodied keyword to support the declaration of embodied agents.

The traditional Modelica modeling is centered on physical entities and lacks the explicit expression of the cognitive behavioral characteristics of agents. The embodied keyword is realized through the reconstruction of the syntax layer:

1. Ontology mapping: forcibly declare the interfaces of perceptual input and action output, so that the code structure is strictly isomorphic with the perceptual action cycle of embodied intelligence.
2. Quantification of cognitive delay: by using cognitivedelay and other built-in parameters, the biologically inspired nerve conduction delay (about 50-200ms) is incorporated into the dynamic equation to avoid the distortion problem that the control delay is simplified to the ideal zero delay in the traditional modeling.
3. Constraint prefix declaration: physical constraints are required to be declared through constraint blocks at the agent definition stage to ensure that subsequent control strategy development naturally meets the physical feasibility.

The extension syntax is as follows:

Listing 1. Embodied keyword

```
embodied <AgentName>
  extends <BaseClassName>;
  parameter <Type> <paramName> =
<value> "Description";
  input <Type> <sensorName>[N] ";
  output <Type> <actuatorName>[M];
  replaceable
<PolicyPackage>.<Algorithm> policy
    constrainedby EmbodiedPolicyInterface;
  constraint <ConstraintType>
<constraintName>
equation
  <equation>;
end <AgentName>;
```

- physical constraint fusion modeling, introducing the physical constraint description equation, allowing the PINN prediction results to be embedded in the Modelica model.

Traditional physical modeling and data-driven modeling have been separated for a long time, mainly in the following aspects:

1. White box model dilemma: pure physical equations are difficult to accurately describe complex nonlinear phenomena (such as turbulence and material fatigue), and the error of traditional Navier Stokes equations in a ship motion case is significant.
2. Black box model risk: the pure neural network prediction may violate the law of conservation of physics, and the non-conservation of energy in the trajectory prediction of a manipulator leads to simulation collapse.

The differential equation level fusion of physical equations and neural networks is realized through the PINN_interface syntax. The extended syntax includes:

Listing 2. PINN_interface

```
equation
 <OutputVariable> = PINN_Interface(
   inputs = {<InputVariable1>,
<InputVariable2>, ...},
   modelFile = "pinn_model.onnx")
  <PhysicalVariable> = <PhysicsEquation>
+ PINN_CorrectionTerm ;
```

- physical constraint Reinforcement Learning Library. Develop physical component library and provide DRL algorithm with physical residual regularization (such as PPO Physics). Its loss function is defined as:

$$L = \alpha \cdot L_{DRL} + \beta \cdot \|N(u) - PDE(U)\|^2 \quad (1)$$

Define DRL algorithm class with physical constraints. Take PPO physics algorithm as an example:

Listing 3. PPO physics

```
package PhysicsRL
model PPO_Physics
   extends DRL.BasePPO;
   parameter Real lambda_physics = 0.5 ;
  protected
   function compute_loss
     input Real[:,:] trajectories;
     input PhysicalModel physics_model;
     output Real total_loss;
   algorithm
     Real policy_loss :=
compute_policy_loss(trajectories);
     Real value_loss :=
compute_value_loss(trajectories);
     Real physics_residual :=
physics_model.evaluate_residual(
       states = trajectories[:,1:state_dim],
       actions =
trajectories[:,state_dim+1:end]);
     total_loss := policy_loss + value_loss +
       lambda_physics * physics_residual^2;
   end compute_loss;
  end PPO_Physics;
end PhysicsRL;
```

The specification provides the underlying support for industrial 4.0 core scenarios such as digital twins and agent group collaboration, and promotes the leap of embodied intelligence from theoretical verification to engineering implementation.

The embodied space architecture based on Modelica mainly includes four core modules:

- Physical domain modeling layer: build a 6-DOF ship dynamics model based on Modelica/MultiBody, integrate wave disturbance

DOI 10.3384/ecp218591

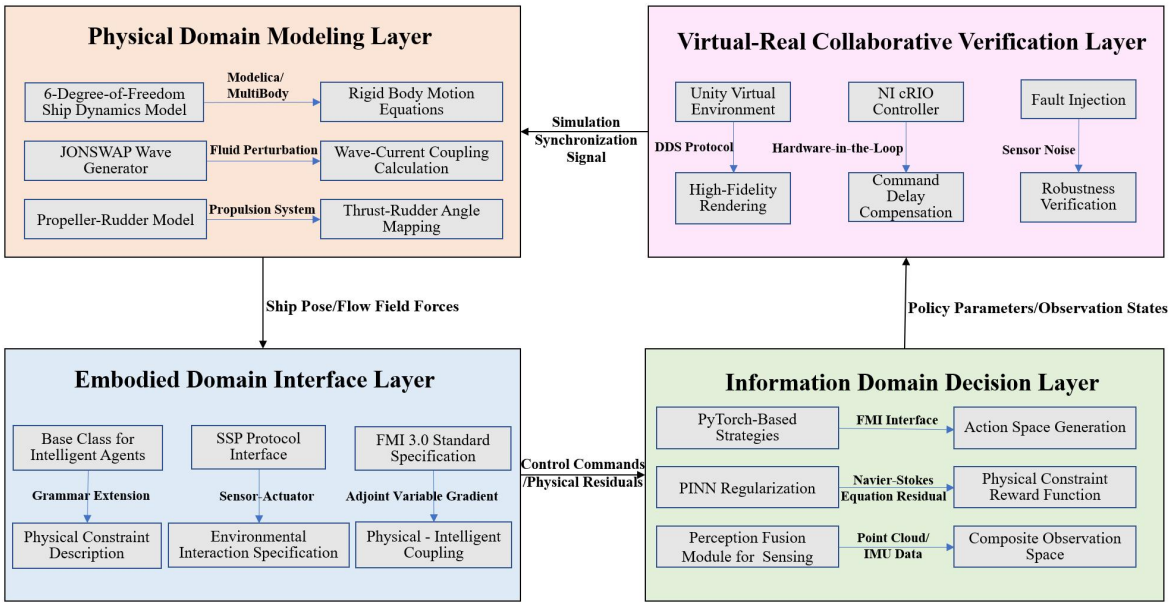


Figure 2. Embodied space architecture diagram based on Modelica.

(JONSWAP spectrum generator) and propulsion system (propeller actuator joint model);

- Embodied domain interface layer: extend Modelica language to define the embodied agent base class, support sensor actuator interface (SSP protocol) and physical constraint description syntax;
- Information domain decision-making layer: call the DRL policy network implemented by PyTorch through Python Modelica bidirectional interface (FMI standard) to dynamically generate control instructions;
- Virtual and real collaborative verification layer: build a high fidelity marine virtual environment based on unity engine, and realize millisecond data synchronization with Modelica simulation through DDS (data distribution service).

2.3 Modelica Embodied Space Architecture

2.3.1 Technical route

Traditional engineering often faces problems such as The main design idea of the embodied space based on Modelica is to achieve the two-way cognitive coupling between intelligent agents and the dynamic environment through the physical-information fusion mechanism, forming a three-layer architecture of "physical entity-digital twin-embodied intelligence". For different physical entities of autonomous systems, a system model and control algorithm are constructed based on the multi-disciplinary unified modeling software MWORKS.Sysplorer to achieve a control closed loop between the system model and the control algorithm. A visualization environment for the autonomous system is built based on the Unity 3D virtual engine to form a digital prototype of the autonomous system. Based on the virtual-real collaborative verification protocol stack of API interfaces, the SSP protocol, and the FMI standard, a unified solution architecture for physical models and autonomous decision-making algorithms is realized, breaking down the barriers between the virtual world and the physical world, expanding the boundary conditions of the digital prototype, and enhancing the ability of the physical entities of the autonomous system to learn independently and adapt to complex environments, thus forming a digital twin in the embodied space. Based on the PINN-Modelica gradient interaction protocol, an embodied intelligence training environment is added on the basis of the digital twin of the autonomous system, integrating key technologies such as perception fusion and decision-making reasoning to improve the intelligent characteristics of the embodied entity. The overall technical roadmap and framework diagram of the embodied space are as shown in the figure 2.

2.3.2 Implementation framework

The embodied space architecture based on Modelica mainly includes four core modules, as shown in the figure 3:

- Physical domain modeling layer: build a 6-DOF ship dynamics model based on Modelica/MultiBody, integrate wave disturbance (JONSWAP spectrum generator) and propulsion system (propeller actuator joint model);
- Embodied domain interface layer: extend Modelica language to define the embodied agent base class, support sensor actuator interface (SSP protocol) and physical constraint description syntax;

- Information domain decision-making layer: call the DRL policy network implemented by PyTorch through Python Modelica bidirectional interface (FMI standard) to dynamically generate control instructions;
- Virtual and real collaborative verification layer: build a high fidelity marine virtual environment based on unity engine, and realize millisecond data synchronization with Modelica simulation through DDS (data distribution service).

environment and physical deployment environment. For the Modelica model of the unmanned ship, a general theoretical model of the unmanned ship is formed based on the physical modeling system platform, and a physical simulation interface is formed based on the standard API interface or communication protocol to interact with the three-dimensional visual environment; For the intelligent algorithm construction and data training scenario, based on the virtual training environment and the universal model of unmanned ship,

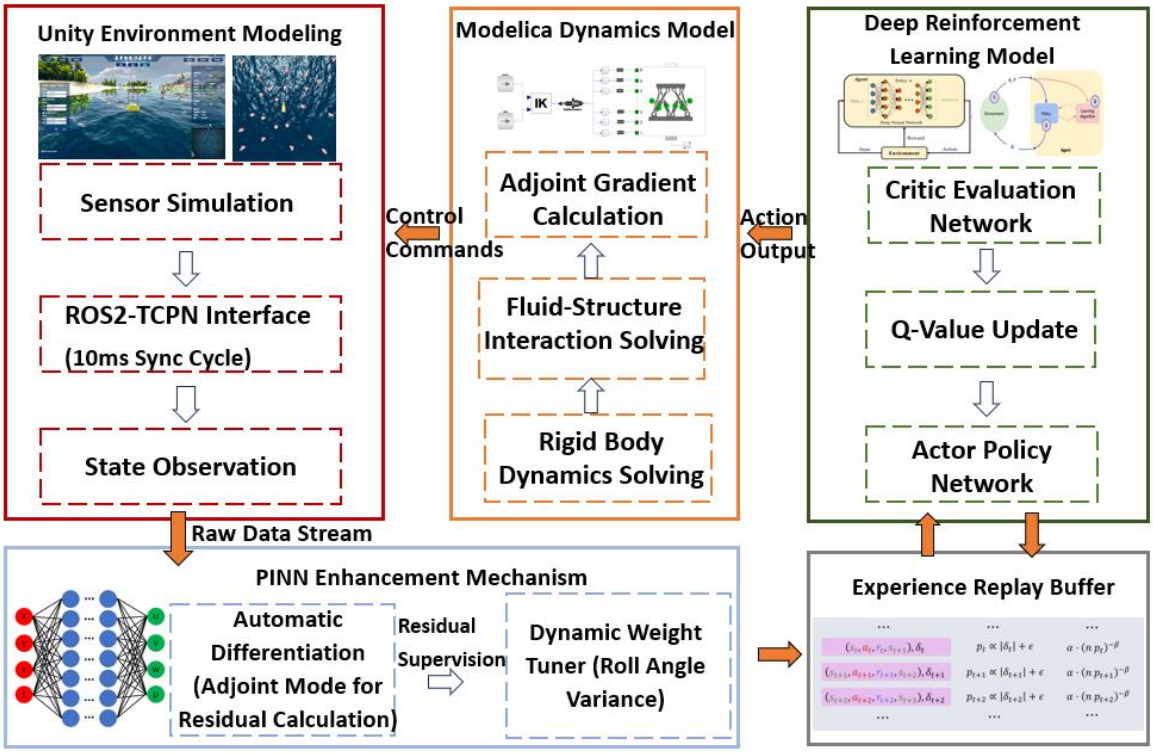


Figure 3. PINN enhanced DRL framework.

2.3.3 Virtual-Physical Co-Simulation Protocol Stack

The virtual real collaborative verification protocol stack of embodied space is the core module to realize the system integration and co-simulation of embodied information physical system. On the one hand, the virtual real collaborative verification protocol stack can organically integrate the digital sample ship and intelligent algorithm to form an intelligent ship model for virtual simulation verification; On the other hand, the virtual and real collaborative verification protocol can complete the data interaction between the virtual model and the physical ship, forming the virtual and real collaborative verification of unmanned ship.

1. Cross-Platform Interface Design.

The interface design of embodied information physical system needs to consider three levels of platform data interaction: physical modeling system, virtual training

the virtual environment interface is formed for the physical simulation interface, and the theoretical calculation is introduced as the constraint boundary in the three-dimensional visual environment and the large data sample training environment, so as to form a decision-making strategy model suitable for the actual unmanned ship. For the physical deployment environment of unmanned ship, the strategy deployment interface is constructed, and the algorithm model of rapid deployment training is completed, so as to realize the virtual real combination of unmanned ship and build a complete unmanned ship body space.

Physical simulation interface: a collaborative simulation interface for building physical models based on the MWORKS platform API interface or the general communication protocol standard. By analyzing the differential algebraic equation (DAE) generated by Modelica multi domain unified modeling language, the 6-DOF model of the unmanned ship can realize data

DOI 10.3384/ecp218591

interaction on different platforms, and finally realize the decoupling deployment of the physical simulation unit. Through the topology description file based on the physical simulation interface, the port connection relationship and parameter metadata are automatically maintained, which significantly reduces the repeated modeling cost of data collaboration.

Virtual environment interface: Based on the ROS2 distributed communication framework, develop customized TCPN (timed colored Petri net) plug-in to build a deterministic data channel. The high fidelity 3D visualization model is embedded in unity virtual environment, and the status synchronization with Physical model of unmanned ship is realized through ROS2-TCPN middleware. A 10ms fixed cycle hard real-time communication link is established.

Policy deployment interface: build AI policy deployment interface based on ONNX (Open Neural Network Exchange) open format. After compatibility check and optimization, the deep reinforcement learning strategy network trained by PyTorch/TensorFlow is transformed into a lightweight ONNX model. By integrating the ONNX algorithm engine, the end-to-end reasoning strategy deployment is realized on embedded controllers (such as NVIDIA Jetson AgX Xavier), which meets the real-time control requirements of edge devices under complex conditions.

2. Multi-Temporal Scheduling Mechanism.

The interface design of embodied information physical system under different platform levels and scenarios, the interface scheduling frequency is different, so it is necessary to add multi time scale scheduling module to the virtual and real collaborative verification protocol stack, including hard real-time layer, soft real-time layer and asynchronous event layer.

Hard real-time layer (1ms level): the hard real-time layer refers to building a real-time unmanned ship system model based on Modelica in the physical modeling system platform, and using the DAE (Differential Algebraic Equation) implicit solution algorithm to deal with the coupling effect between the six degree of freedom motion of the unmanned ship rigid body and fluid viscosity. By introducing variable step size integration algorithm or fixed step size integration algorithm, the real-time simulation control is realized on FPGA hardware accelerator card, which provides the bottom constraint boundary for the UAV body space frame.

Soft real-time layer (10ms level): soft real-time layer refers to the realization of data interaction between virtual environment and unmanned ship digital model based on virtual environment interface when intelligent strategy algorithm reasoning and 3D Virtual Environment Rendering of unmanned ship are carried out in virtual environment platform. The soft real-time layer has no real-time requirements for the embodied space frame, and its overall delay is stably controlled within 10ms.

Asynchronous event layer: when unmanned ships handle discrete events such as obstacle generation and task reset, it is necessary to design event driven architecture (EDA) to design discrete event processing mechanism to realize decoupling scheduling of aperiodic operations such as obstacle generation and task reset.

3 Case Study: Intelligent USV

3.1 Case Architecture Design

The embodied domain, as an intermediary layer between the physical domain and the information domain, realizes the deep integration of physical laws and information algorithms through the dynamic perception and adaptive decision - making of embodied agents. The experimental case selects the Embodied Space Simulation Platform on MoHub (https://mohub.net/). This platform is built based on MWORKS and is a comprehensive artificial intelligence platform focusing on embodied space modeling, unmanned system modeling, and simulation testing. Through the integration of multiple spaces and the collaboration of modular engines, it achieves a full - link closed - loop of the physical domain - embodied domain - information domain.

In this case, the USV - 130 unmanned surface vehicle is taken as the research object. Aiming at its navigation and obstacle - avoidance problems during rescue and search operations under uncertain conditions, an embodied intelligence modeling and training framework based on the MWORKS platform and PINN is proposed. Through modeling, simulation, and experimental verification, PINN is used to provide kinematic and dynamic constraints that conform to physical laws for the obstacle - avoidance algorithm. The influence of three environmental factors, namely wind, waves, and currents, is considered, and the calculation of the corresponding disturbing forces and torques is added to the motion mathematical model to make the algorithm simulation more consistent with reality.

3.2 Implementation of Embodied Domain Modeling

In the modeling and simulation of unmanned surface vehicles, high - precision hydrodynamic modeling is a key challenge for achieving the "simulation - to - reality" transfer. Traditional fluid resistance prediction methods based on empirical formulas, due to simplified assumptions, struggle to accurately describe the nonlinear hydrodynamic behavior in complex environments. In this case, PINN (Physics - Informed Neural Network) is introduced. By integrating the Navier - Stokes equations with measured flow field data, a data - physics dual - driven fluid resistance prediction model is constructed. In the MoHub Embodied Space

Simulation Platform, this method has established a multi - scale hydrodynamic calculation framework. At the macroscopic level, the fluid motion control equations are embedded as soft constraints, and at the microscopic level, the data from the pressure sensors on the hull surface are used to optimize the network parameters. This hybrid modeling approach significantly improves the calculation accuracy of complex flow states, reduces the error in the hull resistance characteristics between the virtual and real environments. Moreover, by encoding the spatio - temporal coupling characteristics of the wind, wave, and current disturbance fields into the input dimensions of PINN, the real - time calculation of dynamic environmental disturbance forces is realized, providing a basis for the physically reliable training of motion control algorithms.

3.2.1 Physical Domain Modeling

USV130 is an intelligent unmanned surface vehicle specially designed for scientific research and education. It integrates multi-source sensing system and has intelligent control modules such as speed and heading locking cruise, autonomous return path planning and obstacle avoidance. The main structure of the equipment is made of glass fiber composite materials. The whole ship is 1.3m (L) × 0.64M (W). It is equipped with a dual water jet propulsion system to achieve mobile control. The technical parameters show that its navigation performance includes: 18 cm working draft, 4 m/s peak speed, and 5 m minimum radius of gyration. The propulsion system can support high-precision motion control in complex hydrological environment. As shown in the following figure 3.



Figure 3. Physical Image of USV130 Unmanned Submarine

USV130 is equipped with Ubuntu 20.04 operating system and ROS noetic robot framework, and the underlying motion control adopts PID algorithm. The driving system supports multimodal operation: it can be controlled by the physical remote-control device, the upper computer software and the handle linkage or executed by the autonomous command based on the algorithm. The communication system is equipped with a dedicated base station to achieve stable data transmission with an effective radius of 200 meters.

The physical parameters required to build the unmanned ship are as follows:

Table 1. USV130 partial physical parameter table.

physical parameter	
Hull mass	30kg
Total length	1.3m
Geometric center	0.45m
Total width	0.64m
Hull width	0.27m
Draft	0.18m

3.2.2 Physics-Information Modeling

In this case, a dynamic model based on MWORKS platform is constructed to describe the motion behavior of the unmanned ship under complex hydrodynamic conditions. The dynamic model framework is divided into three parts: dynamic modeling framework, dynamic formula framework and additional mass matrix Mv framework.

- Dynamic equation modeling: the three degree of freedom dynamic equation of the system includes longitudinal velocity, lateral velocity and yaw angular velocity. The specific expression is as follows.

$$Mv + C(v)\mathrm{v} + D(v)\mathrm{v} = \tau \quad (2)$$

Where, M is the mass matrix, $C(v)$ is the Coriolis force matrix, $D(v)$ is the resistance matrix, and τ is the control input generated by the water jet propulsion system.

- Adaptive backstepping controller:

1. Speed control: realize the stable control of longitudinal speed through feedback linearization, and introduce adaptive compensation term to eliminate the uncertainty of resistance.
2. Heading control: Lyapunov function is designed to ensure heading error convergence. The controller expression is as follows:

$$\tau_z = k_1 e_\psi + k_2 r \quad (3)$$

Where, e_ψ is heading error and r is yaw rate.

3. Adaptive parameter estimation.

The adaptive law is introduced to estimate the resistance coefficient and added mass online, and compensate the uncertain disturbance of the system in real time to ensure the robustness and stability of the control system.

4. Improvement of integrated PINN.

In order to further improve the ability of the model to follow the physical laws, the physical information neural network (PINN) can be combined with the unmanned ship dynamics model. The Navier Stokes equation is embedded into the neural network as a physical constraint, and the PINN interface is introduced into the Modelica model to modify the hydrodynamic term in the

DOI 10.3384/ecp218591

dynamic equation of the unmanned ship, so that the model can strictly follow the hydrodynamic law while learning the data.

The core of the modeling framework of the dynamic model is to integrate the physical parameters and dynamic characteristics of the unmanned ship. The dynamic equation takes the force and motion state of the hull as the core, and calculates the motion response of the unmanned ship through moment balance. The framework contains the following key inputs:

- physical parameters.

Mass: including the influence of load change on the inertial characteristics of the hull.

Geometric center (LCG): describes the longitudinal position of the center of gravity of the hull.

Drag coefficient (CD) and water density (ρ): affect the hydrodynamic characteristics of unmanned ships.

Hull size parameters: including length (L), draft (T), and hull width (Bhull).

- function and modeling of hydrodynamic coefficient.

Hydrodynamic coefficients (HC) are the key parameters connecting hydrodynamic characteristics and motion equations, and their accuracy directly affects the prediction ability of the simulation model. For the catamaran USV, the hydrodynamic coefficients are divided into linear and nonlinear categories: linear damping coefficient (such as) represents the viscous resistance at low speed; The nonlinear damping coefficient (such as) describes the resistance characteristics dominated by the square term of speed under high-speed conditions. In addition, the additional mass coefficient (such as) reflects the inertial effect of the surrounding fluid when the hull accelerates.

3.3 Embodied Domain Simulation Analysis

After constructing the physical model of the unmanned ship based on the MWORKS.Sysplorer platform and implementing the virtual-real collaborative verification protocol stack based on the ROS framework, by randomly setting routes and obstacles, multi-level and multi-scenario data simulation can be carried out within the embodied space. The simulation data can further improve the training samples, enabling the self-improvement and autonomous learning of the PINN algorithm.

Environmental factors are one of the main factors that separate the unmanned ship model from the actual unmanned ship. By setting dynamic environmental parameters in the virtual environment engine, the ROS framework can transform the perception of disturbance factors such as wind, waves, and currents, achieving the deep coupling between the additional disturbances of the dynamic environment and the physical model of the unmanned ship. This adds physical boundary constraints on the basis of the real embodied training environment. The figure 4 below shows the disturbances transmitted to the physical model of the unmanned ship through the ROS framework.

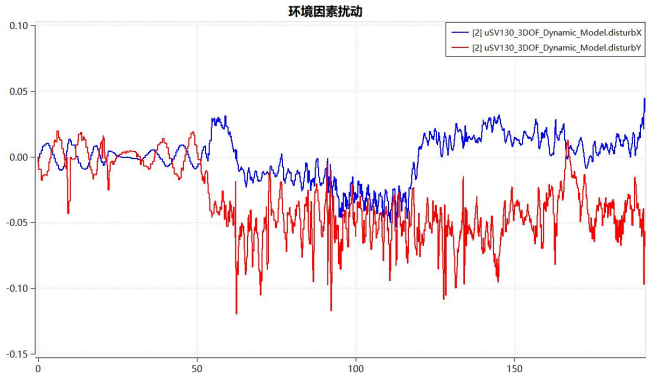


Figure 4. Dynamic Environmental Disturbances of the Unmanned Ship

After setting the route, the entire unmanned ship automatically locates its current position and the target route landmarks. It conducts navigation control by adopting the approach of global path planning and local obstacle avoidance, thus achieving the automatic navigation of the unmanned ship. After the simulation runs, the path changes of the unmanned ship in the dynamic environment can be observed, as shown in the figure 5 below.

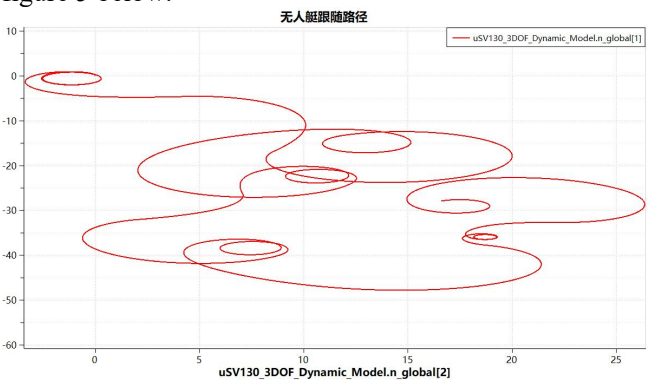


Figure 5. Response Path of the Unmanned Ship

3.4 Simulation Training in the Embodied Space

Traditional physical - based modeling methods have always been unable to clearly describe the hydrodynamic laws of the unmanned ship system. To achieve more accurate virtual - real collaborative verification, the Physics - Informed Neural Network (PINN) is introduced. It corrects the additional hydrodynamic effects in the unmanned ship system, further enhancing the model's ability to follow physical laws and adapt to dynamic environments, and ensuring the accuracy and reliability of the simulation results.

3.4.1 Hydrodynamic Laws

- Navier - Stokes Equation

The N-S equation, as the governing equation for describing the motion of viscous fluids, has its conservation form embedded in the loss function of the Physics - Informed Neural Network (PINN):

$$Residual = \frac{\partial u}{\partial t} + (u \cdot \nabla)u - v\nabla^2 u + \frac{1}{\rho}\nabla p \quad (4)$$

By forcing the prediction results of the neural network to minimize the equation residuals, it ensures that the flow field predictions (such as the flow velocity around the hull and the pressure distribution) conform to the principles of viscous fluid mechanics.

- Added Mass Effect

Based on potential flow theory, the inertial effect of the surrounding fluid when the hull accelerates is calculated as follows:

$$M_A = diag(X_{\dot{u}}, Y_{\dot{v}}, N_{\dot{r}}) \quad (5)$$

The added mass coefficients are determined by empirical formulas based on the hull's geometric parameters, reflecting the coupling characteristics of the hull shape on the fluid inertial response.

- Nonlinear Viscous Resistance Model

A resistance decomposition dominated by the velocity - squared term is adopted:

$$D_{(v)} = \begin{bmatrix} X_{u|u|}|u|u & 0 & 0 \\ 0 & Y_{v|v|}|v|v + Y_r r & N_v|v|v + N_r r \end{bmatrix} \quad (6)$$

The longitudinal resistance coefficient $(X\{u|u|\})$ is calibrated according to the wetted surface area of the hull and the Reynolds number, and the lateral/rotational damping coefficients $(Y\{v|v\}, N_r)$ are fitted from the ship model towing test data.

Coriolis - Centrifugal Force Coupling Effect

It is explicitly expressed in the hull motion equation as:

$$D_{(v)} = \begin{bmatrix} 0 & 0 & -m(y_g r + v) \\ 0 & 0 & m(x_g r - u) \\ m(y_g r + v) & -m(x_g r - u) & 0 \end{bmatrix} \quad (7)$$

where (x_g, y_g) is the position of the center of gravity, which directly affects the dynamic coupling of the roll - yaw motion.

3.4.2 Application of the PINN Algorithm

- PINN - corrected Resistance Prediction

Since the traditional resistance model has significant errors in turbulent flow conditions, for the calculation of the fluid resistance of the unmanned ship, the N - S equation is embedded as a soft constraint in the PINN algorithm for training, and a hybrid resistance model is constructed as follows:

$$\dot{D}(v) = D_{phys}(v) + PINN(v; \theta) \quad (8)$$

- Dynamic Compensation of Added Mass

Considering that the added mass coefficient of the unmanned ship changes with the draft (load fluctuations cause the draft to change by ±10 cm), a PINN surrogate model $M_A = f(T, nabla)$ is established for real - time online estimation:

- Fluid - Motion Coupling Decision - making

Traditional DRL strategies tend to generate actions beyond the feasible domain of fluid dynamics (for example, a sharp turn causing the lateral velocity to exceed the limit). Therefore, it is necessary to introduce a fluid - constraint reward term in the PPO algorithm:

$$r_{physics} = -||v_y|| \cdot \prod(|v_y| > v_y^{max}) - ||r|| \cdot \prod(|r| > r^{max}) \quad (9)$$

and calculate the gradient penalty of the Navier - Stokes equation on the policy parameters through automatic differentiation.

- Wave Disturbance Modeling

Considering the phase lag of the heave/pitch excitation force generated by the JONSWAP wave spectrum, a wave force transfer function is established based on potential flow theory:

$$F_{wave} = \int_{-\infty}^{\infty} H(\omega) \cdot S(\omega) e^{i\omega t} d\omega \quad (10)$$

Where ω is calculated by the hull strip theory, and $S(\omega)$ is the wave spectrum density.

3.4.3 Deployment and Operation of the Unmanned Ship System

In the deployment and operation phase, the preset water scene and the digital prototype of the ship are loaded based on the unity virtual simulation environment, and the sensor data stream (lidar, visual camera) and physical engine parameters are bound through the ROS-TCP communication protocol; Start the obstacle avoidance algorithm node within the ROS framework, load the pre trained PPO/TD3 reinforcement learning model and supervised learning strategy, and run MWORKS Sysplorer carries out ship dynamics calculation; The motion control module drives the digital sample ship model according to the real-time decision-making instructions (path planning, obstacle avoidance angular speed), and feeds back the ship position and attitude, environment interaction status to the simulation interface through ROS Topic; Finally, a real-time closed loop of "perceptual data acquisition - algorithm online reasoning - control instruction execution - physical state return" is formed. During operation, the system supports dynamic adjustment of environmental parameters (wind wave current intensity, obstacle distribution), and continuously optimizes the strategy network through PyTorch's online learning mechanism to ensure the autonomous navigation robustness and task adaptability of unmanned craft in complex scenes.

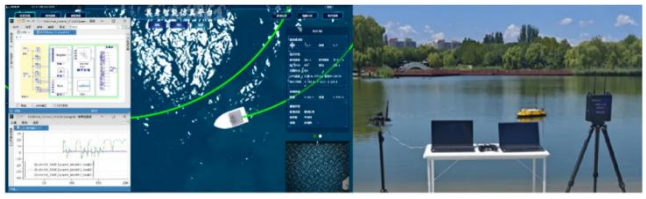


Figure 6. Digital prototype and physical prototype

DOI 10.3384/ecp218591

4 Conclusion

This study proposes an innovative framework of embodied information physical system (ECPS) based on Modelica/MWORKS, and constructs a "physical information embodied" ternary collaborative architecture through the extension of modeling language, which breaks through the separation limitations of physical laws and intelligent decision-making in traditional tools. The framework innovatively integrates the physical information neural network and multi domain modeling theory, and realizes the deep coupling of physical simulation and autonomous decision-making in the field of intelligent unmanned system. By constructing the reinforcement learning paradigm enhanced by PINN, the fluid dynamics equation constraint is embedded in the strategy network training, which significantly improves the ability of the agent to follow the physical laws; Relying on the FMI/SSP standardized protocol, a virtual and real collaborative verification system is built to effectively solve the timing conflict between high-precision simulation and real-time decision-making. The research results provide an integrated methodology of interpretable modeling, verifiable decision-making and scalable deployment for the upgrading of industrial intelligence, promote the leap of embodied intelligence from theoretical verification to engineering implementation, and establish a new benchmark for the development of independent systems in the era of digital transformation.

References

[1]Lin J, Yu W, Zhang N. A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications[J]. IEEE Internet of Things Journal, 2017, 4(5): 1125-1142.

[2]Gratch J. The promise and peril of interactive embodied agents for studying non-verbal communication: a machine learning perspective[J]. Philosophical Transactions of the Royal Society of London, Series B. Biological Sciences, 2023, 378(1875).

[3]Liu Y, Chen W, Bai Y, et al. Aligning Cyber Space with Physical World: A Comprehensive Survey on Embodied AI[M]. arXiv, 2024.

[4]Rong J, Luan N. Safe Reinforcement Learning with Policy-Guided Planning for Autonomous Driving[J]. IEEE, 2020.

[5] China Automotive Engineering Research Institute Co., Ltd. (Guangzhou), China Automotive Data Co., Ltd. (Tianjin), & China Automotive Data Co., Ltd. (2024). An intelligent prediction method for Modelica fluid models (CN Patent No. CN202311575134.1). China National Intellectual Property Administration. Issued April 9, 2024.

[6]Penwarden M, Zhe S, Narayan A, et al. A Metalearning Approach for Physics-Informed Neural Networks (PINNs): Application to Parameterized PDEs[M]. (2023-01-19).

[7]Sivashangaran S, Zheng M. Intelligent Autonomous Navigation of Car-Like Unmanned Ground Vehicle via Deep Reinforcement Learning[J]. IFAC-PapersOnLine, 2021, 54(20): 218-225.

[8]Marian M, Stinga F, Georgescu M T, et al. A ROS-based Control Application for a Robotic Platform Using the Gazebo 3D Simulator[C]//2020 21th International Carpathian Control Conference (ICCC). High Tatras, Slovakia: IEEE, 2020: 1-5.

[9]Chen Y, Koohy S. GPT-PINN: Generative Pre-Trained Physics-Informed Neural Networks toward non-intrusive Meta-learning of parametric PDEs[M]. arXiv, 2023.

[10]Shi H, Shi L, Xu M, et al. End-to-End Navigation Strategy With Deep Reinforcement Learning for Mobile Robots[J]. IEEE Transactions on Industrial Informatics, 2020, 16(4): 2393-2402.

DOI
10.3384/ecp218591

On the precise and efficient representation of industrial controllers in Modelica

Alberto Leva[1]

[1]DEIB, Politecnico di Milano, Italy alberto.leva@polimi.it

Abstract

This paper highlights some still-open issues in the Modelica representation of industrial controllers, particularly when simulation efficiency must balance with fidelity to their real-world implementation as digital components of cyber-physical systems. The treatise focuses on PID controllers due to their predominant role, though the ideas readily extend to virtually any other control structure. The aim is to stimulate discussion around this important yet often overlooked topic, while also suggesting directions for future development.

Keywords: Modelica, industrial controllers, PID controllers, digital control, digital twins, cyber-physical systems.

1 Introduction

Over the last decades, the Modelica ecosystem has grown to embrace a wide range of domains where control systems play a fundamental role. In many of those domains – process systems being a notable example – control blocks are combined to form articulated structures (Skogestad 2023). These structures are then deployed onto heterogeneous hardware/software architectures, resulting in complex cyber-physical systems. The *realistic* simulation of such systems is still a non completely solved open problem, as conventional "textbook" controller models fall short in terms of completeness, flexibility, and – if fidelity in the digital part matters – of computational efficiency (Cimino et al. 2023).

In the following we discuss the problem just evidenced, referring to the PID controller due to its generality and diffusion — though the underlying ideas are evidently general. After the brief motivating example in Section 2, Section 3 deals with completeness, illustrating the I/O structure required to manage a control block in a realistic application setting; Section 4 describes the various forms that a linear (PID) control law can take, while Section 5 addresses nonlinear functionalities like saturation management and shows how differently they can be realised.

The compound of the above should give the reader an idea of how flexible a controller model should be in order to put the analyst in the position of representing what will be really implemented, precisely enough and with an acceptable effort. Section 6 spends just a very few words on the important problem of simulating a control block accounting for its digital realisation, i.e., of balancing fidelity with computational efficiency. Finally, Section **??** draws some conclusions and sketches out the desired effects of the discussion we aim to trigger.

Throughout the paper, key design decisions are highlighted. While some recommendations are offered, no definitive solutions are proposed, as the complexity and criticality of the overall problem make it difficult to predict the long-term consequences of present choices. The aim of this work, as just said, is in fact primarily to foster informed discussion within the community, so as to promote thoughtful and effective decisions going forward.

2 A motivating example

First of all, before entering the problem, the reader might ask whether we really need to bother. The answer could take more than one paper, but suffice a toy example. Consider a control loop with a process and a PI controller, respectively described by

$$P(s) = \frac{\mu}{(1+sT_1)(1+sT_2)}, \quad C(s) = K\left(1+\frac{1}{sT_i}\right) \quad (1)$$

with $\mu = 2$, $T_1 = 2$, $T_2 = 0.25$, $K = 2$, $T_i = 2$. Build for it five simulation models, all fully continuous-time, with different antiwindup strategies: actuation error feedback, also known as integral recomputation, with tracking time T_t (denoted by `aerr1` for $T_t = 2.5$ and `aerr2` for $T_t = 0.5$); conditional integration (denoted by `cint`); clamping (`clamp`), internal feedback (`ifb`).

In the absence of saturation all models behave exactly the same, as expected; we omit plots for brevity. We conversely show in Figure 1 what happens of the controlled variable (`PV_`$\star$) vs. the set point (`ref`) and of the control signal (`CS_`$\star$) in the presence of [0,1] saturation. It is evident that realism requires the simulated antiwindup mechanism to match the implemented one.

Antiwindup aside, several PID functionalities are managed by industry-standard controllers in heterogeneous ways, often involving additional inputs and outputs, both modulating and logic, with respect to the minimal interface of a controller. And to top, all of the above intertwines with the form chosen for the PID control law, as well as with its realisation as digital algorithm.

Summing up, if industrial realism is required, there is no single "reference" controller model implementation —

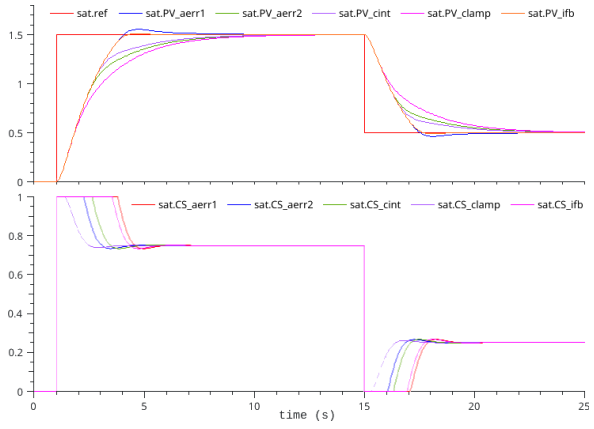


Figure 1. Motivating example: set point and controlled variable (top), control signal (bottom).

but at the same time, attempting to replicate the whole zoo of available products would not be practical. Indeed, a systematic design approach needs devising.

3 Interface and functionalities

Figure 2 illustrates a typical set of I/O signals for an industrial (PID) controller, adopting the Modelica colour coding, together with a brief description of their meaning. These signals refer to functionalities that are crucial for using PIDs in control structures; a few examples follow.

First, without a Bias input one cannot realise feedforward compensation, decoupling, or any structure where the output of a controller must receive an additive correction, as failure to account for that correction in the computation of the controller state may cause highly undesired behaviours. A notable case is backward decoupling, illustrated in Figure 3 by comparing the "textbook" model (top) and one in which the additive correction is duly introduced *upstream* saturation management (bottom).

Second, one cannot manage saturation in cascade controls without increment and decrement locks, as the saturation values for the inner controller come from the actuator, while those for the outer controller are substantially unknown. Assuming all gains positive for simplicity, one has to respectively connect the HI and LO saturation signals from the inner controller to the increment and decrement locks F+ and F- of the outer controller, so as to prevent it from pushing the inner one further into saturation if this occurs.

Third, and again about cascade controls, if the inner controller is set to tracking mode, the outer one must be forced into tracking as well. More in general, whenever a modulating control block is set to operate in any mode but automatic, all the blocks whose loops are consequently open cannot be in automatic mode (particularly if they are not asymptotically stable, as is the case when integral action is present). Enforcing this constraint requires logics to conveniently manipulate the TS and TR signals of the involved blocks. Figure 4, where the notation should be self-explanatory, provides an example of how the second and third point can be handled; variants are possible but discussing them is not our point here.

Fourth, TS and TR are necessary also to manage the switchover among two or more controllers: to avoid undesired bumps in the control signal applied to the rest of the system, all controllers must receive that signal as TR, the active one must be in automatic mode (TS=false) and all the others in tracking mode (TS=true). We omit a block diagram for this case given its simplicity.

More examples could be given – a longer discussion can be found in (Leva 2024) – but the point for this paper is that handling the above matter requires critical model design choices. Should we provide several PID blocks (from "basic" to "complete")? Or just one with flags (named for example hasTracking, has Locks and so forth) to enable/disable the required functionalities, plus the necessary conditional connectors? Or any combination thereof?

4 The LTI control law

The variety of forms or structures that the LTI (linear, time-invariant) control PID control law can take is large indeed (O'Dwyer 2009) but some "families" can be identified; with no exhaustiveness claim we could mention one degree of freedom (1-dof) forms, described as transfer functions $C(s)$ from the error $e := SP - PV$ to the control output CS, i.e.,

$$C(s) = K\left(1 + \frac{1}{sT_i}\right)\frac{1+sT_d}{1+sT_d/N}; \tag{2}$$

non-interacting (aka parallel), i.e.,

$$C(s) = K\left(1 + \frac{1}{sT_i} + \frac{sT_d}{1+sT_d/N}\right); \tag{3}$$

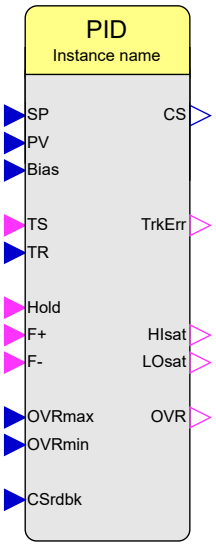
two degree of freedom (2-dof) forms characterised by different $SP \to CS$ and $PV \to CS$ signal paths, most notably the ISA one

$$CS(s) = K\Bigg(bSP(s) - PV(s) + \frac{1}{sT_i}(SP(s) - PV(s)) + \frac{sT_d}{1+sT_d/N}(cSP(s) - PV(s))\Bigg) \tag{4}$$

with the notable "output derivation" case ($c = 0$), as well as other analogous ones realised by pre-filtering the set point signal.

A source of confusion here comes from a certain tendency of the product literature to give different names to the same concept, but this is quite easy to cure with a proper documentation — and here too, a design equilibrium needs finding. More serious problems can arise from the different meanings of the PID parameters, as the reader can immediately guess from the small sample of possible structures in Figure 5.

DOI
10.3384/ecp218603



SP	Set point.
PV	Process (controlled) variable.
CS	Control signal.
Bias	Additive correction to CS (state consistency enforced).
TS	Track Switch: when true makes CS follow TR (state consistency enforced).
TR	Track Reference: followed by CS when TS is true.
TRkErr	Tracking Error: true if TS is true but for any reason (e.g., range) CS is not following TR.
HIsat	True when CS hits the high saturation value.
LOsat	True when CS hits the low saturation value.
Hold	When true causes CS to remain constant (state consistency enforced).
F+	Freeze plus: when true prevents CS from increasing (state consistency enforced).
F-	Freeze minus: when true prevents CS from decreasing (state consistency enforced).
OVRmax	Override max: takes over CS if greater than the computed value (state consistency enforced).
OVRmin	Override max: takes over CS if smaller than the computed value (state consistency enforced).
OVR	True if CS is overridden by OVRmax or OVRmin.
CSrdbk	Actuator readback for some antiwindup strategies (not discussed herein).

Figure 2. Typical I/O connections in an industrial (PID) controller.

In this case the problem will be finding a set of structures as small as possible but capable of accommodating for all the relevant LTI law variants. One structure cannot be enough because some have limitations and some do not (e.g., a series form like the topmost in Figure 5 cannot have complex zeroes), in some forms N modifies the zeroes with respect to the ideal PID form while in some others it does not, and there are other "system-theoretical" reasons to account for.

Furthermore, though in this paper there is not the space for addressing the subject with the required depth, the said set will need to encompass representations of biasing and direct/reverse action, as well as of velocity (as opposed to positional) forms; and moving toward non-LTI facts as we are going to do in the next section, of overriding and control signal rate limits.

5 Nonlinearities

The first item in this list is clearly saturation management or antiwindup, and here the author takes the occasion to suggest removing the adjective "integral" as for windup to occur a controller just needs to have a large enough gain and a slow enough mode with respect to the loop cutoff frequency. The point, in any case, is that here arbitrary choices are inevitable: denoting by $x_c(t)$ be the controller state and writing its output equation as

$$u(t) = c_c x_c(t) + d_c e(t); \tag{5}$$

when $u(t)$ is stuck into saturation, *any* x_c fulfilling (5) is I/O consistent and therefore fits — and notice that the same applies every time $u(t)$ is anyhow "altered" with respect to the value coming from the LTI law (whatever its form), that is with tracking, increment/decrement locks, holding, and so on.

There are other nonlinear functionalities in the list, and they interact with one another, and they could even *conflict* with one another: a few examples follow.

- Shall TR be allowed to violate saturation limits?
- Shall overrides prevail over tracking or *vice versa*?
- Shall increment/decrement locks be subject to tracking/holding or not?

Here too, different products make different choices, sometimes not immediate to figure out if not by thoroughly searching through their documentation. Further design problems are thus at the horizon, as we need to allow the analyst to take knowledgeable decisions because the behaviour of control structures is heavily impacted by facts like those just mentioned.

The author tends to prefer a solution in which the controller blocks do not contain any logic if not that entailed by the definition of I/O signals and their role as per Figure 2, while the coordination of the above functionalities (including the way possible conflicts are solved) is realised externally by means of logic blocks. Such a design approach is in the favour of flexibility and clarity, but on the other hand requires a certain level of user consciousness as well as modelling language knowledge: another point for the community to discuss.

6 Digital realisation

When modelling and simulating a digitally controlled process – or said in more recent terms, when creating and using a digital twin for a cyber-physical system (Lee 2008) – a precise representation of control algorithms is sometimes of relatively low importance but some other times extremely critical (Åström and Murray 2021; Cimino et al. 2024).

To effectively support a model-based design toolchain, especially for complex systems as discussed for exam-

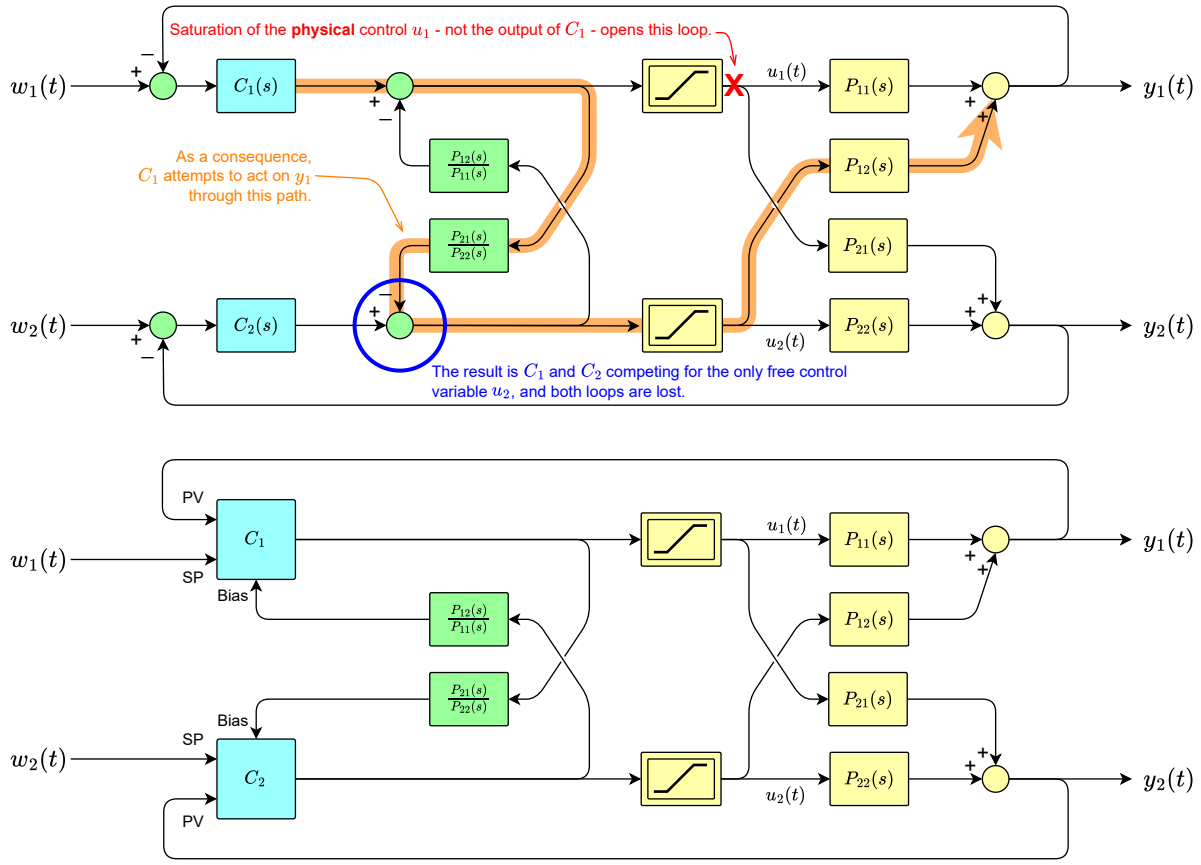


Figure 3. Decoupling control and the need for a Bias input.

ple in (Grieves and Vickers 2017), controller models must therefore be capable of operating sometimes in a time-driven manner, allowing for example for the use of variable-step solvers, and sometimes in an event-based manner, most typically (but not only) to represent the multi-clocked nature of industrial control architectures.

The Modelica language offers multiple means to address such needs, well known to the reader and not central to discuss in this paper. No matter what mix of the said means is adopted, however, the problem remains of structuring a controller model base in such a way that achieving and maintaining consistency between time- and event-driven (in a more industrial jargon, continuous- and discrete-time) models be as natural as possible.

To illustrate the ideas above, we report in Listing 1 a possible minimum set of base classes for Continuous-time (CT) and discrete-time (DT) PI models, and then in Listings 2 through 5 some elementary examples, referring to a 1-dof PI controller with antiwindup realised in the various ways mentioned in the motivating example in Section 2. Incidentally, these are the models used to produce the simulations reported in Figure 1; for the convenience of the reader willing to replicate he experiments and further investigate the matter, all the neces-

Listing 1. Base classes for continuous-time (CT) and discrete-time (DT) PI models.

```
partial model Base_CTPI
  parameter Real K      "gain";
  parameter Real Ti     "integral time";
  parameter Real CSmax "upper saturation value";
  parameter Real CSmin "lower saturation value";

  Modelica.Blocks.Interfaces.RealInput  SP
          "set point";
  Modelica.Blocks.Interfaces.RealInput  PV
          "process (controlled) variable";
  Modelica.Blocks.Interfaces.RealOutput CS
          "control signal";
end Base_CTPI;

partial model Base_DTPI extends Base_CTPI
  parameter Real Ts     "sampling time";
  // computed by algorithms in derived classes
  discrete Real cs;

equation
  cs = CS; // zero-order holder

end Base_DTPI;
```


DOI
10.3384/ecp218603

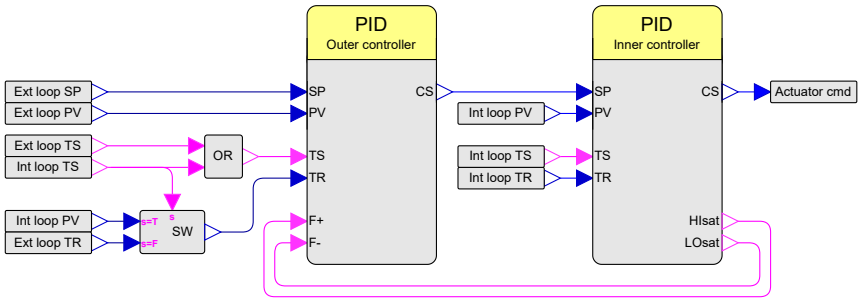


Figure 4. Cascade control and the need for locks and tracking logic (control scheme only, process not shown).

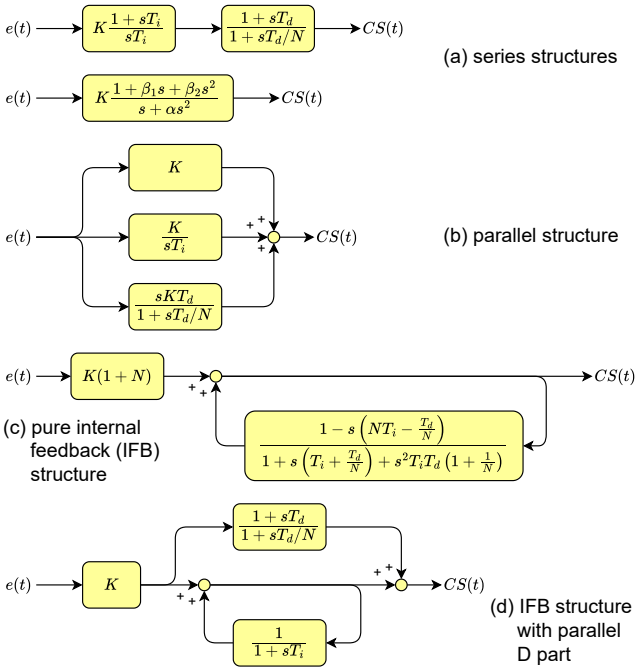


Figure 5. Some structures for the 1-dof LTI PID control law.

Listing 2. CT and DT PI models with actuation error anti-windup.

```
model CTPI_AERR extends Base_CTPI;
  parameter Real Tt;
  Real uns, ui(start=0);

equation
  der(ui) = K/Ti*(SP-PV)-(uns-CS)/Tt;
  uns     = ui+K*(SP-PV);
  CS      = max(CSmin, min(CSmax, uns));

end CTPI_AERR;

model DTPI_AERR extends Base_DTPI;
  parameter Real Tt;
  discrete Real csp, csi, csi1(start=0);

algorithm
  when sample(0, Ts) then
    csp  := K*(SP-PV);
    csi  := csi1+K*Ts/Ti*(SP-PV);
    cs   := max(CSmin,min(CSmax,csp+csi));
    csi1 := csi-(csp+csi-cs)*Ts/Tt;
  end when;

end DTPI_AERR;
```

sary Modelica code is available under a 3-clause BSD licence at `https://github.com/looms-polimi/Modelica_PID_MWEs`.

Coming back to the our main subject, te challenge is to move from extremely simple blocks like those just shown, where most of the problems mentioned above are not addressed, to others capable of representing all the needed functionalities of which we gave a sample in the previous sections. To undertake this challenge we need a methodological and a technological framework, connected to one another as strictly and rigorously as possible. On the methodological side a choice could be switching (nonlinear) dynamic systems, written as

$$\begin{cases} CS(t) &= g_{\sigma(t)}\big(x_c(t), SP(t), PV(t), TS(t)\dots\big) \\ \dot{x}_c(t) &= f_{\sigma(t)}\big(x_c(t), SP(t), PV(t), TS(t)\dots\big) \\ \sigma(t^+) &= s\big(\sigma(t), x_c(t), SP(t), PV(t), TS(t)\dots\big) \end{cases} \tag{6}$$

in the continuous time (aka time-driven) context and as

$$\begin{cases} CS(t_k) &= g^*_{\sigma(t_k)}\big(x_c(t_k), SP(t_k), PV(t_k), TS(t_k)\dots\big) \\ CS(t_{k+1}) &= f^*_{\sigma(t_k)}\big(x_c(t_k), SP(t_k), PV(t_k), TS(t_k)\dots\big) \\ \sigma(t_{k+1}) &= s^*\big(\sigma(t_k), x_c(t_k), SP(t_k), PV(t_k), TS(t_k)\dots\big) \\ t_{k+1} &= t_k + q \end{cases} \tag{7}$$

in the discrete time (aka *periodic* event-driven) one.

In both (6) and (7) the σ subscript indicates that the state and the output equation can take a finite number of forms, for example to match the operating modes of a block in each of which the unique state vector x_c is managed in a different manner. In (6) t is the continuous time and the $^+$ superscript denotes values after a σ switching; in (7) q is a time quantum such that switching events can only occur at its multiples. The reader interested in more detail on these aspects can refer to (Miskowicz 2018) and the works quoted therein.

As for the technological side, a starting point could be the IEC 61499 standard (Lyu and Brennan 2020; Wies-

Listing 3. CT and DT PI models with conditional integration antiwindup.

```
model CTPI_CINT extends Base_CTPI;
  Real ui(start=0);

protected
  // numerical workaround, might be critical
  parameter Real CSeps = (CSmax-CSmin)*1e-6;

equation
  der(ui) = if CS>CSmin+CSeps and CS<CSmax-CSeps
            then K/Ti*(SP-PV) else 0;
  CS      = max(CSmin, min(CSmax,K*(SP-PV)+ui));

end CTPI_CINT;

model DTPI_CIN extends Base_DTPI;
  discrete Real csp,csi0(start=0),csi,csi1;

algorithm
  when sample(0, Ts) then
    csp  := K*(SP-PV);
    csi0 := csi1+K*Ts/Ti*(SP-PV);
    csi  := if csp+csi0>CSmin and csp+csi0<CSmax
            then csi0 else csi1;
    cs   := max(CSmin, min(CSmax,csp+csi));
    csi1 := csi;
  end when;

end DTPI_CINT;
```

Listing 4. CT and DT PI models with clamping antiwindup.

```
model CTPI_CLAMP extends Base_CTPI;
  Real xde(start=0), de, du;

protected
  parameter Real tau = Ti/1000;

equation
  xde+tau*der(xde) = SP-PV;
  de               = (SP-PV-xde)/tau;
  du               = K*de+K/Ti*(SP-PV);
  der(CS)          = if CS < CSmin and du < 0
                     or CS > CSmax and du > 0
                     then 0 else du;

end CTPI_CLAMP;

model DTPI_CLAMP extends Base_DTPI;
  discrete Real csp, csi(start=0), csi1;

algorithm
  when sample(0, Ts) then
    csp  := K*(SP-PV);
    csi  := csi1+K*Ts/Ti*(SP-PV);
    cs   := max(CSmin,min(CSmax,csp+csi));
    csi1 := cs-csp;
  end when;

end DTPI_CLAMP;
```

Listing 5. CT and DT PI models with internal feedback antiwindup.

```
model CTPI_IFB extends Base_CTPI;
  Real x(start=0);

equation
  CS = x+Ti*der(x);
  CS = max(CSmin,min(CSmax,K*(SP-PV)+x));

end CTPI_IFB;

model DTPI_IFB extends Base_DTPI;
  discrete Real x(start=0), x1;

algorithm
  when sample(0,Ts) then
    x  := (Ti*x1+Ts*cs)/(Ti+Ts);
    cs := max(CSmin,min(CSmax,K*(SP-PV)+x));
    x1 := x;
  end when;

end DTPI_IFB;
```

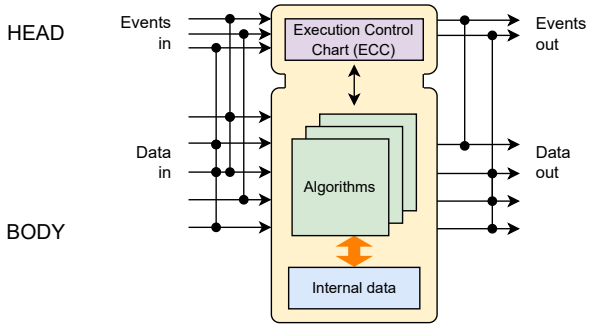


Figure 6. The concept of Function Block as per the IEC 61499 standard.

mayr, Mehlhop, and Zoitl 2023) and most notably its definition of Function Block (FB) exemplified in Figure 6. An FB is a modular, reusable component representing a functional unit in a distributed automation system. It offers data I/O to receive and produce information, and event I/O to execute and trigger actions.

Also, an FB is made of two main components, namely the Event-Condition-Action (ECA) automaton – or Execution Control Chart (ECC) – and the dynamic system. The ECA automaton governs the FB evolution based on events and conditions, triggering actions in response to specific events; the dynamic system is responsible for handling internal data, processing algorithms, and managing system state transitions over time.

The relationships between the two frameworks – methodological and technological – seem to the author quite evident and potentially capable of grounding the sought design and development process, yet they are not totally unambiguous, and in their application to the addressed domain decisions need taking. For example, continuous-time state/output equations naturally correspond to conditional DAEs, and discrete-time state/output

DOI
10.3384/ecp218603

equations to (branching) FB algorithms. Also, a sequential logic conceptually modelled as automaton naturally corresponds to an FB ECC in which one could talk about "mode σ" where σ takes values like AUTO, TRACKING, HIsat and so forth.

But coming back to our design-centric attitude, how could we *efficiently* implement this? By algorithm(s) with `when` clauses? With which event management policy? And most important, could we use the *same* automaton for time- and event-driven representations? Might we need (for the time-driven context) to represent as DAEs functionalities that were natively created as algorithms? Might we also need additional state variables for that purpose? Could this evidence numerical, solver-related issues/choices/precautions? As can be seen, we have a lot to discuss also on this front.

7 Conclusions and future work

Despite the numerous existing controller simulation models, also and particularly within the Modelica ecosystem, this paper has highlighted that substantial conceptual design choices remain to be made in the field. While many controller models – especially PID ones – are widely used, there is in fact a clear need for more flexibility to better reflect the complexities of real industrial systems, and most important, to allow an analyst to easily ensure that the simulated controls will match the particular implementation to consider. It is also important to stress that these choices are not merely technical; on the contrary, they involve a methodological shift that has yet to be fully realised.

In fact, at least to the best of the author's knowledge, no existing (Modelica) controller model has been entirely developed along the approach proposed here, and premising that the approach itself is still in its infancy and most likely contains several imperfections, nonetheless the observed *scenario* suggests that several critical design aspects are still unresolved. The challenge lies in synthesising the scattered and often fragmented knowledge available, alongside the need to adhere to industrial standards (the IEC ones being a paradigmatic but certainly not the unique example). Moreover, the development of such controller models requires careful and accurate planning, especially given the wide range of knowledge and expertise required. These conceptual and methodological gaps should be seen not just as challenges but also as opportunities for future research and innovation in the field of digital twinning for cyber-physical systems.

As an aside, though the matter could not find space in this paper, we have to notice that in addition to their relevance for industrial applications, flexible controller models also hold significant potential for educational purposes. These models could be particularly valuable in undergraduate and graduate programs, as well as in the context of continuing education, where they can bridge the gap between theoretical understanding and practical, real-world applications.

For the future, the hope is that this paper can stimulate a discussion on the problems here evidenced and (briefly) discussed, which he believes to be very relevant for the effective simulation of industrial (and particularly process) controls. Indeed, while the examples presented are intended – as said – to spark discussion, they could also mark a concrete first step toward a shared open-source initiative aimed at developing a structured, extensible library of industrial controllers, coordinating time- and event-based descriptions (here too, as exemplified above) to conjugate precision and efficiency. Yet, defining a coherent architecture and sustainable development – and particulary, maintenance – practices demands collective input, which is why the author warmly invites the community to engage in shaping both the technical direction and governance needed to realise such an initiative.

References

Åström, Karl Johan and Richard M Murray (2021). *Feedback systems: an introduction for scientists and engineers*. Princeton, NJ, USA: Princeton University Press.

Cimino, Chiara et al. (2023). "Efficient control representation in Digital Twins: An imperative challenge for declarative languages". In: *IEEE Transactions on Industrial Informatics* 19.11, pp. 11080–11090.

Cimino, Chiara et al. (2024). "Scalable and efficient digital twins for model-based design of cyber-physical systems". In: *International Journal of Computer Integrated Manufacturing* 37.10-11, pp. 1232–1251.

Grieves, Michael and John Vickers (2017). "Digital twin: mitigating unpredictable, undesirable emergent behavior in complex systems". In: *Transdisciplinary Perspectives on Complex Systems*, pp. 85–113.

Lee, Edward A (2008). "Cyber physical systems: design challenges". In: *Proc. 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing*. Orlando, FL, USA, pp. 363–369.

Leva, Alberto (2024). "Teaching PID to future professionals as a contribution to fill a historical gap". In: *IFAC-PapersOnLine* 58.7, pp. 79–84.

Lyu, Guolin and Robert William Brennan (2020). "Towards IEC 61499-based distributed intelligent automation: a literature review". In: *IEEE Transactions on Industrial Informatics* 17.4, pp. 2295–2306.

Miskowicz, Marek (2018). *Event-based control and signal processing*. Boca Raton, FL, USA: CRC press.

O'Dwyer, Aidan (2009). *Handbook of PI and PID controller tuning rules*. Singapore: World Scientific.

Skogestad, Sigurd (2023). "Advanced control using decomposition and simple elements". In: *Annual Reviews in Control* 56, 100903:1–100903:44.

Wiesmayr, Bianca, Sven Mehlhop, and Alois Zoitl (2023). "Close enough? Criteria for sufficient simulations of IEC 61499 models". In: *Proc. 19th IEEE International Conference on Automation Science and Engineering*. Auckland, New Zealand, pp. 1–7.

DOI
10.3384/ecp218603

Master controller for offshore wind power and hybrid grids

Rüdiger Franke[1] Marcin Bartosz[2] Rasmus Nyström[3]
ABB, [1,3]{ruediger.franke,rasmus.nystroem}@de.abb.com, [2]marcin.bartosz@pl.abb.com

Abstract

Offshore power generation and transmission requires long subsea cables. When using AC, the inductance of the cables results in reactive power that contributes to the load of the cables and increases the voltage at busbars. At the same time, offshore power systems are becoming increasingly complex. They are evolving from dedicated collector grids and transmission cables for wind farms through shared use of transmission cables by multiple wind farms to hybrid offshore grids that connect wind farms to multiple countries.

This paper discusses operational challenges of offshore grids and how they are solved with model-based master control solving state estimation and optimal power flow problems in real-time.

The optimization model is built with Modelica using the PowerSystems library and FMI for its deployment in the control system. The optimization solver runs model evaluations and sensitivity analyses over different time intervals in parallel.

The Kriegers Flak Combined Grid Solution (KF CGS) serves as example. It is one of the world's first hybrid offshore grids. The methods have been implemented as Master Controller for Interconnected Operations (MIO). After extensive tests with a digital twin prototype, MIO has been deployed in a geo-redundant system with ABB Ability™ OPTIMAX® and System 800xA.

Keywords: Offshore grids, Hybrid Interconnectors, Optimal Power Flow, Modelica, PowerSystems, state estimation, optimal control.

1 Introduction

Offshore power and intercontinental power transmission are playing an important role for establishing climate neutral energy systems. The European Wind Power Action Plan sets ambitious goals for new installations. The North Sea Energy Consortium alone heads for at least 260 GW of offshore wind energy by 2050, with intermediate targets of at least 193 GW by 2040 and 76 GW by 2030, starting from 33 GW in 2023 (European Commission, 2023).

In December 2020, the Kriegers Flak Combined Grid Solution (KF CGS) was inaugurated by the transmission system operators 50Hertz, Energinet, the German Federal Minister of Economics and the Danish Minister for Climate, Energy and Utilities. It transports wind power from the four offshore wind farms (Baltic 1 and 2, Kriegers Flak A and B) with a total capacity of 950 MW to shore. Additionally, it promotes energy trade between Germany and Denmark. At the heart of the connection of these two energy grids through the Baltic Sea there is a Master Controller for Interconnected Operations (MIO).

This paper discusses the methods and technologies employed in the first of its kind master controller for interconnected operation (Marten et al, 2018).

2 Kriegers Flak Combined Grid Solution

The KF CGS connects the transmission cables of the German wind farms Baltic 1 and Baltic 2 with the transmission cables of the Danish wind farms Kriegers Flak A and B. This creates a hybrid offshore grid for the collection of wind power and the exchange of power between the northern and the continental European transmission grids, see Figure 1.

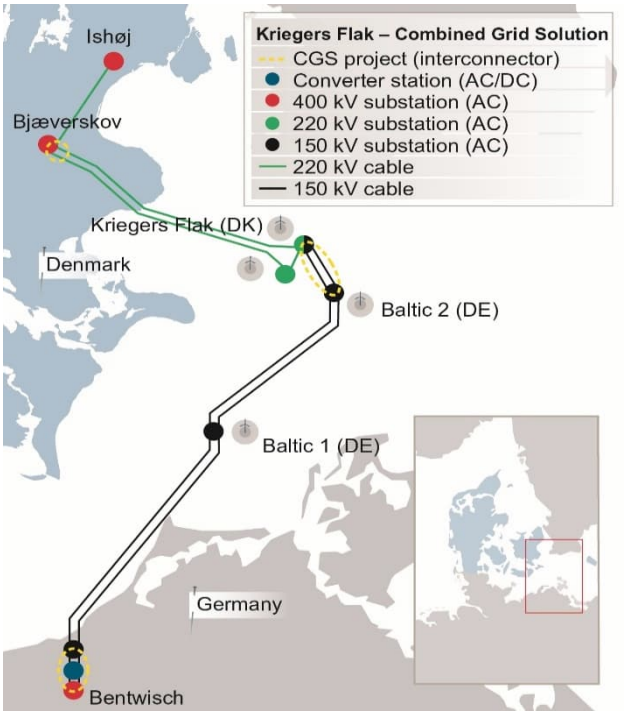


Figure 1: Overview of Kriegers Flak Combined Grid Solution (KF CGS) encircled in yellow. Source: 50Hertz

The KF CGS imposes several operational challenges. A Back-to-Back (BtB) HVDC converter station in Bentwisch enables the connection of the two unsynchronized AC transmission grids. It controls the power exchange at the interconnector in the Baltic Sea by manipulating power flow and voltage from the German coast. The meshed submarine grid interconnection is (n - 0) secure, meaning that a single failure can result in a system outage. Figure 2 shows a single line diagram.

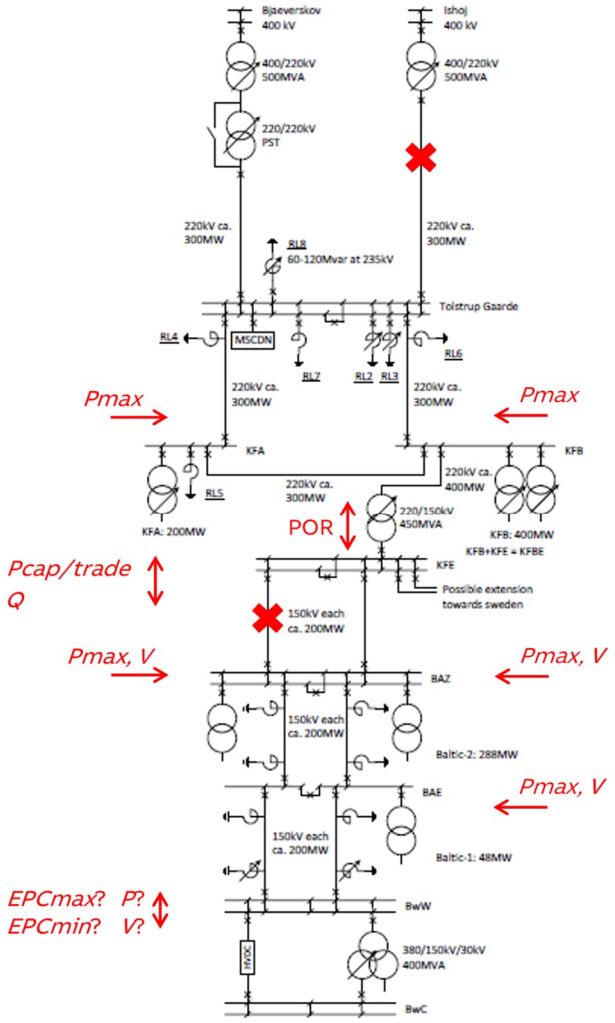


Figure 2: Single line diagram of KF CGS outlining operational challenges. Source: 50Hertz

This results in several tasks for the master controller. First, it provides forecasts for available transfer capacity *Pcap* at the point of reference (POR) using wind forecasts as input. Second, it controls the traded power *Ptrade* and reactive power *Q* in real time, thereby accounting for cable load limits, forecast errors, contingencies and other disturbances in the system. Primarily, the master controller manipulates active power and voltage at the BtB converter. Additionally, it may limit the maximum power Pmax of wind farms. Finally, it provides limits for emergency power control (EPC) in real time.

3 Model-based control with Modelica

Modelica has been widely used for model-based control for many years. Applications range from the treatment of optimal control problems (Franke, 2002) to the creation of inverse models and the generation of embedded controller code (Otter et al, 2012).

Modelica is also used for the modeling and simulation of power systems ranging from microgrids (Fachini et al 2023) to transmission systems (Feghali et al, 2023).

This section outlines the technologies used to implement the master controller for KF CGS.

3.1 PowerSystems library

The Modelica PowerSystems library (R. Franke, H. Wiesmann, 2014) provides a flexible way to model electric power systems at different levels of detail, up to advanced modeling of electromagnetic transients (B. Bachmann, H. Wiesmann, 2000).

This is achieved by different phase systems as shown in the library structure in Figure 3.

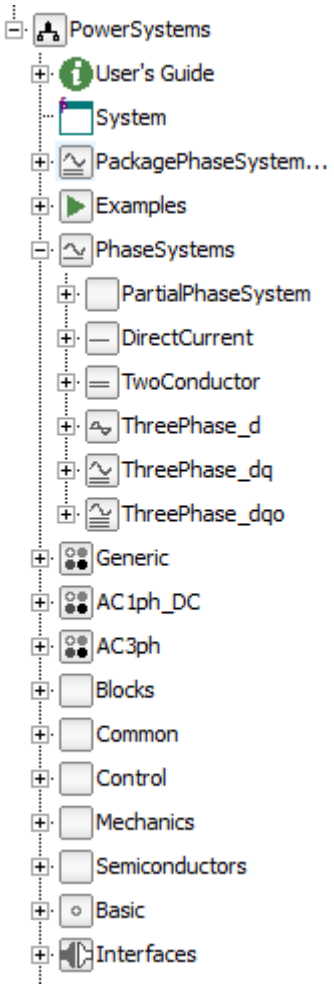


Figure 3: PowerSystems library structure.

DOI 10.3384/ecp218611

Each phase system defines the number `n` of independent current and voltage components, the number `m` of reference angles and appropriate supporting functions like the `j` "operator".

The interfaces define a general power terminal.

```
connector Terminal
  replaceable package PhaseSystem;
  PhaseSystem.Voltage
    v[PhaseSystem.n];
  flow PhaseSystem.Current
    i[PhaseSystem.n];
  PhaseSystem.ReferenceAngle
    theta[PhaseSystem.m]
      if PhaseSystem.m > 0;
end Terminal;
```

Table 1 summarizes the PhaseSystems that are predefined in the PowerSystems library.

Table 1: Phase systems by PowerSystems library.

PhaseSystem	n	m	Description
DirectCurrent	1	0	One voltage and one current component in natural coordinates
TwoConductor	2	0	Two voltage and two current components for Spot AC1ph_DC components
ThreePhase_d	1	0	One modal component for active power — like DirectCurrent, but converting voltage values to three phase
ThreePhase_dq	2	1	Two modal components for active and reactive power; one reference angle for frequency — cf. complex phasors with variable frequency
ThreePhase_dq0	3	2	Three modal components for active, reactive and dc power; two reference angles for transient dq0 components

Note the use of the function j that generalizes complex calculations known from quasi-static AC models to arbitrary phase systems. ThreePhase_dq, with two model components for active and reactive power defines a multiplication with the complex *j*:

```
function j
  input Real x[n];
  output Real y[n];
algorithm
  y := {-x[2], x[1]};
end j;
```

The simpler ThreePhase_d neglecting reactive power defines:

```
function j
  input Real x[n];
  output Real y[n];
algorithm
  y := zeros(n);
end j;
```

The more detailed PhaseSystem_dq0 (direct-quadrature-zero) also considers a component for dc power in asymmetric systems, besides active and reactive power. It defines:

```
function j
  input Real x[n];
  output Real y[n];
algorithm
  y := cat(1, {-x[2], x[1]},
           zeros(size(x,1)-2));
end j;
```

A general base model with two terminals reads

```
partial model PartialTwoTerminal
  replaceable package PhaseSystem =
    PackagePhaseSystem;
  package PS = PhaseSystem;
  function j = PhaseSystem.j;
  Terminal term_p(
    redeclare package PhaseSystem = PS);
  Terminal term_n(
    redeclare package PhaseSystem = PS);
  SI.AngularFrequency omegaRef;
equation
  v = term_p.v - term_n.v;
  i = term_p.i;
  if PS.m > 0 then
    omegaRef =
      der(PS.thetaRef(term_p.theta));
  else
    omegaRef = 0;
  end if;
  Connections.branch(term_p.theta,
    term_n.theta);
end PartialTwoTerminal;
```

A generic steady-state impedance can now be formulated as

```
model Impedance
  extends PartialTwoTerminal;
  parameter SI.Resistance R;
  parameter SI.Inductance L;
equation
  v = R*i + omegaRef*L*j(i);
  zeros(PS.n) = term_p.i + term_n.i;
```

```
  term_p.theta = term_n.theta;
end Impedance;
```

Accordingly, a generic steady-state admittance reads

```
model Admittance
  extends PartialTwoTerminal;
  parameter SI.Conductance G;
  parameter SI.Capacitance C;
equation
  i = G*v + omegaRef*C*j(v);
  zeros(PS.n) = term_p.i + term_n.i;
  term_p.theta = term_n.theta;
end Admittance;
```

A forced switch can be modeled with curve parameter `s`

```
model Switch
  extends PartialTwoTerminal;
  Modelica.Blocks.Interfaces.RealInput
    closed(min=0, max=1);
protected
  Real[PS.n] s;
equation
  {v, i} = {(1-closed)*PS.v_nominal*s,
            closed*PS.i_nominal*s};
end Switch;
```

The package AC3ph provides many relevant component models like Lines, Nodes, Breakers, Shunts and Transformers for the application at hand, see Figure 4.

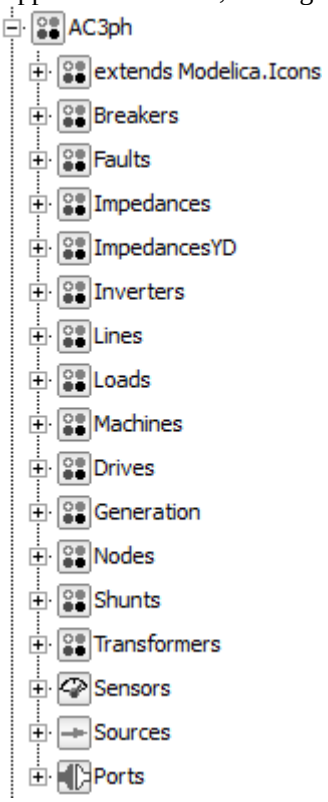


Figure 4: PowerSystems AC3ph component models.

3.2 Dynamic Optimization

Modelica distinguishes between equation-based models and executable simulation code. A model translator generates executable code. Beyond simulation runs in the Modelica tool at hand, the executable code can also be exported to other tools. It has been used for dynamic optimization in several industrial applications, e.g. (Franke, 2002). The introduction of the Functional Model Interface (FMI) and synchronous language features standardized the interface of executable discrete-time models (Franke et al, 2017).

Alternatively, (Åkesson et al, 2010) proposed a syntactical extension of the equation-based Modelica language to Optimica for optimization. Recent work couples Modelica with Matlab to implement model predictive control (Alizadeh et al, 2023).

Numerical solution as large-scale nonlinear program
The nonlinear solver HQP used by ABB OPTIMAX® treats discrete-time optimal control problems as large-scale mathematical programs. Discrete-time states and controls are collected in the state vector x and the control vector u, respectively. This gives the discrete-time optimal control problem:

$$J = f_0(x^K) + \sum_{k=0}^{K-1} f_0(x^k, u^k) \quad \rightarrow \quad \min_{x^0,\, u^k}$$

with the discrete-time state equations

$$x^{k+1} = f^k(x^k, u^k),$$
$$y^k = g^k(x^k, u^k), \quad k = 0, \dots, K-1$$

and the constraints

$$c^k(x^k, u^k) \geq 0, \quad k = 0, \dots, K-1$$
$$c^K(x^K) \geq 0$$

(1)

The executable model imported as Functional Model Unit (FMU) for model exchange implements the state equations f^k. It is important to note that the explicit relationship between f^k and x^{k+1} results in linear optimization constraints that can be exploited for efficient parallel optimization. The model can be evaluated in parallel in each time step. Additionally, the FMU provides optimization objective functions f_0 and the constraints c^k as outputs.

The states and the control inputs of all time intervals are collected into one large vector of optimization variables

$$v = (x^0, u^0, x^1, u^1, \dots, x^{K-1}, u^{K-1}, x^K).$$

(2)

DOI 10.3384/ecp218611

This results in the mathematical program

$$\begin{aligned} J(v) &\underset{v}{\rightarrow} min & J&: \mathbb{R}^{n_v} \rightarrow \mathbb{R}^1 \\ h(v) &= 0 & h&: \mathbb{R}^{n_v} \rightarrow \mathbb{R}^{m_e} \\ g(v) &\geq 0 & g&: \mathbb{R}^{n_v} \rightarrow \mathbb{R}^{m_i} \end{aligned} \tag{3}$$

HQP treats large-scale nonlinear optimization with Sequential Quadratic Programming (SQP).

Basing on the Lagrangian

$$\begin{aligned} L(v,\lambda,\mu) &= J(v) - \lambda^T h(v) - \mu^T g(v) \\ L&: \mathbb{R}^{n_v} \times \mathbb{R}^{m_e} \times \mathbb{R}^{m_i} \rightarrow \mathbb{R}^1 \end{aligned} \tag{4}$$

the solution must fulfill the Karush Kuhn Tucker (KKT) conditions

$$\begin{aligned} \nabla_v L(v,\lambda,\mu) &= \nabla J(v) - \nabla h(v)^T \lambda - \nabla g(v)^T \mu = 0 \\ \nabla_\lambda L(v,\lambda,\mu) &= -h(v) = 0 \\ g(v) &\geq 0 \\ \mu &\geq 0 \\ g(v)^T \mu &= 0 \end{aligned} \tag{5}$$

HQP applies Lagrange Newton iterations

$$\begin{aligned} \nabla^2 L(v,\lambda) \begin{pmatrix} \Delta v \\ \Delta\lambda \end{pmatrix} &= -\nabla L(v,\lambda) \\ \begin{pmatrix} v^+ \\ \lambda^+ \end{pmatrix} &:= \begin{pmatrix} v + \Delta v \\ \lambda + \Delta\lambda \end{pmatrix} \end{aligned} \tag{6}$$

to find the solution. The Lagrange Newton iteration is given here for the case $m_i=0$. HQP augments the Lagrangian to treat inequality constraints with an Interior Point method. The Jacobian $\nabla J(v)$ and the Jacobian matrices $\nabla h(v)$ and $\nabla g(v)$ are formed analytically exploiting directional derivatives provided by the FMU per time interval. The Jacobians of each time step are collected into large sparse matrices for the overall dynamic optimization problem giving

$$\nabla h(v) = \begin{pmatrix} \frac{\partial f^0}{\partial x^0}\frac{\partial f^0}{\partial u^0} & -I & & \\ & \ddots & & \\ & & \frac{\partial f^{K-1}}{\partial x^{K-1}}\frac{\partial f^{K-1}}{\partial u^{K-1}} & -I \end{pmatrix} \tag{7}$$

and

$$\nabla g(v) = \begin{pmatrix} \frac{\partial \bar{c}^0}{\partial x^0}\frac{\partial \bar{c}^0}{\partial u^0} & & & \\ & \ddots & & \\ & & \frac{\partial \bar{c}^{K-1}}{\partial x^{K-1}}\frac{\partial \bar{c}^{K-1}}{\partial u^{K-1}} & \\ & & & \frac{\partial \bar{c}^K}{\partial x^K} \end{pmatrix} \tag{8}$$

The Hessian of the Lagrangian $\nabla^2 L(v,\lambda,\mu)$ is formed numerically applying a rank 2 update in each time interval based on the progress over subsequent iterations. As the linear coupling between subsequent time intervals plays no role for second order derivatives in the Hessian, the large-scale nonlinear program is partial separable. This allows an efficient multi-rank update of the large Hessian of the Lagrangian. This is also why no analytical second order derivatives are required.

4 Application to KF CGS

A model of the Kriegers Flak Combined Grid Solution was implemented in OpenModelica using the PowerSystems library. For the operational optimization considered here, we assume symmetric load of the three phases and quasi-stationary behavior. This is achieved with the phase system ThreePhase_dq. Once selected globally, the connectors and most components adjust to it automatically, for insance cables, shunts and breakers. Some components, such as inverters, have multiple implementations in the PowerSystems library. The simplified InverterAverage uses time-averaged variables, whereas the rigorous Inverter model implements fast switching in the time domain. The phase system ThreePhase_dq0 would cover transient scenarios and unsymmetric loads at the cost of larger models involving fast dynamics.

The model is then exported as FMU for model exchange with the C++ runtime. Several activities are configured for state estimation, predictive planning optimization and real-time optimization.

Figure 5 shows a screenshot of the model. To keep the size of the model diagram better manageable, some details were encapsulated into sub models. Particularly a generic AC transmission line was formed from a π-line combined with an optional shunt and an optional switch at each end, see Figure 6.

The KF CGS model was tested in OpenModelica. It consists of 4722 variables and equations. Many of them are trivial equalities going back to connections between component models.

The translator identifies 788 non-trivial equations, 144 model inputs and 335 model outputs. For dynamic optimization, these numbers multiply with the number of time steps K, such as K=288 for three days with a step size of 15 minutes.

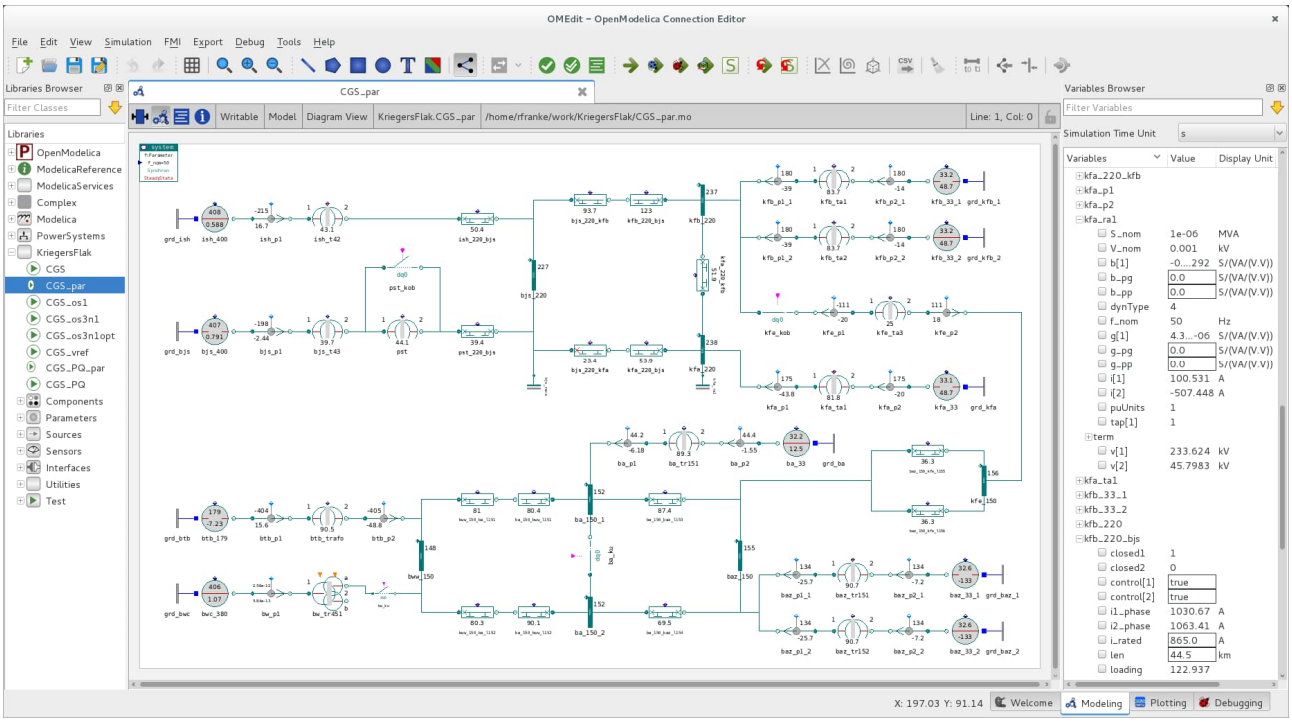


Figure 5: Screenshot of OpenModelica with KF CGS model.

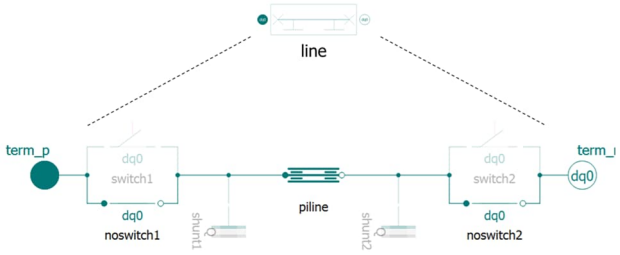


Figure 6: Generic sub model of an AC line with optional shunts and switches. Model icon at the top, model diagram at the bottom.

The variables relevant for optimization objective and constraints are quadratic functions of the internal voltage and current phasors. This gives active power and reactive power

$$P = v \cdot i$$
$$Q = -j(v) \cdot i \qquad (9)$$

along with absolute voltages, e.g. measured at busbars, and absolute currents, e.g. determining cable loads

$$V = \sqrt{v \cdot v}$$
$$I = \sqrt{i \cdot i} \qquad (10)$$

The steady-state system model depicted in Figure 5 leads to algebraic loops. Tearing reduces mutual dependencies to one linear equation system of dimension 81. The variables of this equation system are curve parameters `s` of switches at the lines, besides currents i and voltages v of transformers, see Section 3.1. The system is linear thanks to the use of rectangular dq coordinates.

Overall, this gives a quadratically constrained quadratic optimization problem that allows a fast and robust numerical solution.

Figure 7 shows the principle of the closed loop control. Measurements are absolute voltages, currents and power flows. Tuned model inputs u are voltage phasors in dq coordinates. Optimized variables y are set points for the BtB converter and wind farms.

The translated model is deployed with ABB OPTIMAX® in a geo-redundant setup. Several optimization activities are running using multiple instances of the same model.

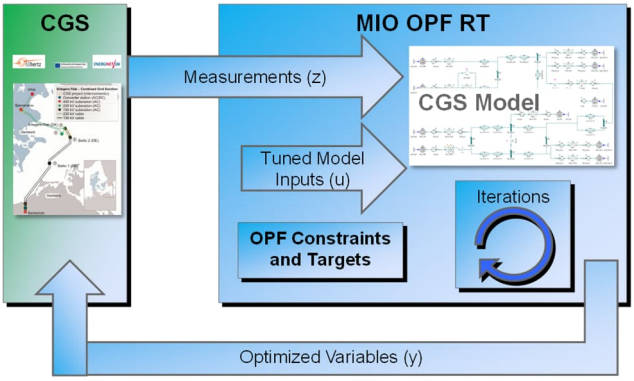


Figure 7: Principle of closed loop control with optimal power flow (OPF).

DOI 10.3384/ecp218611

Table 2: Speedup obtained with parallel optimization and different solver configurations.

Speedup for different solver configurations	Finite Differences	AD with refactoring	AD with factor reuse
Sequential shooting, Dense model blocks, no AD, 1 CPU	1.0		
Sequential shooting, Sparse model blocks, 1 CPU	1.9	1.3	3.9
Parallel multiple shooting, Sparse model blocks, 2 CPUs	2.8	2.2	4.6
Parallel multiple shooting, Sparse model blocks, 5 CPUs	4.6	3.7	6.3
Parallel multiple shooting, Sparse model blocks, 20 CPUs	7.0	6.5	7.6

4.1 Planning Optimization

Based on given wind power forecasts and maintenance schedules, the maximum remaining power transfer capacities from Denmark to Germany and from Germany to Denmark are obtained, maximizing the power flow through the system subject to utilization of wind, constraints on cable loadings and limits on busbar voltages.

The planning optimization turns out to be the most time critical task as the coverage of a time horizon of up to three days with a resolution of 15 minutes results in a large-scale dynamic optimization problem. The formulation as discrete-time optimal control problem (1) with the large vector of variables (2) allows the separation of the time horizon and the evaluation of the model in parallel for each time step. The method is known as Multiple Parallel Shooting.

Table 2 summarizes results obtained on a 20-core high performance server. A speedup of 1.9 is achieved by not only exploiting sparsity from the time staggered structure of the discrete-time optimal control problem, but also sparsity of Jacobian matrices at each time step as given in the modelDescription.xml file of the FMU. Coloring in particular leads to a reduction of the number of directional derivatives required to obtain the whole Jacobian matrix. Interestingly, Automatic Differentiation (AD) turned out slower than Finite Differences initially. A speedup to 3.9 is achieved when re-using factors of the 82-dimensional linear equation system for subsequent determinations of directional derivatives.

The solution time can be further reduced and a speedup of up to 7.6 is achieved by increasing the number of CPU cores to 20. The speedup of 7.0 for Finite Differences is almost as high with 20 CPU cores. This is achieved by higher utilization of parallel cores for repeated factorizations of the linear equation system.

Automatic Differentiation with re-use of factors achieves a high speedup of 6.3 already with 5 CPU cores. This leaves the remaining cores for other tasks, like the solution of two capacity planning problems towards Denmark and towards Germany at the same time.

4.2 Real-time Optimal Control

Uncertainties of wind forecasts, the intermittent nature of wind as well as possible disturbances are addressed by re-optimizing set points in closed loop control during the actual operation. Additionally, optimization is employed to determine limits for the underlying control during a possible emergency. Limits for power transfer change dynamically with the infeed of wind as well as with the switching structure of cables, shunts and transformers.

State estimation: existing measurements of absolute voltages and currents are used to estimate system wide voltages and currents in dq coordinates. This is achieved by minimizing the residuals between measurements of absolute quantities and respective model outputs. From a control point of view, the electrical system is in steady-state. Dynamic states arise from rate-of-change bounds of controlled electrical assets.

Real-time optimization: starting from the estimated state, optimal power and voltage set points for the BtB converter and possibly required curtailments of wind farms are determined every 5 seconds. The underlying optimal power flow problem maintains the given target value *Ptrade* at the point of reference while minimizing losses in the system subject to constraints on cable loadings and busbar voltages.

Emergency Power Control (EPC): starting from the estimated state, the maximum power transfer capacities towards Denmark and towards Germany are determined subject to constraints on cable loadings and busbar voltages in real-time and communicated as EPC limit to the BtB converter.

4.3 Digital Twin Prototype

A digital twin was employed during the engineering of MIO. It comprised the FMI model, the OPTIMAX® runtime and the communication protocol along with existing grid simulation software of 50Hertz and Energinet. This made it possible to test a vast number of operational scenarios with simulation prior to the commissioning of the actual system. It was key to shortening the commissioning time and for ensuring the reliable operation of this complex offshore grid.

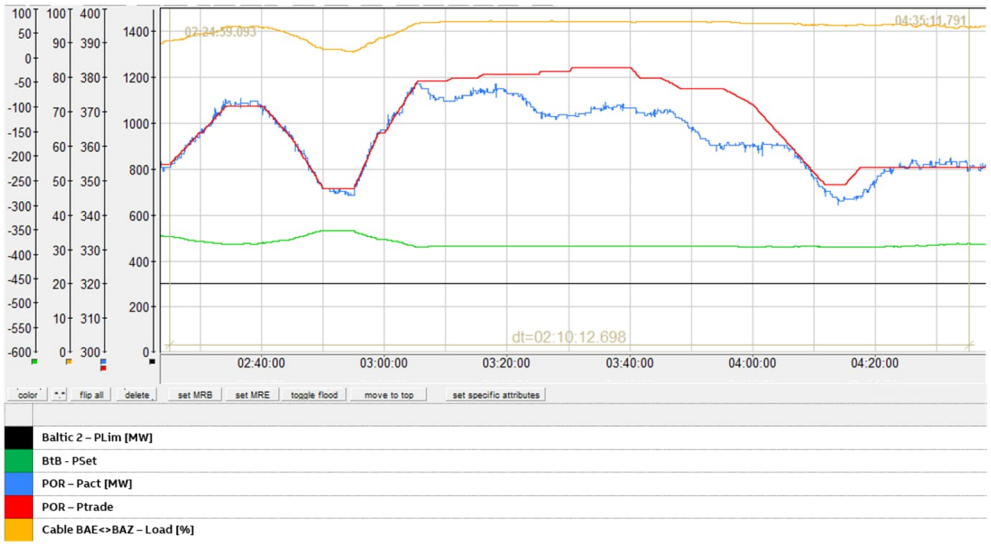


Figure 8: Real-time optimization: Overseeing planned energy transfer, respectful of network constraints.

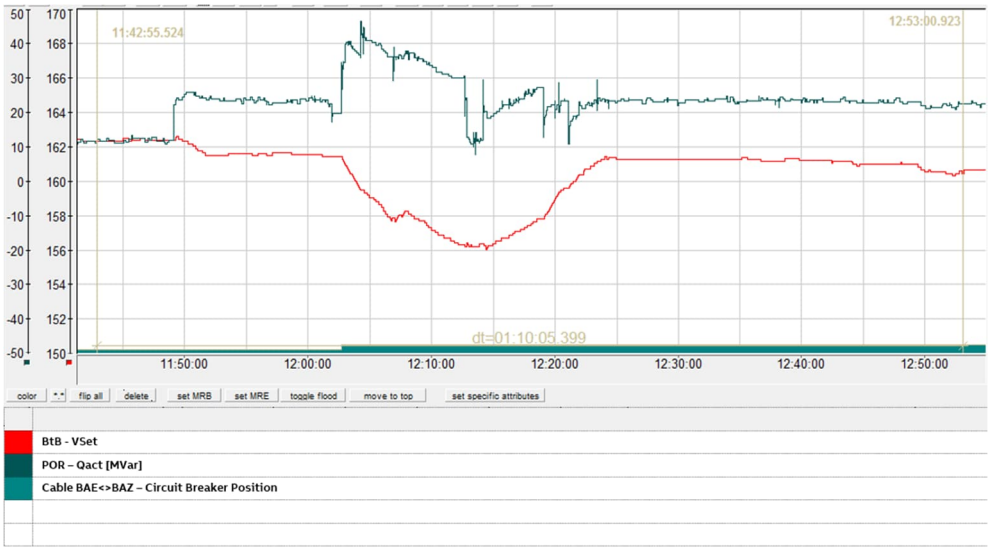


Figure 9: Real-time optimization: BtB voltage control and POR reactive power constraints.

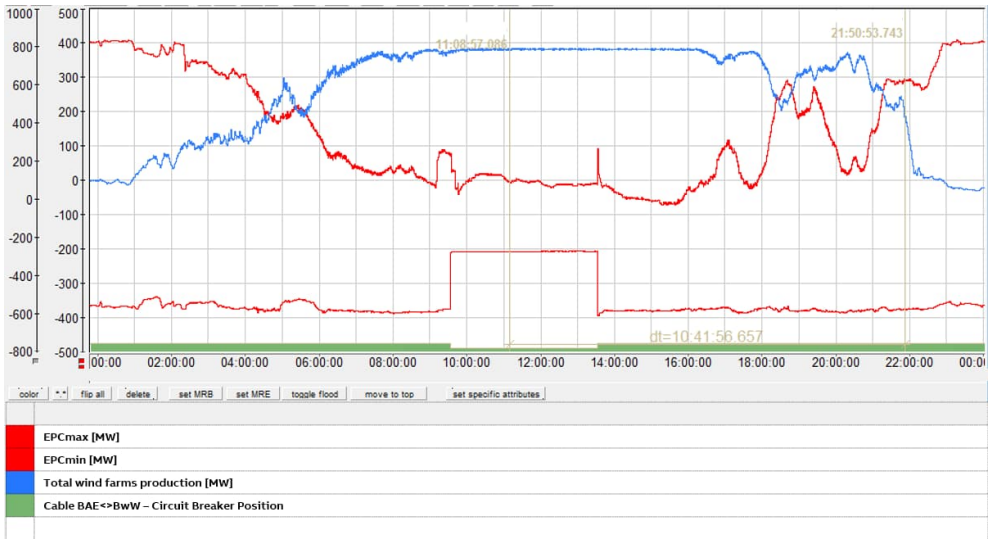


Figure 10: Emergency Power Control: Adjustment of the safe operating limit considering operational conditions.

DOI
10.3384/ecp218611

5 Operational Examples

This section discusses several examples of the actual operation of the Kriegers Flak hybrid offshore grid with real-time optimal power flow, maintaining planned power transfers while prioritizing wind power and considering limits on cable loadings. See also the single line diagram in Figure 2 for relating acronyms used in this section to the overall system.

The plots show active power in MW, reactive power in Mvar, voltage in kV, line loading in % and switch position as 0 (off) or 1 (on).

Real-time optimization: Figure 8 shows an example, where power transfer is optimized. MIO manipulates the BtB such that the actual power transfer via POR follows the planned transfer until 03:05. Then the load on cable BAE—BAZ reaches its limit as the actual wind power is higher than expected during the planning. The wind farm Baltic 2 is not curtailed. Instead, MIO manipulates the BtB to maintain the cable load limit, resulting in lower than planned power transfer until 4:20.

Figure 9 illustrates BtB voltage control during the connection of the BAE—BAZ cable at 12:03. MIO manipulates the BtB to limit the reactive power flow through POR by reducing the voltage. Simultaneously, the underlying control system adjusts transformer tap positions and reactive power compensators. This is why the BtB can return to a higher voltage until 12:24.

Emergency Power Control: Figure 10 illustrates the adaptive EPC limits responding to dynamic variations in wind power and contingencies. Starting from 01:00, MIO reduces *EPCmax* gradually to accommodate for increasing wind power. Additionally, unforeseen contingencies are directly impacting power transmission capabilities. This is seen as example by the loss of the BAE—BwB cable at 09:35. Consequently, MIO reduces *EPCmin* until 13:30

6 Conclusions

The increasing installation of offshore wind power results in more demanding power collection and transmission tasks. Multiple wind farms get connected to offshore grids. Hybrid offshore grids connect the wind farms to multiple countries and thus serve for power transmission between those countries as well. This increases the operational complexity.

This paper shows how hybrid offshore grids can be operated with model-based nonlinear control, solving optimal power flow problems in real-time. The methods were implemented in the Master Controller for Interconnector Operation (MIO) for the Kriegers Flak Combined Grid Solution (KF CGS). KF CGS connects multiple wind farms in the Baltic Sea to the northern and continental European transmission grids. MIO was tested extensively with simulations using a digital twin prototype and has been in successful operation in the real system since December 2020.

As a result, the transmission of power through offshore grids is maximized, complex flexibilities arising from connections to multiple wind farms and multiple countries are exploited and the reliability is improved.

The formulation of the electrical models in dq coordinates results in linear equation systems for voltage and current phasors with quadratic output relationships for absolute voltages, e.g. as measured at busbars, and currents, e.g. determining cable loads. The absolute quantities are constrained in the optimal power flow problem. This results in a quadratically constrained quadratic optimization problem, allowing a reliable and fast numerical solution for the application in closed loop real-time control.

Significant speedups are achieved with parallel multiple shooting in dynamic optimization problems for the planning of the operation. Parallel automatic differentiation and optimization had been addressed in the PARADOM project.

Current research investigates the extension of future offshore installations with the production of green hydrogen and the use of gas pipelines complementing electric cables. The OpenSCALING project addresses the treatment of large-scale green hydrogen production with Modelica.

Acknowledgements

This work was supported in parts by the Federal Ministry of Education and Research (BMBF) within the projects PARADOM and OpenSCALING – BMBF funding codes: 01IH15002E and 01IS23062B, respectively.

References

European Commission: European Wind Power Action Plan, Brussels, October 2023. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A52023DC0669&qid=1702455143415

A. Marten, V. Akmatov, T. Bentzon Sorensen, R. Stornowski, D. Westermann, C. Brosinsky: Kriegers flak combined grid solution: coordinated cross-border control of a meshed HVAC/HVDC offshore wind power grid, IET Renewable Power Generation, 2018, Vol. 12 Iss. 13, pp. 1493-1499. https://ietresearch.onlinelibrary.wiley.com/doi/epdf/10.1049/iet-rpg.2017.0792

R. Franke: Formulation of dynamic optimization problems using Modelica and their efficient solution, 2nd International Modelica Conference, 2002. https://modelica.org/events/Conference2002/papers/p39_Franke.pdf

M. Otter, B. Thiele, H. Elmqvist: A Library for Synchronous Control Systems in Modelica, 9th International Modelica Conference, Munich, 2012. https://www.ep.liu.se/ecp/076/002/ecp12076002.pdf

F. Fachini, S. Bhattacharjee, M. Aguilera, L. Vanfretti, G. Laera, T. Bogodorova, A. Moftakhar, M. Huylo, A. Novoselac: Exploiting Modelica and the OpenIPSL for University Campus Microgrid Model Development, 15th International Modelica Conference, Aachen, October 2023. https://ecp.ep.liu.se/index.php/modelica/article/view/936/844

J.E. Feghali, Q. Cossart, G. Bureau, B. Letellier, I. Menezes, F. Rosiere, M. Chiaramello: An Open-Source Benchmark of IEEE Test Cases for Easily Testing a New Approach for Steady State Calculations in Power Systems, 15th International Modelica Conference, Aachen, October 2023. https://ecp.ep.liu.se/index.php/modelica/article/view/974/882

B. Bachmann, H. Wiesmann: Advanced Modeling of Electromagnetic Transients in Power Systems, *Modelica Workshop 2000*, Lund, September 2000. https://modelica.org/events/workshop2000/proceedings/Bachmann.pdf

R. Franke, H. Wiesmann: Flexible modeling of electrical power systems – the Modelica PowerSystems library, 10th International Modelica Conference, Lund, 2014. https://www.ep.liu.se/ecp/096/054/ecp14096054.pdf

R. Franke, S.E. Mattsson, M. Otter, K. Wernersson, H. Olsson, L. Ochel, T. Blochwitz: Discrete-time models for control applications with FMI, 12th International Modelica Conference, Prague, 2017. https://www.ep.liu.se/ecp/132/057/ecp17132507.pdf

J. Åkesson, K.-E. Årzén, M. Gäfvert, T. Bergdahl, H. Tummescheit. Modeling and optimization with Optimica and JModelica.org—languages and tools for solving large-scale dynamic optimization problems. Computers and Chemical Engineering, 34(11):1737–1749, November 2010.

M. H. Alizadeh, A. M. Sahlodin, A. Palanisamy, F. Casella, P. Fritzson: Application of the OpenModelica-Matlab Interface to Integrated Simulation and Successive Linearization Based Model Predictive Control, 15th International Modelica Conference, Aachen, October 2023. https://ecp.ep.liu.se/index.php/modelica/article/view/937/845

 DOI 10.3384/ecp218611

The FlightControl library for aircraft control design applications

Reiko Müller[1]

[1]DLR Institute of Flight Systems, Oberpfaffenhofen, Germany
reiko.mueller@dlr.de

Abstract

Controller development of aerial vehicles is a long-standing task during aircraft design and has a large impact on the resulting systems performance. In today's integrated design loops, ideally all components of the aircraft must be considered in simulation and testing in order to develop complex system architectures and meet performance requirements. One of the tools that are suited for modeling and simulation of a multidisciplinary aircraft and systems assembly is DLR's FLIGHTDYNAMICS Library, which takes advantage of the capabilities of the MODELICA modeling language. In this work, a new augmentation to the library is discussed, which implements a range of common aircraft control concepts which can be used for design of new controllers, closed-loop simulation and experimental testing via code-export. The library is set up in a modular way, so that flight guidance and flight control systems can be developed for multiple aerial platforms, including manned/unmanned fixed- and rotary-wing aircraft. For this paper, a simulation example is provided by means of an autoland controller that shall be designed for a high-fidelity 6-DoF fixed-wing aircraft model in MODELICA. The combination of aircraft and controller models is subjected to a three step synthesis process, which yields a controller that is robust against external and internal disturbances.

Keywords: *Flight control, flight dynamics, design optimization*

1 Introduction

As was laid out in previous publications (Looye 2008; Looye et al. 2014), the DLR FLIGHTDYNAMICS library is a well established tool for aircraft system design and simulation in the language MODELICA. It can be employed in a multitude of applications, spanning from systems design / Multidisciplinary Design Optimization (MDO) (Liersch and Hepperle 2011), trajectory optimization (Müller 2021), mission simulation as well as reliability and validation studies. Many of these setups require some sort of flight controller and / or flight guidance functions in order to adhere to the respective simulation scenarios. Therefore, an effort was made to bring flight controllers and flight guidance functionality to the library / to MODELICA, where previous approaches relied upon model export (i.e., via Functional Mockup Interface (FMI)) and usage of other tools for the controller design. Hence, this paper exemplifies the usage of the flight controller sub-library to relevant aerospace applications.

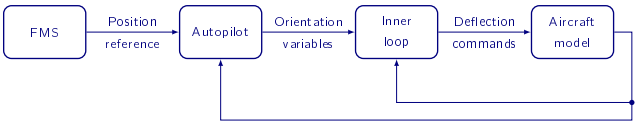


Figure 2. Standard cascaded flight controller layout: with flight guidance/trajectory generation (FMS) → Trajectory tracking (autopilot) → Tracking of aircraft orientation (Inner loop control).

In Section 2, the overall structure of the sub-library is laid out. Section 3 covers the flight guidance functions (e.g. Flight Management System (FMS)). The actual flight controllers are discussed in Section 4, separated by the respective control task and design methodology. The application example covers the robust design of an autolanding controller and is laid out in Section 5.

2 Library structure

The FLIGHTCONTROL library was structured according to the classical perceiption of fixed-wing aircraft flight control where the control variables are associated with cascaded layers of tasks. This is possible due to the time-scale separation principle: the inner layer has the fastest dynamics, which allows to assume constant command inputs from the outer layer, where in turn, from the outer layer perspective, the reaction of the inner layer to commands are assumed to be instantaneous, see Figure 2. This separation is reflected in the library with subpackages for guidance (FMS), autopilot (shown in Figure 1) and inner loop controllers. Another distinction is made between classical flight control concepts relying on linear PID control and approaches containing more model information (like Nonlinear Dynamic Inversion (NDI) controllers) which are both collected in separate subpackages. The examples shown in the paper are of the first type, as they are better explainable with regards to flight physics as some of the more elaborated counterparts.

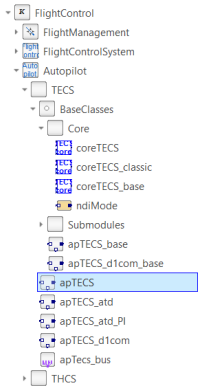
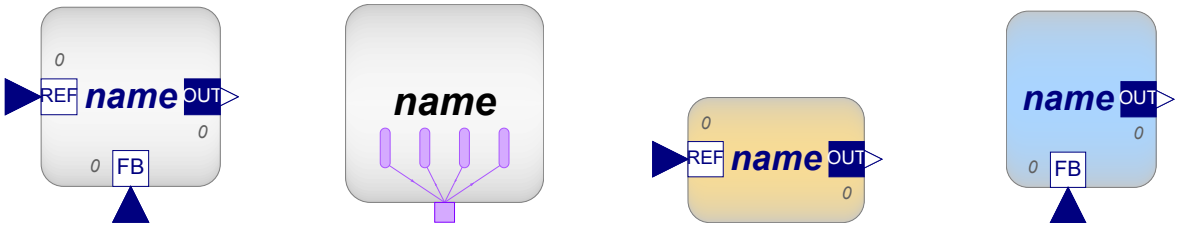


Figure 1. Opened autopilot package.



(a) The partial controller class. **(b)** The partial controller class using expandable bus inputs/outputs. **(c)** Partial guidance base class. **(d)** The partial computation base class.

Figure 3. Partial models that are used to discern between blocks with/without reference/feedback inputs, all blocks have an optional boolean input for integrator reset (not shown).

2.1 Interconnection with other libraries

As mentioned before, the FLIGHTCONTROL library lends itself to be combined with the DLR FLIGHTDYNAMICS library for easy integration with aircraft models or derivation of model-based controllers. With a clear I/O concept based on standard connectors and buses and a variable naming based on common standards (e.g. LN 9300, ISO 1151-1), it is also easy to integrate the flight controller library with other MODELICA based libraries for aircraft simulation. The base class models for controller blocks are shown in Figure 3. In addition, usage in form of exported models is possible, for example as Functional Mockup Unit (FMU) that can nowadays be integrated in a multitude of tools and languages (for example MATLAB®/SIMULINK®, PYTHON, etc.).

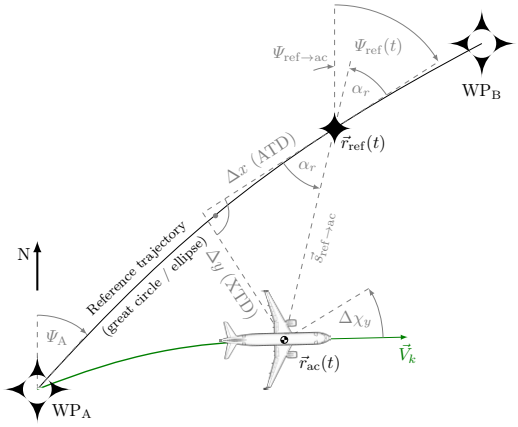


Figure 4. Variables for lateral tracking of a 4-D trajectory.

3 Guidance functions

The guidance module or often called FMS provides reference values for the downstream Flight Control System (FCS). While not being dependent on feedback in the sense of a controller, the FMS still needs information from the aircraft, for example to calculate a trajectory based on the current aircraft state or to provide switching of controller / autopilot modes. This is also reflected in the library, where models exist for 4-D continuous trajectory generation, and also a mode-based FMS relying on a state machine formulation with MODELICA STATEGRAPH2. Apart from this, the guidance focuses on the system kinematics, which are model - independent. For example the reference velocity $\vec{V}_k$ can be calculated from a trajectory given in ellipsoidal coordinates (φ, λ, h) by:

$$\vec{V}_k = [v_N, v_E, v_D]^\text{T} = \begin{bmatrix} (R_n + h)\cdot\dot{\varphi} \\ (R_e + h)\cdot\dot{\lambda}\cos\varphi \\ -\dot{h} \end{bmatrix}, \qquad (1)$$

with R_e, R_n as Earths radii of curvature. From this the usual reference variables V_k (total inertial speed), γ_k (trajectory pitch angle) and χ_k (course angle) can be calculated. If there is feedback from the aircraft, the inputs to e.g., 4-D trajectory tracking can also be provided, which would be the Along Track Distance (ATD) (for controlling time), as well as Cross-Track Distance (XTD) and altitude (for control of lateral / vertical offset), see Figure 4. The FMS can also be a part of a larger block involving an optimization to provide optimal trajectory guidance. In this case, the trajectory is parameterized (for example by polynomials/splines) and models a position (and time) trajectory that can be adjusted by an optimization algorithm to fulfill constraints and / or minimize objective functions (for details see (Müller 2020; Müller 2021)).

4 Flight control system

As mentioned in the introduction, for a cascaded flight controller, a separation can be made with regards to the time-scales of the control variables, and to the degrees of freedom of the aircraft.

4.1 Autopilot

As the outer loop controller, the autopilot keeps the aircraft on the desired trajectory specified by the FMS, usually in the variables altitude h, velocity V, heading angle Ψ as well as track length or time reference (sometimes sideslip angle β is also directly controlled, e.g. to facilitate cross-wind landings.). A well known and widely adopted autopilot controller for the longitudinal motion of the aircraft is the Total Energy Control System (TECS), see (Lambregts 2013a) for details. Considering the equilibrium of potential and kinetic energy of the aircraft,

DOI 10.3384/ecp218621

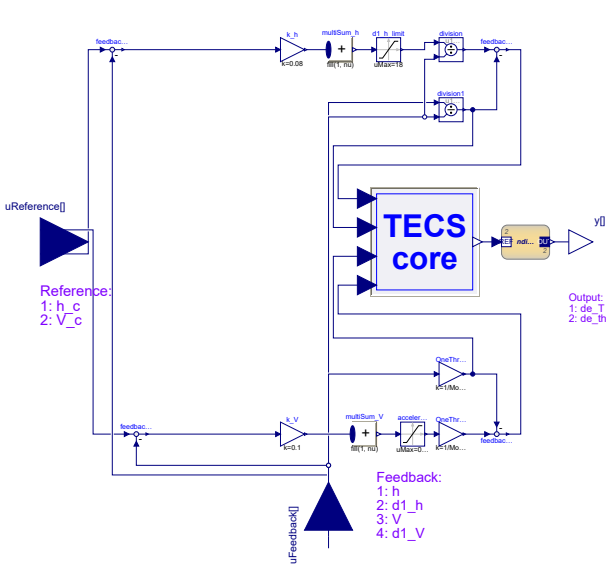


Figure 5. Realization of the Total Energy Control System (TECS). For 4-D trajectory tracking this controller serves as base class and is augmented with control of the ATD. By keeping ATD to zero, it is ensured that the aircraft will not fly ahead or behind the reference time-wise.

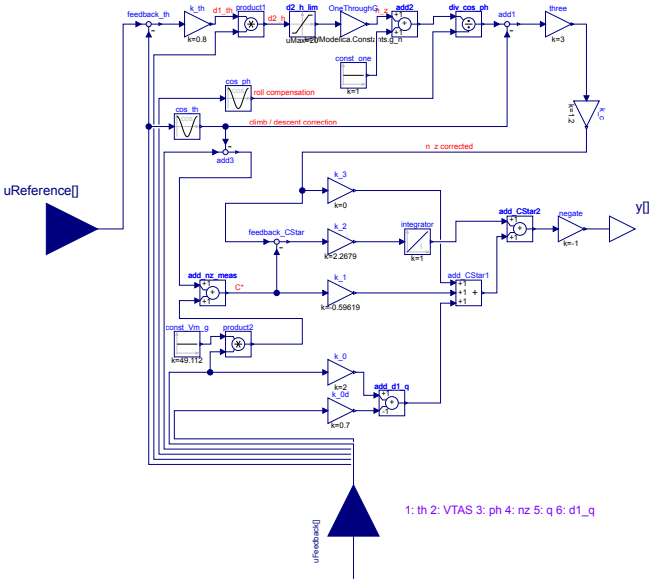


Figure 6. Aircraft pitch controller implementing the $C*$ - criterion for control performance and passenger comfort.

it builds upon the longitudinal force equation of a mass point:

$$m\dot{V}_k = T - D - mg \cdot \sin\gamma_k. \tag{2}$$

When assuming trimmed horizontal flight, errors in $\dot{V}_k$ and γ_k are cancelled with thrust and pitch commands δT and $\Delta\Theta$, working as energy generating and energy distributing control respectively. This coupling hence allows to control both channels in one block. An implementation using the FLIGHTCONTROL library is shown in Figure 5. A similar concept for the lateral aircraft motion was devised in the form of Total Heading Control System (THCS) (Lambregts 2013b), which is also implemented in the library but not shown here.

4.2 Inner loop

For the inner loop controller (sometimes denoted as Stability and Control Augmentation (SCA)), a separation into the roll/pitch/yaw channels (and thrust) can be made for fixed wing aircraft, as their longitudinal and lateral movement are usually not tightly coupled. The task of an inner loop controller is to increase mode damping and stabilization by generating appropriate control surface deflection commands (aileron, elevator, rudder) out of the reference inputs (e.g., commanded roll-, pitch- and yaw rates p_c, q_c, r_c) received from the autopilot. A number of controller architectures are possible here, implementing for example classical linear PID controllers (see e.g., (Brockhaus, Alles, and Luckner 2011; Stevens, Lewis, and Johnson 2015)) and model-based (e.g., dynamic inversion) approaches.

As a concrete example for an inner loop controller implemented in the library, see Figure 6. It represents a pitch controller, implementing the well-known C^* - criterion, where:

$$C^* = \frac{1}{g}\left(-\ddot{z} + x_{\text{pilot}}\dot{q} + V_m q\right) \tag{3}$$

is a weighting between vertical acceleration at the pilot position ($-\ddot{z} + x_{\text{pilot}}\dot{q}$) and pitch rate q. Since pilots judge aircraft reaction at high speeds depending on the former and at low speeds on the latter, this variable is well suited as pitch control variable throughout the flight envelope. The upper part converts pitch- to vertical load factor commands (also corrected for turning flight), which is then compared with the measured acceleration. Adaptation to an aircraft model can then be achieved by adjusting the five gains aligned in the middle, e.g. via pole placement and using aircraft derivatives obtained from linearization (which usually has to be done for a multitude of operating points in the flight envelope). An alternative to this for robust controller synthesis is an iterative process consisting of worst-case search across the envelope, multi-case optimization and Monte-Carlo analysis, which is shown in Section 5.

In addition to these structured flight control concepts, Multiple-Input Multiple-Output (MIMO) controllers can also be easily integrated, for example by using appropriate classes from the LINEARSYSTEMS2 library. Owing to the acausal modeling feature of MODELICA, a long tradition of using inverse models within the FLIGHTDYNAMICS library is preserved here in several controller implementations using concepts like NDI (Enns et al. 1994). All of the mentioned approaches however require model information, which is not included in the FLIGHTCONTROL library and has to be provided separately.

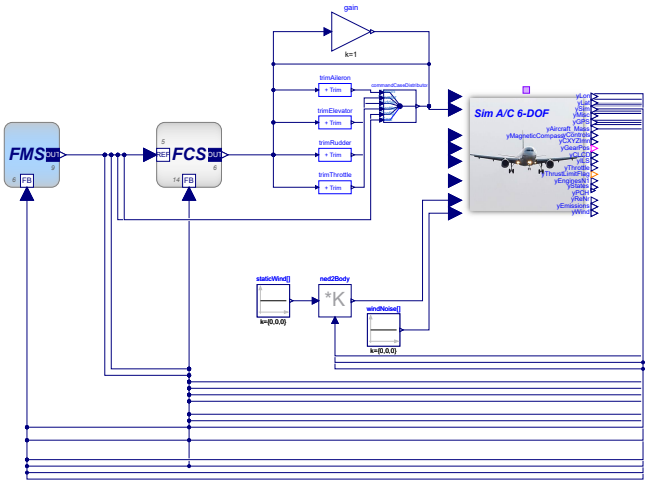


Figure 7. Assembly of FMS and Flight Control System (FCS) with aircraft model.

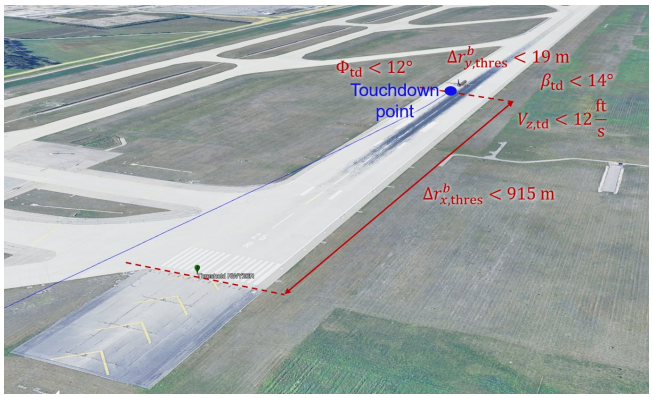


Figure 8. Touchdown criteria as specified by the CALC document.

4.3 Special control functions

The library also contains functions that are not part of primary flight control. This includes for example controllers for lateral and vertical gust load alleviation, which were deployed in (Müller and Ritter 2017). Controllers implementing Pseudo Control Hedging are for example important in trajectory optimization where non-feasible/flyable trajectories must be hedged with respect to different variables (mostly thrust). This is also crucial for simulation of High Altitude Pseudo Satellite (HAPS) aircraft with their very narrow velocity envelope, as was shown in (Müller, Kiam, and Mothes 2018). Related to landing, there are controllers for high-lift systems, which are based on various available specifications by manufacturers. Regarding autolanding, the library also contains controllers to follow the glideslope and glidepath, for manageing flare, groundroll and taxi. In the following example scenario, a respective autoland controller is designed.

5 Simulation example

An application of the FlightController library in conjunction with the FLIGHTDYNAMICS library as well as an optimization tool is shown in this section. The goal of this scenario is to investigate effects of sensor malfunction and disturbances on safe landing performance of an aircraft using GPS- and ILS-aided autolanding systems. The scenario is based on the description given in the Civil Aircraft Landing Challenge (CALC) proposed by Airbus and Onera in 2016 (AIRBUS 2016).

5.1 Modeling and scenario description

The 6-Degree of Freedom (DoF) aircraft and systems model is provided by the FLIGHTDYNAMICS library, as is shown in Figure 7. The GPS-, glideslope- and localizer sensors are the relevant systems in this setup and are subjected to distubances listed further below. The trajectory is part of the NAMUG 26R RNAV approach to Munich airport, starting from waypoint DM428 and landing on RWY26R, where the aircraft has to conduct a right turn. For this scenario, the 4-D FMS was not activated, since the focus was not to model aircraft scheduling but instead the adherence to the autoland criteria specified by the CACS challenge. The main objectives the aircraft has to fulfill when touching down are:

1. Absolute bank angle Φ_{td} lower than 12°,
2. Absolute sideslip angle β_{td} lower than 14°,
3. Lateral distance from the centerline $\Delta r^b_{y,thres}$ smaller than 19 m,
4. Longitudinal distance from the threshold $\Delta r^b_{x,thres}$ smaller than 915 m and
5. Vertical velocity $V_{z,td}$ at touchdown smaller than 12 $\frac{ft}{s}$.

The geometry of the constraints can also be seen in Figure 8. Disturbances are considered in the following systems and inputs:

1. A maximum of 100 ms in each of the roll/pitch/yaw and throttle channels of the controller, modeling worst-case cumulative time-delays in the whole controller.
2. Sensor noise in the GPS receiver with a maximum angle error corresponding to 5 m of position deviation.
3. Sensor noise in the glideslope and localizer sensors with a maximum signal boundary deviation of ± 10 % from the reference.
4. Winds with maximum values of 25/10/15 kts for head-/tail-/crosswinds. These values are translated in the model to wind strength and direction.

In order perform the autolanding, a combined FMS and FCS controller was used which either tracks the 3-D GPS

DOI 10.3384/ecp218621

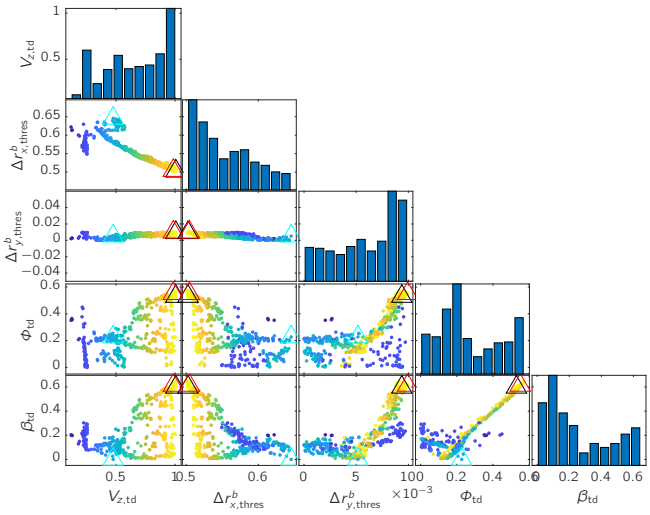
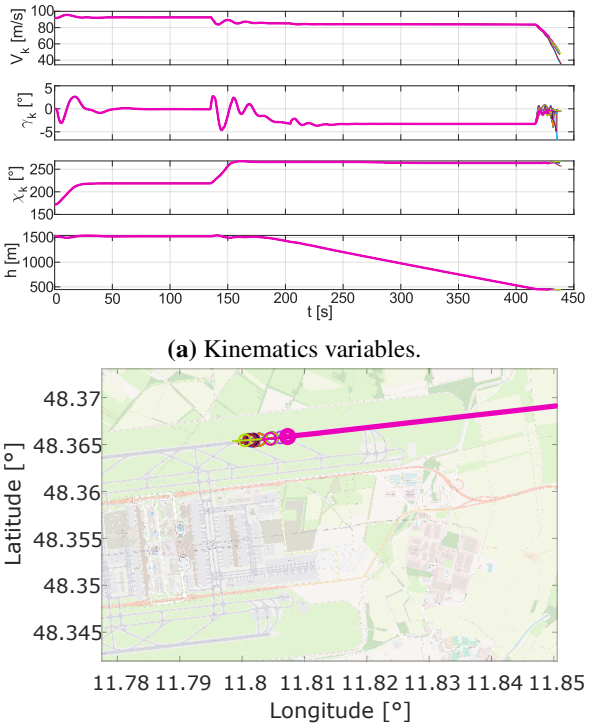


Figure 9. Anti-optimization results after 100 generations with values normalized to the CALC constraints (same colored data points belong together).



(a) Kinematics variables.

(b) Runway map with final trajectory phase, touchdown points are marked as circles.

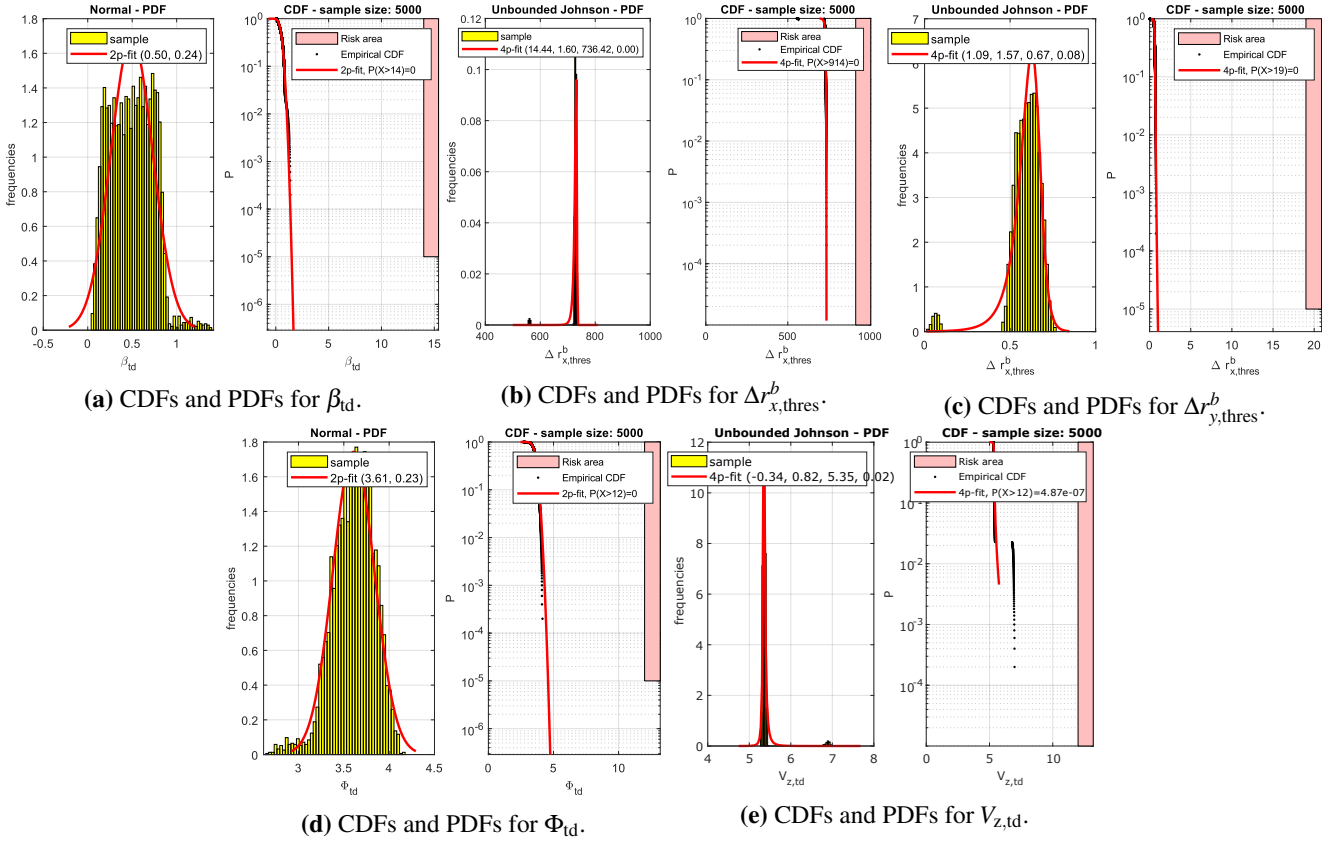
(c) Orientation in Euler angles.

Figure 10. Results of the controller optimization for the Instrument Landing System (ILS) guided approach.

trajectory or controls lateral/vertical deviations from the localizer and glideslope signals to zero. For the flare and touchdown, a variable-τ law controller was implemented (for details see e.g. (Looye 2007)). In accordance to the CALC challenge specifications, the simulation is stopped after all three gears have weight on the wheels and commence braking, therefore excluding ground roll and taxi from the scenario. According controllers are however available in the library, for simulation examples (rejected takeoff, mission simulation) see (May, Müller, and Looye 2021).

5.2 Simulation and optimization

One way of obtaining a robust controller is an iterative process where in a first step, a worst-case search / anti-optimization was conducted in order to find the cases where criteria assume their worst value. This was realized in this scenario using a genetic algorithm optimizer (either e.g. from the OPTIMIZATION library (Pfeiffer 2012) or the DLR tool Multi Objective Parameter Synthesis (MOPS) (Joos 2016; Ossmann and Joos 2019)) with a population of 20 individuals and 100 generations. The parameters for the antioptimization are the controller time delays, the sensor noises and boundary devation as well as the wind disturbance mentioned in the last section. This yields the results shown in Figure 9, which represents 2-D cuts (one for each combination of the five criteria) through the 5-dimensional solution space generated by the optimization, along with histograms of occurrence. All the feasible, pareto-optimal solutions of the 100 consecutive generations are plotted (the colormap does not represent a value but shows which solutions belong together), and allows insight to possible criterion dependencies (for example, longitudinal deviation seems to decrease for higher touchdown speeds). It was found that only $V_{z,td}$ (marked with black triangles) generated critical results that are close or higher than 1 (value is normalized with maximum). In the second step, the controller gains and time constants for path tracking / glideslope capturing and the flare law are optimized for worst-cases in a common optimization setup. This is achieved by consecutive evaluation of each cases and min-max weighting of the resulting criterion values (the optimizer tries to minimize the current highest normalized criterion value) until all constraints are satisfied or the optimizer cannot find any better solution. Due to the non-convex nature and complexity of the model and setup, a gradient-free optimization algorithm (pattern-search, (Hooke and Jeeves 1961)) was employed. Also since only $V_{z,td}$ exceeded its admissible value (larger than one), the other criteria were considered active during the optimization, but were not as critical as the velocity in the min-max sense. The results of this optimization is shown in Figure 10, with the optimal solution colored in magenta.



(a) CDFs and PDFs for β_{td}.

(b) CDFs and PDFs for $\Delta r^b_{x,\text{thres}}$.

(c) CDFs and PDFs for $\Delta r^b_{y,\text{thres}}$.

(d) CDFs and PDFs for Φ_{td}.

(e) CDFs and PDFs for $V_{\text{z,td}}$.

Figure 11. Monte-Carlo analysis of the found optimal controller subjected to random disturbances as specified in Section 5.1.

In the kinematics- (Figure 10a) and orientation (Figure 10c) plots, it can be seen that initial solutions touch down at slower speeds and higher incidence , which also leads to oscillations in the pitch axis due to the landing gear getting airborne again. Also cross-wind has a high effect during touchdown as the aircraft has not much control authority left to counter it. This leads to larger roll angles, which can also exceed the tolerances there. Furthermore there is a lateral deviation also caused through controller time delay and wind disturbance, creating an offset to the touchdown point and increasing deviation in the following roll phase (a roll controller is not implemented here). The ILS sensor noise however does not contribute to increased oscillations in the state variables which is probably due to its low magnitude and the damping of the controller. The final magenta controller however manages to fulfill all the requirements of the CALC including the longitudinal threshold deviation, which can be seen in the trajectory plot in Figure 10b.

In the third step, the obtained controller must be verified with respect to uncertainties and disturbances that might not have been present during the synthesis process. This is achieved in this case by a Monte-Carlo analysis taking into account the uncertain parameters listed in Section 5.1. The results of this process is shown in Figure 11, where 5000 simulations were conducted in an initial investigation. The failure probabilities postulated by CALC are shown as light red squares in the plots and may not be exceeded if the controller shall be considered verified[1]. For the criteria histograms, respective Probability Density Functions (PDFs) were fitted, leading to Cumulative Distribution Functions (CDFs) that mostly fit the obtained samples very well. One exception is the vertical velocity where small secondary occurrences are present around 7 $\frac{\text{ft}}{\text{s}}$ which are not covered by the unbounded Johnson PDF. This would make a sample-based analysis necessary in order to state the risk. Apart from this, all samples stay clear of the risk area with good margins, which allows the preliminary statement that the controller is safe in this exemplary simulation scenario. If that would not be the case, the three step process would have to be repeated iteratively, until the third verification step is successful.

6 Conclusion and outlook

In this paper, an overview of the FLIGHTCONTROL library was given, which was developed as an addon to the well-established DLR FLIGHTDYNAMICS library. The layout of the library and the main contents were described, along with examples of the main building blocks of an aircraft flight controller. Although these examples are mainly related to fixed-wing aircraft control, the FLIGHT-

[1] However, due to the small probabilities involved, the sample size should be much higher to ensure this with a certain confidence level. As this was considered out of scope for this exemplary simulation study and due to the long expected computation time, such a study was not performed here.

DOI
10.3384/ecp218621

CONTROL library also allows to integrate rotary wing, drone and other aircraft concepts, due to its modularity and straightforward application with other libraries (like the LINEARSYSTEMS2 and MSL).

A high-fidelity simulation study of an autoland approach trajectory was selected to showcase the controller synthesis capabilities of the library. For this scenario, an autoland controller was developed and tuned which behaves robustly under a wide range of external and internal disturbances. This was verified by an anti-optimization with subsequent multi-case optimization for controller gain tuning and finally a Monte-Carlo analysis to determine the safety of the found controller.

In the future, the FLIGHTCONTROL library will be further extended with means to automatically determine a controller's performance (e.g. via optimization and integration of common performance and handling quality measures). Furthermore, the support for model-based design approaches shall be extended and possibly concepts from optimal control (e.g. Model Predictive Control (MPC)) be included.

References

AIRBUS, ONERA (2016). "A Civilian Aircraft Landing Challenge". In.

Brockhaus, Rudolf, Wolfgang Alles, and Robert Luckner (2011). *Flugregelung*. Springer. ISBN: 9783642014437. URL: http://books.google.de/books?id=2IKXH3skXBwC.

Enns, Dale et al. (1994). "Dynamic inversion: an evolving methodology for flight control design". In: *International Journal of control* 59.1, pp. 71–91.

Hooke, R. and T.A. Jeeves (1961). "Direct Search Solution of Numerical and Statistical Problems". In: *Journal of Association for Computing Machinery (JACM)* 8.1, pp. 212–229.

Joos, Hans-Dieter (2016-06). "MOPS - Multi-Objective Parameter Synthesis, User's Guide V6.6". URL: https://elib.dlr.de/104761/.

Lambregts, Antonius A. (2013a). "TECS Generalized Airplane Control System Design - An Update". In: *Advances in Aerospace Guidance, Navigation and Control*. Springer, pp. 503–534.

Lambregts, Antonius A. (2013b). "THCS generalized airplane control system design". In: *2013 CEAS Conference on Guidance, Navigation and Control, Delft, The Netherlands*.

Liersch, Carsten M. and Martin Hepperle (2011-12). "A distributed toolbox for multidisciplinary preliminary aircraft design". In: *CEAS Aeronautical Journal*. CEAS Aeronautical Journal Volume 2.Number 1-4, pp. 57–68. URL: https://elib.dlr.de/74509/.

Looye, Gertjan (2007). *An Integrated Approach to Aircraft Modelling and Flight Control Law Design*. ISBN: 9789053351482.

Looye, Gertjan (2008). "The new DLR flight dynamics library". In: *Proceedings of the 6th International Modelica Conference*. Vol. 1, pp. 193–202.

Looye, Gertjan et al. (2014-09). "Object-Oriented Aircraft Modelling with the DLR FlightDynamics Library". In: *63. Deutscher Luft- und Raumfahrtkongress*. Augsburg, Germany: Deutsche Gesellschaft für Luft- und Raumfahrt.

May, Marc, Reiko Müller, and Gertjan Looye (2021). "Aircraft Mission Simulation with the updated FlightDynamics Library". In: *Modelica Conferences*, pp. 155–162.

Müller, Reiko (2020). "The Gnomonic Projection for B-Spline parameterized 4-D Trajectory Optimization Problems". In: *AIAA AVIATION 2020 FORUM*. American Institute of Aeronautics and Astronautics. DOI: 10.2514/6.2020-3205. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2020-3205. URL: https://arc.aiaa.org/doi/abs/10.2514/6.2020-3205.

Müller, Reiko (2021). *Four-dimensional multi-objective trajectory optimization using high-fidelity aircraft models and a projected parameterization : = Vierdimensionale multikriterielle Bahnoptimierung unter Verwendung hochgenauer Flugzeugmodelle und einer projizierten Parametrierungsmethode / vorgelegt von Reiko Müller*. eng. Aachen.

Müller, Reiko, Jane Jean Kiam, and Federico Mothes (2018). "Multiphysical Simulation of a Semi-Autonomous Solar Powered High Altitude Pseudo-Satellite". In: *Aerospace Conference, 2018 IEEE, Big Sky, MT, USA*. Institute of Electrical and Electronics Engineers, pp. 1–16.

Müller, Reiko and Markus Ritter (2017). "Virtual flight testing of a controller for gust load alleviation using FMI for cosimulation". In: *Proceedings of the 12th International Modelica Conference*. Vol. 132. Modelica Association and Linköping University Electronic Press, pp. 921–928.

Ossmann, Daniel and Hans-Dieter Joos (2019). "Multiobjective optimization–based fault-tolerant flight control system design". In: *International Journal of Robust and Nonlinear Control* 29.16, pp. 5341–5355. DOI: https://doi.org/10.1002/rnc.3955. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rnc.3955. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/rnc.3955.

Pfeiffer, Andreas (2012). "Optimization library for interactive multi-criteria optimization tasks". In: *9th International Modelica Conference*.

Stevens, Brian L., Frank L. Lewis, and Eric N. Johnson (2015). *Aircraft control and simulation: dynamics, controls design, and autonomous systems*. John Wiley & Sons.

DOI
10.3384/ecp218621

Quasi-Periodic Feedforward Control Based on Inverse Model Tabled FFT

Tilman Bünte[1] Jakub Tobolář[2]

[1,2]Institute of Vehicle Concepts, German Aerospace Center (DLR), {Tilman.Buente, Jakub Tobolar}@dlr.de

Abstract

Mitigating periodic oscillations (e.g. in rotating systems) is a common control engineering problem. Fast Fourier Transform (FFT)-based methods are well-suited for respective analysis. While FFT algorithms inherently assume signal periodicity, rotating systems often exhibit true periodic behavior (e.g., shaft rotation frequencies). Using angle-sampled data rather than time-sampled data allows direct analysis of oscillations relative to rotational cycles, which is particularly useful for tracking unbalance or periodic external excitation in rotating assemblies.
Modelica provides several built-in resources to address these challenges. First of all, inverse models have the potential to derive an ideal control signal in time domain. For periodic disturbances, this ideal control is likely to be approximated well by a periodic, i.e. Fourier-transformable signal. Modelica is an appropriate model environment to store and retrieve tabled FFT data depending on operating conditions such as rotational speed. In a real-time application, synthesizing control signals from precomputed Fourier tables offers a practical alternative to executing potentially complex inverse models online, reducing computational effort and system complexity. The paper demonstrates this approach using the example of mitigating oscillations induced by an internal combustion engine in a hybrid automotive power train.
Keywords: Model Inversion, Fast Fourier Transform, Fourier Synthesis, Vehicle Power Train, Torsional Vibration Suppression, Disturbance Cancelling

1 Introduction

This paper addresses mitigation or cancellation of periodic or quasi-periodic oscillations in technical systems caused by external or self-excitation mechanisms. To ensure clarity and simplicity, the presentation here focusses on mechanical 1D-rotational systems. However, the results can analogously be applied to other technical fields, for instance electric circuits or acoustic systems. The presence of a suitable actuator such as an electric motor is assumed which can be used as a torque source for control action. Considerations are here restricted to feedforward control without denying the potential of feedback control. More precisely, the approach presented here could well be combined with additional feedback control or other advanced methods like iterative control, invariant manifold tracking, and hybrid testing.

The novelty and main contribution of this paper is the Modelica based combination of the techniques

a) of model inversion for the synthesis of a theoretically perfect feedforward control,

b) the Short-Time Fourier Transform (STFT) of the quasi-periodic control signal and

c) model reduction of the control by approximating it with a correspondingly Fourier synthesized signal.

Moreover, a Modelica package is introduced and a practical application is demonstrated. To sum up the corresponding state of the art:

a) Model inversion in Modelica is a well established technique (Thümmel et al. 2005).

b) STFT is primarily used for analysing non-stationary signals whose frequency spectrum changes over time. In (Bünte 2011) it is used to determine describing functions of Modelica models. In (Kuhn 2017) it is used for stationary periodic steady state identification of AC electrical systems.

c) FFT for model reduction is applied for instance in the fields of material science (Gierden 2024) and Finite Element Method (FEM) based product design (Hülsebrock 2023).

In addressing the issue of quasi-periodic disturbances impacting a rotational system, we consider that the primary period of any given oscillation maintains a constant ratio relative to a single revolution of a reference shaft. For instance, an internal combustion engine (ICE) generates a periodic torque and hence torsional oscillations in the drivetrain due to the sequential firing of its cylinders. The harmonic structure of the crankshaft torque oscillations is typically influenced by factors like ICE speed, load torque, and throttle position. Given the known period of the oscillation, for a specific operational case (ICE speed, throttle position etc.) a STFT of the periodic crankshaft torque can be performed over the time window of one period. The resulting Fourier coefficients for a limited number of harmonics approach the dynamic torque characteristic of the ICE for this operational case. Thus, the complex thermodynamic process plus piston and rod related

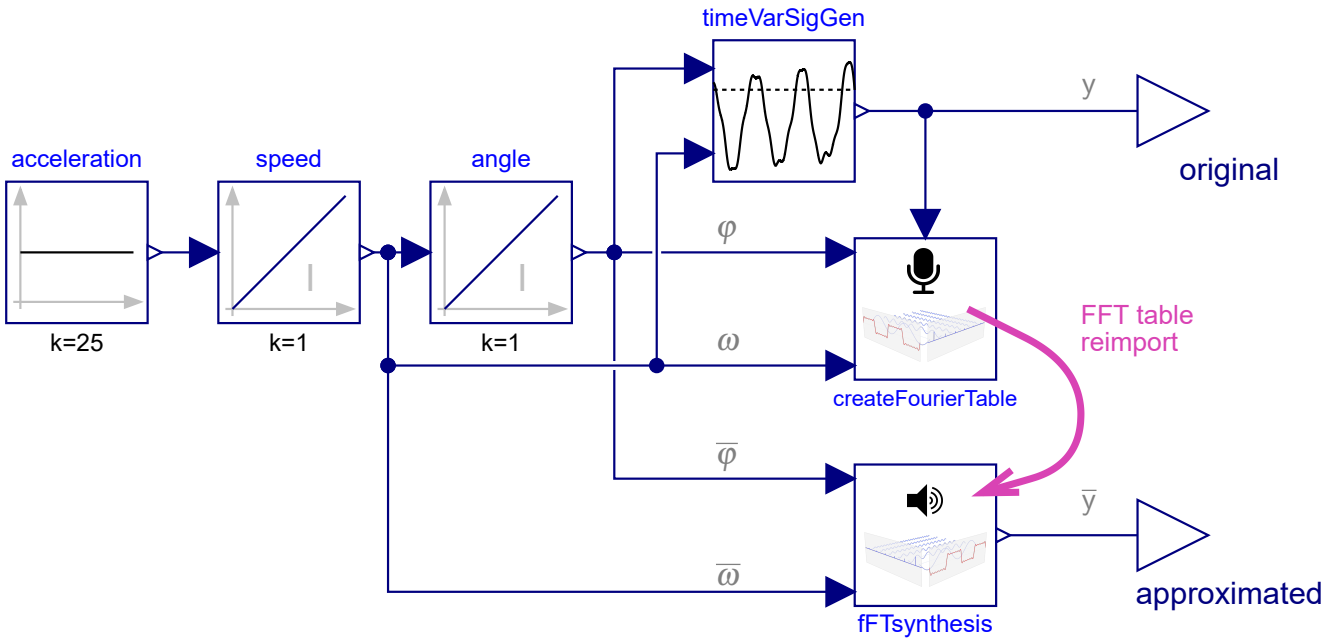


Figure 1. Demo Modelica model combining instances of the blocks `CreateFourierTable` and `FFTsynthesis`.

inertia dynamics in an ICE could be approximated and represented by a set of Fourier coefficients for the torque on the crankshaft (Chauvin et al. 2004), stored in a (multidimensional) table depending on operational parameters while using appropriate gridding. Obviously, a Modelica model of an ICE can be easily built as a Fourier synthesis-based signal generator acting as torque source and using the above-mentioned table for parameterization.

The paper is organized as follows. Section 2 presents a concise Modelica package which comprises classes for a) the repeated determination of STFT coefficients for a signal of a time varying oscillating system and b) a signal synthesizer as an effective approximative representation of the previously decomposed system and c) a demo model combining a) and b). A complex quasi-periodic nonlinear system which may be difficult to be solved numerically can thus potentially be replaced by interpolation from a lookup table. For some cases this can serve as a model reduction technique e.g. in view of real time capability. Section 3 considers the cancellation of quasi-periodic disturbances on a system by means of model inversion (well familiar to Modelicans (Thümmel et al. 2005)). Section 4 integrates the features of Sections 2 and Section 3 and thus, represents the main contribution of this work: A feedforward control scheme is presented which is based on the STFT decomposed ideal control signal from the inverse model. In Section 5, a hybrid automotive power train model with ICE and integrated starter generator (ISG) is used to demonstrate the practical applicability and relevance of the feedforward control scheme. Finally, Section 6 provides a conclusion and an outlook.

2 A Modelica Package for Fourier Decomposition and Playback of Quasi-Periodic signals

A Modelica package `FourierTable` was designed which provides

- a recording block `CreateFourierTable` using repeated STFT of a quasi-periodic signal y occurring in a Modelica model and
- a counterpart block `FFTsynthesis` for playback of a signal $\bar{y}$ using Fourier synthesis.

See Fig. 1 for an example which combines both classes in one Modelica model. A reference y ("original") is generated by a time varying signal generator `TimeVarSigGen`. The simulation of this demo model allows for direct comparison of y and its approximation $\bar{y}$. To do this, it is necessary to simulate twice. In the first run, the Fourier coefficient table is generated by `CreateFourierTable` and saved to a mat-file. During initialisation of the second run, this table is loaded as parameterization of `FFTsynthesis` which facilitates a meaningful comparison.

For the recording using `CreateFourierTable` it is assumed that the total Modelica model has the following features:

- There is a signal ω and its integral φ (e.g. with the physical meaning of angular speed and position, respectively, of a rotating shaft). It is further assumed that ω is varying slowly enough, such that "quasi-periodicity" of a considered signal y is always sufficiently well maintained. The reader is referred to Section 2.4 for deeper considerations.

DOI 10.3384/ecp218629

- The actual fundamental period of the signal y corresponds to an increase of φ by 2π (one revolution of the shaft). Consequently, periodicity of y relates to φ, not time.
- The composition of the signal y in terms of harmonics depends only on one single time varying parameter q. The latter could be but not necessarily is $q = \omega$. For the sake of simplicity, this case is assumed in the following.

During a simulation of the considered system while typically ω is monotonously increasing, events are triggered when ω reaches predefined threshold values. After each event, a STFT analysis of y is performed over the window of a complete period of φ (i.e. one revolution). Hereafter, the Fourier coefficients are stored into a new row of a table with the actual ω value in the first column, cf. Table 1 for illustration.

For playback (normally in another Modelica model), the previously collected tabled data is used as parameterization for an instance of `FFTsynthesis`. Using $\overline{\varphi}$ and $\overline{\omega}$ as inputs to the playback block, Fourier coefficients are interpolated from the table and a quasi-periodic signal $\overline{y}$ is generated which approaches y of the original system for $\overline{\omega} = \omega$ and $\overline{\varphi} \pmod{2\pi} = \varphi \pmod{2\pi}$.

The record and playback blocks are now described in more detail.

2.1 Data Sampling and Repeated STFT of the Quasi-Periodic Signal

The block `CreateFourierTable` has the following parameters:

```
parameter Integer n = 9 "Exponent; Number
 of angle samples per period is np = 2^n";
parameter Modelica.Units.SI.AngularVelocity
  wnodes[:] = linspace(10,100,5) "List of
 frequency nodes where data acquisition for
 STFT evaluation is triggered";
parameter Integer nh = 2 "Number of
 harmonics";
```

The block mainly consists of a single algorithm section. It is activated each time when the monotonously increasing signal ω surpasses one of the scheduled `wnodes` values. Now, sampled data for y is being collected during the newly started period in order to prepare for later STFT. The current φ period is divided into an equidistant grid of 2^n samples according to the model parameter `n` and data collection is triggered at each φ grid point. After the period is complete, the block is put to sleep til the next ω-induced activation in order to save non-essential events and thus to reduce computational effort.

As soon as data collection over the period is completed, an STFT is performed using the function `Math.FastFourierTransform.realFFT()` (Cooley and Tukey 1965) from Modelica Standard Library. A second input to the function is the number of frequencies specified by the model parameter `nfi` $= 1 +$ `nh` equalling the DC part plus the desired number of harmonics to be considered. The resulting amplitudes a_i and phases ϕ_i of the harmonics are stored into a new row of a table whose first column holds the mean value of ω during the collected period, cf. Table 1. Hence, the ω node differs a bit from the actual `wnodes` value. In most cases the exact values of the table interpolation nodes does not matter; the parameterization of `wnodes` can be used for an approximate scheduling. The table is filled during the simulation of the total model while ω is monotonously increasing and sweeps over the range of scheduled `wnodes` values. Section 2.4 will provide comments on selecting `wnodes` and appropriately scheduling the rate of change for ω.

Upon end of simulation, the resulting table is stored as a matrix which represents the set of Fourier coefficients for the quasi-periodic signal y at the given ω nodes. The sampled data collection of quasi-periodic signals and Fourier decomposition was reused from (Bünte 2011).

2.2 Signal Approximation by Fourier Synthesis

A set of Fourier coefficients (compliant with the format of the table generated from `CreateFourierTable` as described before) is the only parameter of the block `FFTsynthesis`:

```
parameter Real table[:,:] "FFT table as
 function of angular velocity (w rev/s (col.
  1), amplitudes (col. 2:nfi+1), phases (
 col. nfi+2:end))";
```

For the given input value of $\overline{\omega}$ (cf. Fig. 1), the corresponding set of Fourier coefficients is interpolated from the table. The synthesis of the signal $\overline{y}$ is straightforwardly computed by superposition of the `nfi` frequencies (DC part plus `nh` harmonics) with the (real or virtual shaft angle) input $\overline{\varphi}$ as independent variable:

$$\overline{y}(\overline{\varphi},\overline{\omega}) = \sum_{i=0}^{nh} a_i(\overline{\omega}) \cos\left(i\,\overline{\varphi} + \phi_i(\overline{\omega})\right) \qquad (1)$$

For a constant $\overline{\omega}$ the resulting signal $\overline{y}$ is strictly periodic. If $\overline{\omega}$ is slowly varying compared to the periodicity 2π of the cosine function argument, then $\overline{y}$ renders *quasi*-periodic.

2.3 Demo Model Simulations

Fig. 2 presents the results of two successive simulations of the demo model introduced with Fig. 1. The *original* output from the signal generator `TimeVarSigGen` consists of a DC part plus a stationary first harmonic plus a second harmonic whose amplitude changes with ω. Finally, there is a stationary fifth harmonic with a small amplitude:

```
y = -2.212
    +3.346*cos(1*phi + 0.526)
    +1.233*cos(2*phi + 0.745)*(0.3+1*cos
     ((w-10)/5))
    +0.300*cos(5*phi - 0.745);
```

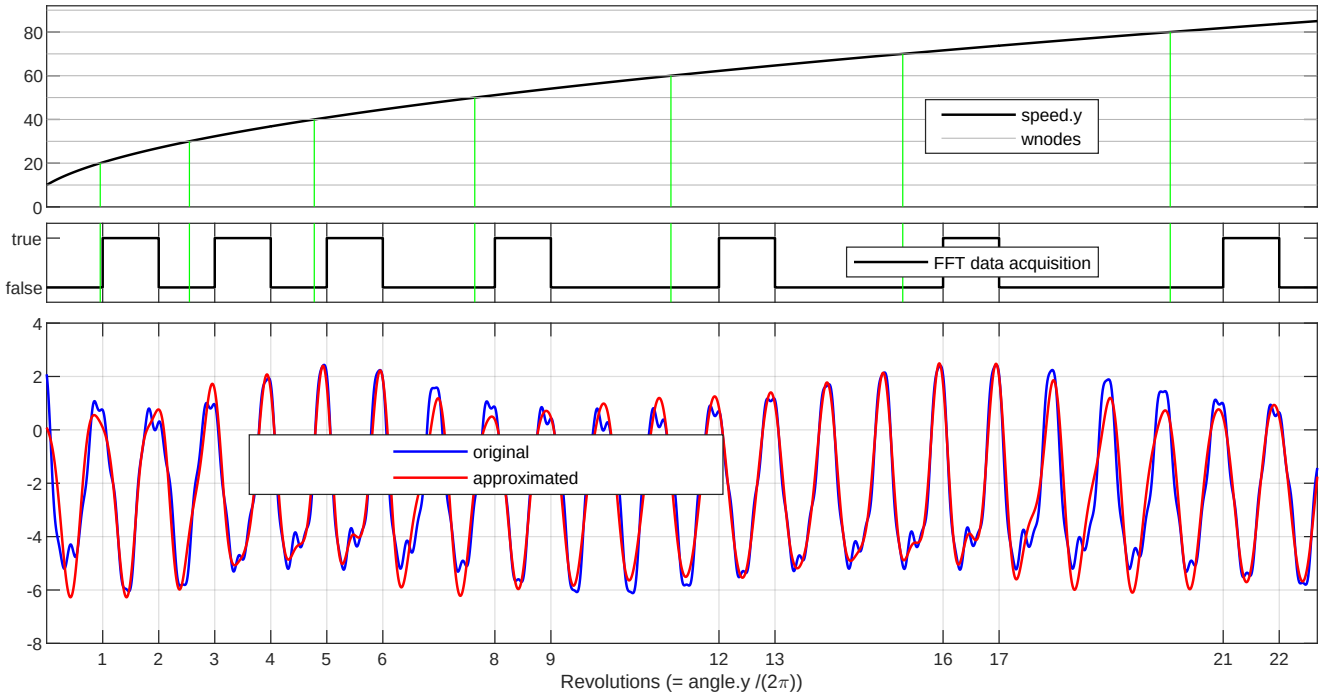


Figure 2. Simulation results from the demo Modelica model shown in Fig. 1. Top: Rotational speed and thresholds; Middle: Indication of data acquisition phases; Bottom: Comparison of original signal y with its approximation $\bar{y}$.

A first run of the model is made here with `CreateFourierTable.nh` $= 2$, i.e. only two harmonics are to be considered for the STFT. The table generated by `CreateFourierTable` is automatically saved to a file and then used as parameterization for the block `FFTsynthesis` in a second simulation. (Additional simulations with the model would deliver the same results as from the second run.) The upper plot in Fig. 2 shows the rising of $\omega =$ `speed.y` and the constant `wnodes` thresholds. Green vertical lines indicate the instants where the thresholds are surpassed. The FFT data acquisition phases, each covering the complete subsequent revolution, can be seen from the middle plot. Rather than time, in all plots the number of revolutions equalling $\varphi/(2\pi)$ is used as abscissa. The lower plot compares the *original* signal `TimeVarSigGen.y` with the *approximated* signal `FFTsynthesis.y`. Table 1 exhibits the example's numerical values for the Fourier coefficients interpolation table, compliant with the notation used in eq. (1). Where $\overline{\omega}$ is close to interpolation nodes, the lower plot shows good agreement of the approximated signal $\bar{y}$ (red) with the original reference y (blue). Understandably, here the interpolation of the Fourier coefficients for signal reconstruction matches best the recorded characteristics.

2.4 Parameters Influencing Approximation Quality

Obviously, the accuracy of the approximated signal depends on some parameters during data acquisition. Some skill and experience is required for their choice. In practice, the parameters of `CreateFourierTable` (see Section 2.1) and the total model can be adapted such that sufficient precision of the approximated signal is attained:

1. If denser gridding of `wnodes` is used, then the agreement of $\bar{y}$ with y is correspondingly better in-between

Table 1. Interpolation table for Fourier coefficients, compliant with the notation used in eq. (1), resulting from the demo model simulations.

$\overline{\omega}$	a_0	a_1	a_2	$\phi_0[^\circ]$	$\phi_1[^\circ]$	$\phi_2[^\circ]$
23.827	−2.2350	3.3433	0.71776	0.0000	31.639	−136.93
34.612	−2.1557	3.4258	0.65922	0.0000	27.589	42.980
42.742	−2.2273	3.3041	1.5210	0.0000	30.678	42.697
52.629	−2.2416	3.2993	0.40789	0.0000	31.544	−137.03
63.456	−2.1803	3.3917	0.026966	0.0000	28.770	45.790
72.687	−2.2122	3.3350	1.5903	0.0000	30.089	42.699
82.791	−2.2347	3.3057	0.14749	0.0000	31.174	−136.82

DOI
10.3384/ecp218629

the interpolation points.

2. The same applies to the rate of change of ω during data acquisition. If ω or any other operating conditions q are not changing slowly enough, periodicity is not sufficiently satisfied and the result may be spoiled. Generic statements about reasonable limits on the rate of change are difficult to derive and exceed the objective of this publication. A heuristic approach will serve well in most applications and the rise of ω should be controlled to be sufficiently moderate in the simulation such that the results are satisfactory.

3. The number `nh` of harmonics considered in the recording affects the achievable accuracy and should be chosen accordingly.

4. A large value of the parameter `n` contributes to good results. The Nyquist–Shannon sampling theorem (Franklin et al. 1990) should be regarded in the context of the considered harmonics. On the other hand, more data sampling events are triggered in the block `CreateFourierTable` during the FFT data acquisition phases and the analysis simulation takes more effort.

Note that in the example shown in Fig. 2 some of the above enumerated recommendations are violated for the sake of illustrating their effect: Due to the rapid rise of ω the periodicity assumption was not held. As a side effect, the resulting set of interpolation nodes for ω (cf. Table1) is coarse. The STFT only considered two harmonics, therefore the fifth harmonic oscillation of the original signal is not captured.

3 Inverse Model Based Disturbance Cancellation

Here, we use the term *disturbance* to mean an undesired effect on a specific system signal induced by (quasi-)periodic external or self-excited quantities.
Inverse linear and nonlinear models are of significant meaning as building blocks for the design of control systems, both for feedforward and feedback control. One of the outstanding features of Modelica is that nonlinear inverse models may under circumstances be automatically generated from the plant model (Thümmel et al. 2005).
Particularly, when addressing the problem class considered in this paper, the ideal control signal for cancelling disturbance can be derived using an inverted model, provided that the conditions for model inversion are fulfilled such as uniqueness and stable zero dynamics. Details on this can be found in (Thümmel et al. 2005; Thümmel 2007). To accomplish model inversion, the equation defining the control signal (i.e. the actuator input) is removed. Hence the total model equations will be solved for this quantity to be an output. The number of equations needs to remain balanced with the number of unknowns such that the model equations can be solved and the model can be translated and simulated. Therefore, replacing the removed control signal, an equation is added to the Modelica model which represents a desired behaviour of a specific model's quantity. In our case, this is an adequate disturbance cancelling condition, e.g. with the meaning that a desired target signal is set to zero or follows perfectly a certain reference.

In the case that the inverse model is lean and real-time capable, it can be used to generate the control signal for a given (periodic) disturbance and that control signal can be applied feedforward to a real system. If the basic model matches well enough the real system, then the control signal has the potential to significantly mitigate the disturbance. However, in general, solving the inverse model in real time may be unfeasible or computationally expensive. Moreover, the approach presupposes that the inverse model stays in tune with the real system in terms of system states and/or operating conditions. These challenges suggest to use an abstraction of the inverse model which combines the approaches from Section 2 and this one, to be introduced in the next section.

4 Feedforward Disturbance Cancellation Using Fourier Synthesized Approximation of the Ideal Control Signal

As this section is the core, the key idea of the paper is presented in four subsequent steps while using the Modelica implementation of the recording and playback blocks presented in Section 2:

1. As outlined in the previous section, an inverse model can be employed to generate an ideal, theoretic control signal for a given objective such as cancelling a target signal. In case of constant operating conditions including a perfectly periodic disturbance and the system model being in steady state oscillation this approach yields a periodic solution for the control variable.

2. The *pattern* of this periodic signal in terms of its harmonics can be expressed as Fourier coefficients and be considered as an eligible blueprint for periodic feedforward control action (to be applied with appropriate phase) for the given system model at steady operating conditions.

3. The blueprint can be applied to a real system by means of a Fourier synthesizer if

   - the periodic trigger for the disturbance can be measured or estimated and
   - the system matches the operating conditions of the underlying model which was used to generate the pattern.

4. If the operating conditions of the real system do vary and it is not in perfect steady state, then modified patterns are applied, each fitting the prevailing operating conditions.

    - For a manifold of operating conditions, the patterns are Fourier synthesized based on interpolated Fourier coefficient datasets which were generated each via STFT using a sufficiently dense node gridding to achieve good approximation.
    - For practical application of the approach, the recording of the patterns with the inverse system model should be done while the operating conditions are varying sufficiently slowly such that the inverse system model is always in quasi steady state.
    - The same applies to the playback of the patterns on the real system. The results will be best if the operating conditions vary only slowly such that the real system model is in quasi steady state oscillation.

5 Automotive Application Example: Hybrid Power Train

The application example used here is based on a simplified automotive mild hybrid power train consisting of a four-stroke four-cylinder internal combustion engine (ICE) plus integrated starter generator (ISG) mounted on the extension of the crankshaft.

5.1 Modelica Power Train Uncontrolled Base Model

The total Modelica model of the power train, depicted in Fig. 4, is completed by an abstracted drive train including rotational spring and damper plus an inertia representing the total load including the translated mass of the vehicle body. To facilitate a perfectly common scenario for various variants to be compared later, the model features a signal source (`speedScheduler`) which imposes a predesigned rotational speed profile to the inertia.
The ICE produces a quasi-periodic dynamic torque that varies with rotational speed. This torque consists of two main components:

- The forces generated by the pressure in the four cylinders, which follow the firing order and occur every second crankshaft revolution, influenced by the geometric arrangement like in boxer or V engines (Laschet 1988).
- The asymmetric inertial forces of the pistons and connecting rods.

Fig. 3 illustrates the resulting periodic torque pattern at full throttle for the ICE model used with a set of curves representing different crankshaft speeds.

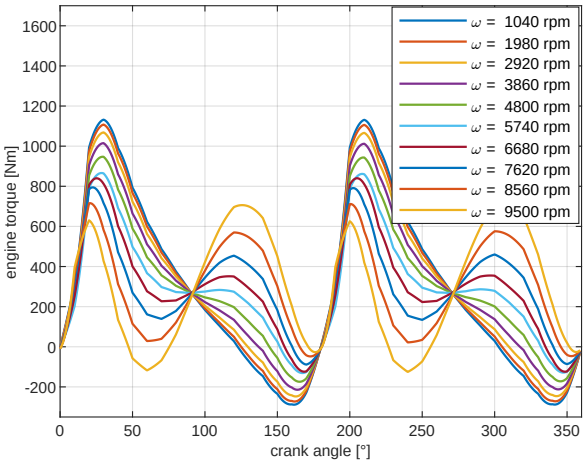


Figure 3. Full throttle periodic output torque of the four-stroke four-cylinder ICE depending on crank shaft speed ω.

The objective of the feedforward control derived in the next two subsections is to compensate for the quasi-periodic ICE torque variations around the DC torque component of the ICE such that torsional oscillations in the drive train are minimized.

5.2 Fourier Coefficient Recording of Ideal Control Using Inverse Model

Following the concept introduced in Section 2.1, a Fourier coefficient table is generated using the inverted model shown in Fig. 5. Therefore, starting from the model of the uncontrolled base model (Fig. 4), a `CreateFourierTable` block is added to the system. The block `Math.InverseBlockConstraints` from the Modelica Standard Library is used to invert the model such that the combined output torque of ICE plus ISG equals a constant torque (set to 280 Nm here to reflect the approximate mean torque of the ICE). In return, the equation defining the input torque of the ISG is dropped. By effect of the model inversion this quantity is calculated during the simulation and, repetitively, the STFT is performed after discrete rotational speed values are surpassed.

5.3 Power Train Feedforward Control for Approximative Disturbance Cancellation

Feedforward control of the hybrid power train using the Fourier synthesis introduced in Section 2.2 is applied here. The model shown in Fig. 6 uses the ISG torque as control variable. An instantiation of the block `FFTsynthesis` contains the FFT approximated and tabled information of the perfect disturbance cancellation control previously acquired using Fig. 5 as described above.

5.4 Comparative Simulation Results

For the plots shown in Fig. 7, results are compiled from simulations with the three models described in Sections 5.1-5.3 and Figures 4-6, respectively.

The scenario for all three simulations is the same: The drive train is accelerated constantly according to a given

DOI
10.3384/ecp218629

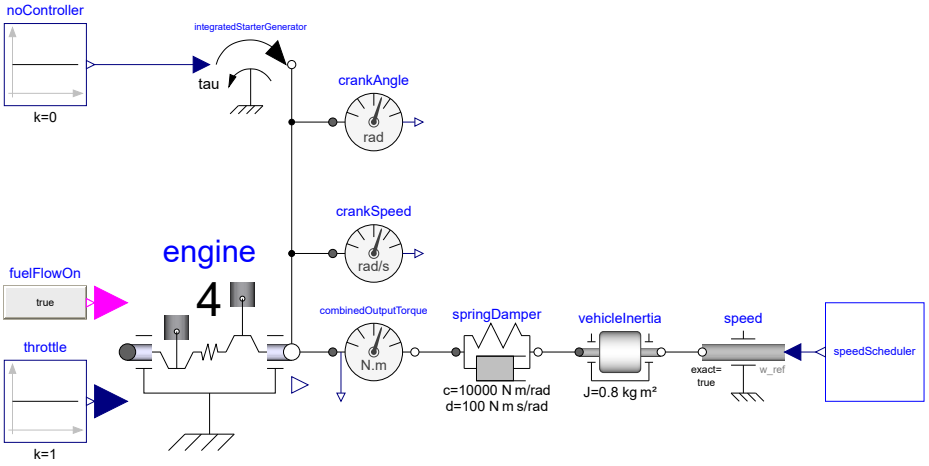


Figure 4. Uncontrolled power train model with imposed speed.

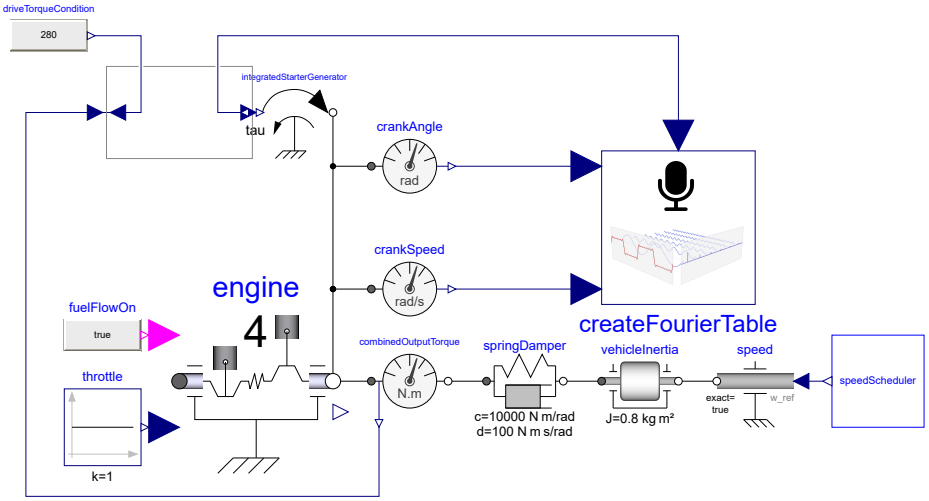


Figure 5. Inverse power train model for repetitive STFT recording of ideal control action.

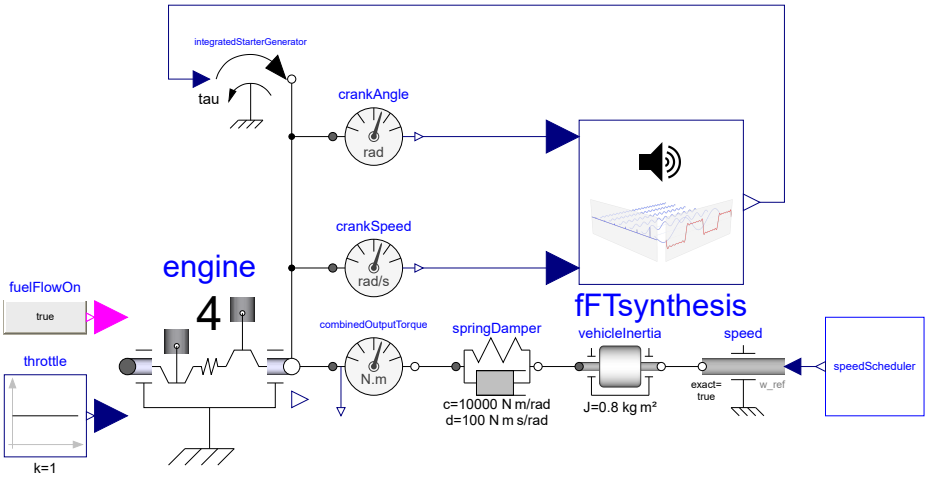


Figure 6. Feedforward controlled power train.

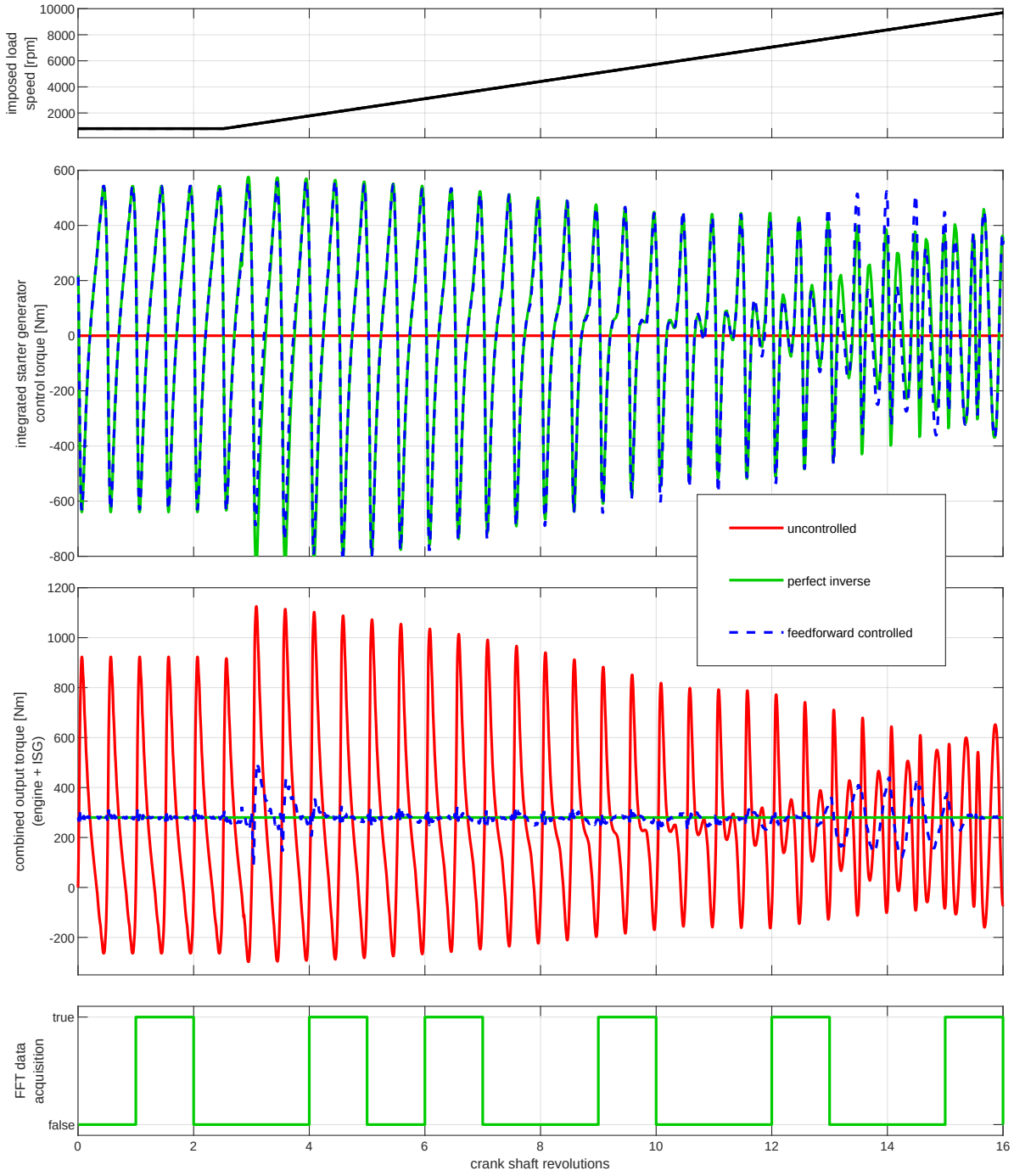


Figure 7. Simulation results comparing various variants of the automotive power train under quasi-periodic disturbance. 1st row plot: rotational speed; 2nd row: ISG control torque; 3rd row: total torque from ICE+ISG; 4th row: data acquisition phases.

DOI
10.3384/ecp218629

speed ramp. By this means, the acceleration is even higher than in a flying start with the purpose that within only few revolutions a large range of speeds is covered. Thus, the speed dependency of the quasi-periodic disturbance is well cognizable in the plots of single simulations. The uppermost plot in Fig. 7 shows the imposed speed of the drive train being the same for all three models. The control input (i.e. the ISG torque) is plotted in the second row plot. The control torque of the uncontrolled base model (corr. to Fig. 4, red line) is zero. The resulting combined output torque of ICE plus ISG exhibits high amplitude quasi-periodic spikes (see third row plot, red line). With model inversion applied (green line each, corr. to Fig. 5) the combined torque is set to zero and the resulting perfect (theoretic) ISG control torque renders quasi-periodic. During the same simulation, data for STFT is acquired in the complete revolutions where the corresponding flag is true, see lowermost plot. After completion of each data acquisition phase, computed Fourier coefficients are stored in the FFT table in a node associated with the mean speed of the past revolution. As usual with interpolation from tabled data, the result is most precise when the query speed is close to one of the speed nodes. This effect can be observed from the simulation results with the feedforward controlled model (corr. to Fig. 6, blue dashed lines). At speeds where FFT data was acquired with the inverse model, the feedforward control performs significantly better in terms of the combined output torque.

6 Conclusion and Outlook

In this paper the combination of the following three Modelica based techniques has been introduced:

- Model inversion for the synthesis of a theoretically perfect feedforward control aiming at compensation of time varying periodic oscillations.
- Repetitive Short-Time Fourier Transform (STFT) of the quasi-periodic control signal.
- Model reduction of the ideal control by approximating it with a correspondingly Fourier synthesized signal depending on varying operation conditions.

A Modelica package has been explicated and the practical application to mitigation of self-excited quasi-periodic disturbances in an automotive hybrid power train was demonstrated.

The effectiveness of the presented feedforward control scheme depends on the periodicity of the disturbances. The operating conditions (for example the rotational speed) should be varying slowly enough such that quasi-periodic behaviour prevails at all times, i.e. the relevant signal should be well approximated in sections by periodic signals. Moreover, a measurement or estimation of the disturbance trigger (e.g. the rotational angle of a shaft) should be available. And, the disturbance to be mitigated should be truly (reproducibly) depending on the trigger. Under these assumptions a feedforward control following the concept described here should be worthwhile to be considered for cancellation of oscillations.

Note that suppression of disturbances which are stochastic (noise) or of other external or non-(quasi-)periodic nature are not in the scope of the approach. Therefore, these were not considered here.

Possible enhancements of the approach include the following:

- The use of more elaborated window functions to mitigate spectral leakage artefacts.
- Currently, the window for data acquisition and Short-Time Fourier Transform covers one shaft revolution, with the shaft angle starting at the next multiple of 2π. An appropriate window shift could handle this issue to get the ω interpolation nodes of the resulting table closer to the scheduled `wnodes` values.
- The Fourier coefficient table employed so far covers a single varying parameter q which was synonymic with ω in this paper. However, a second influencing variable (such as the throttle position in our application example) could be considered and the table could provide bivariate interpolation (using `CombiTable2Ds` instead of `CombiTable1Ds` within `CreateFourierTable`). Of course, this requires an elaborate parameter gridding and approaching schedule (going beyond the previous concept explained with `wnodes`) for the analysis/recording simulation while using the `CreateFourierTable` features.

References

Bünte, Tilman (2011). "Recording of Model Frequency Responses and Describing Functions in Modelica". In: *8th International Modelica Conference 2011*. URL: https://elib.dlr.de/68920/.

Chauvin, J., G. Corde, P. Moulin, M. Castagne, N. Petit, and P. Rouchon (2004). "Real-time combustion torque estimation on a diesel engine test bench using an adaptive Fourier basis decomposition". In: *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*. Vol. 2, 1695–1702 Vol.2. DOI: 10.1109/CDC.2004.1430288.

Cooley, James W. and John W. Tukey (1965). "An Algorithm for the Machine Calculation of Complex Fourier Series". In: *Mathematics of Computation* 19.90, pp. 297–301. ISSN: 00255718, 10886842. URL: http://www.jstor.org/stable/2003354 (visited on 2025-07-30).

Franklin, Gene F., David J. Powell, and Michael L. Workman (1990). *Digital Control of Dynamic Systems*. Second, pp. xxi + 837. ISBN: 0-201-11938-2.

Gierden, Christian (2024). "Model order reduction for FE-FFT-based multiscale material modeling". Dissertation. Aachen: Rheinisch-Westfälische Technische Hochschule Aachen. DOI: 10.18154/RWTH-2024-05191.

Hülsebrock, Moritz (2023). “Methode zur parametrischen globalen Modellordnungsreduktion für eine effiziente Validierung dynamischer Systeme”. de. PhD thesis. Darmstadt: Technische Universität Darmstadt, xvi, 114 Seiten. DOI: https://doi.org/10.26083/tuprints-00022844.

Kuhn, Martin R. (2017). “Periodic Steady State Identification of electrical circuits”. In: *12th International Modelica Conference*. Linköping Electronic Conference Proceedings. Linköping University Electronic Press, pp. 493–505. URL: https://elib.dlr.de/115022/.

Laschet, Andreas (1988). *Simulation von Antriebssystemen: Modellbildung der Schwingungssysteme und Beispiele aus der Antriebstechnik*. Zugl.: Aachen, Techn. Hochschule, Dissertation u. d. T. : Laschet, Andreas: Entwicklung eines Verfahrens zur rechnergestützten Simulation von Torsionsschwingungen in Antriebssystemen. Berlin (de): Springer.

Thümmel, Michael (2007). “Modellbasierte Regelung mit nichtlinearen inversen Systemen und Beobachtern von Robotern mit elastischen Gelenken.” Dissertation. Technische Universität München. URL: https://elib.dlr.de/51315/.

Thümmel, Michael, Gertjan Looye, Matthias Kurze, Martin Otter, and Johann Bals (2005). “Nonlinear Inverse Models for Control”. In: *Proceedings of the 4th International Modelica Conference*. Ed. by Gerhard Schmitz. LIDO-Berichtsjahr=2005, pp. 267–279. URL: https://elib.dlr.de/12298/.

Liaison: an open-source tool for distributed co-simulations

Luis Sanchez-Heres[1] Fredrik Olsson[2] Jan Östh[3]

[1,2,3] Maritime department, RISE Research Institutes of Sweden, Sweden,

[1] luis.sanchez-heres@ri.se, [2] fredrik.x.olsson@ri.se

[3] jan.osth@ri.se

Abstract

This paper presents Liaison a new open-source tool designed to address challenges related to portability, security, and intellectual property when sharing Functional Mock-up Units. Built on the Functional Mock-up Interface 3.0 standard, Liaison has a server-client architecture that leverages Zenoh for communication. Zenoh is a novel pub/sub/query protocol that supports a variety of network topologies such as peer-to-peer, routed, mesh and brokered communication with minimal configuration, setting Liaison apart from similar tools. An example of a case that led the authors to develop this tool is presented.

Keywords: distributed co-simulations, maritime onboard systems

1 Introduction

The Functional Mock-up Interface (FMI) (Blochwitz et. al, 2011, Blochwitz et. al, 2012, Junghanns et. al, 2021) is a widely adopted standard in system simulation, providing a versatile and vendor-neutral framework for model exchange (ME) and co-simulations (CS) (Gomes et. al, 2018). At its core, FMI defines a set of conventions and interfaces that govern how simulation models interact with simulation environments, allowing for the encapsulation of models as Functional Mock-up Units (FMU). A FMU encapsulates a simulation model along with their associated parameters, inputs, and outputs, as compiled code black-boxes enabling the exchange and utilization across simulation tools that has implemented support for the FMI standard. For ME, the FMU does not include the model's solver, whereas for CS, the FMU also includes the solver of the model.

In practice, several factors can make sharing FMUs unfeasible. Such factors can include, but are not limited to:

- Portability issues (e.g. different operating systems or license requirements for execution of the FMU)
- Intellectual Property (IP) protection.
- Fear of unintended re-distribution of the FMU
- Fear of a FMU containing malicious code.

To overcome these obstacles, a distributed approach can be utilized where the FMU stays within the control and premises of the model owner but connects to a simulation environment over a network when desired. The connection can be made over a local area network or a wide area network such as the Internet.

Table 1 summarizes the issues solved by running distributed co-simulations locally, in a LAN, and in a WAN.

Table 1. Potential issues solved by running distributed co-simulations.

Issue	*Local*	*LAN*	*WAN*
FMU portability	x	x	x
Security	x	x	x
IP protection			x

FMU portability can be addressed by running distributed co-simulations locally, in a LAN, and in a WAN. In both LAN and WAN setups, a computer with a suitable operating system, environment, or hardware, can be used to serve the FMU, while another computer can be used to run the simulation. Meanwhile, security and IP protection can be address by running distributed co-simulations on a WAN across organisations. In this use case, one organisation creates a client FMU from a source FMU and delivers it to another organisation. This approach enables co-simulations without the transfer of intellectual property or risk of running malicious code.

Several tools for such distributed co-simulations have been developed. The most relevant for the work presented in this paper, is the FMU-Proxy (Hatledal et al., 2019) which is an open-source framework for distributed co-simulations supporting FMI 1.0 and 2.0. The original FMU is wrapped into a proxy FMU that is accessed through Remote Procedure Calls (RPCs). In this way, the platform becomes independent of both the FMU implementation language as well as the platform it runs on. FMU-Proxy relies on a combination of different

communication protocols such as HTTP, HTTP2, Websockets and TCP/IP.

In this paper we present Liaison (Liaison 2025), a new open-source tool for distributed co-simulations based on the FMI 3.0 standard and the Zenoh communication protocol (Corsaro et al., 2023). The decision to develop a new tool for distributed co-simulations was motivated by that existing tools at the time of project start did not support FMI 3.0, and that the adoption of Zenoh promised a simple and efficient communication framework (Palma, 2024) supporting also geographically distributed nodes deployed across wide area networks with minimal required network settings configuration (such as opening ports on host and client computers, etc). Zenoh runs on top of traditional communication protocols such as TCP/IP and UDP/IP and transparently bridge across different protocols. It is a pub/sub/query protocol and supports a variety of network topologies such as peer-to-peer, routed, mesh and brokered communication.

This introduction is followed by Section 2 where we present the high-level architecture of Liaison, network topologies and describes its implementation. In Section 3 we present an example case that led the authors to develop this tool (Software-in-the-Loop testing of sail control systems for wind assisted propulsion of ships (Laursen et al. 2023)). Section 4 presents some results connected to the computational overhead for the different use cases and Section 5 presents the outlook and future work.

2 Liaison

Figures 1 to 3 present the three basic configurations on which Liaison can be used. In all the figures, an "original" FMU is made available to a simulation software (co-simulation engine) with a Liaison FMU and a Liaison Server. A Liaison FMU is a client application that substitutes the original FMU and communicates with a Liaison Server. A Liaison Server is a server application that forwards the function calls made to the Liaison FMU by the simulation software to the original FMU.

Figure 1 and 2 present the configurations for running a distributed co-simulation locally and over a LAN. Both configurations are useful for addressing security and portability issues. For example, the FMU can be run inside of a sandbox/Docker environment or in another computer with the appropriate operating system or hardware resources. Meanwhile, Figure 3 corresponds to running a distributed co-simulation in a WAN. In addition to addressing portability and security issues, this configuration also enables running co-simulation without sharing IP across organizations, avoiding the need for non-disclosure agreements, software licenses, obfuscation, and similar practices.

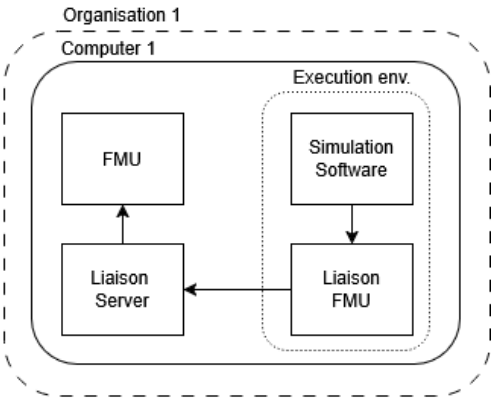


Figure 1. Configuration for running a distributed co-simulation locally with Liaison.

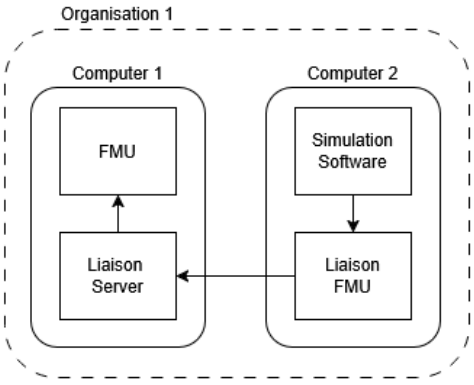


Figure 2. Configuration for running a distributed co-simulation over a LAN with Liaison.

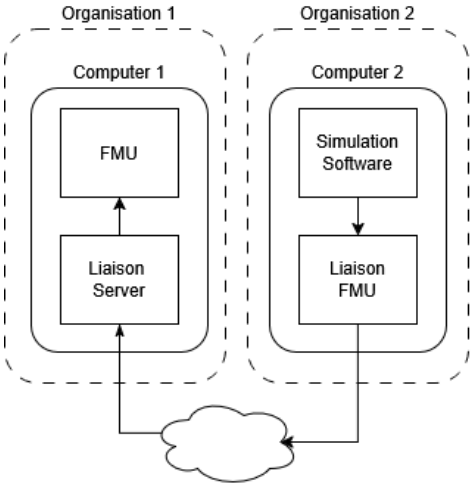


Figure 3. Configuration for running a distributed simulation over a WAN with Liaison.

2.1 Networking

Liaison uses Zenoh for communication. By default, Liaison operates using the "peer" mode in Zenoh which entails that:

- Every node discovers other nodes through a discovery mechanism.

DOI 10.3384/ecp218639

- Every node connects to and maintains a connection to all other nodes found through the discovery mechanism.

A schematic of this setup is illustrated in Figure 4. This setup is a convenient default as it allows for seamless, zero-configuration connections in a LAN.

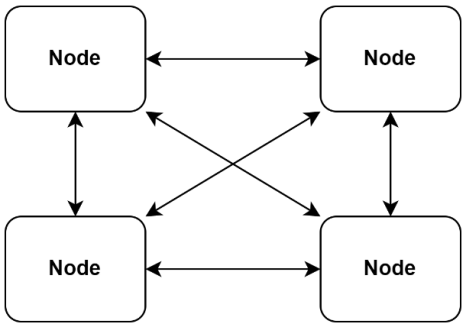


Figure 4. Schematic of 4 nodes connected in peer-to-peer mode. I.e. every node has a direct connection with every other node.

Further, Liaison allows for unimpeded configuration of the underlying Zenoh library. This allows for a range of different network topologies to be used, for details see (Corsaro et al., 2023). The most interesting one for the use of Liaison is the "client" mode which entails that:

- Every node connects to a well-known router on startup.
- The router ensures end-to-end connectivity in between all nodes connected to it.

A schematic of this setup is illustrated in Figure 5.

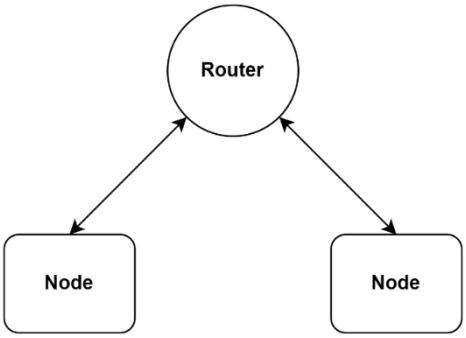


Figure 5. Schematic of two nodes connected in client mode via a (software) router. Each node maintains only a single connection to the router and the router ensures end-to-end connectivity in between nodes.

Although the "client" mode of communication in Zenoh requires every node to make an initial connection to a well-known router, it does not require every node to know the connectivity details (such as IP number and port) of all the other nodes in the network. Effectively, this allows for nodes in different LANs, possibly geographically distributed, to interconnect via a router deployed on a well-known address accessible over a WAN (such as the internet) without the need of knowing the connectivity details of each other. A schematic of this setup is outlined in Figure 6.

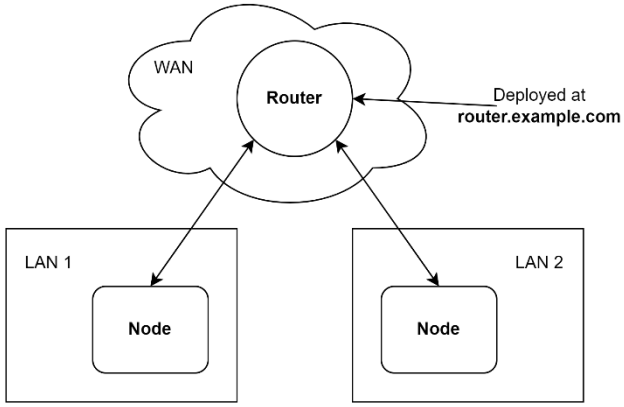


Figure 6. Schematic of 2 nodes connecting across a WAN from two separate LANs in client mode via a (software) router deployed on a well-known address.

Further, the underlying Zenoh library may be configured for secured communication channels, regardless of network topology, using either client/server TLS or mutual TLS.

2.2 Implementation

Liaison is an application written in C++ and compiled for Windows and Linux. It is released as open source under the Apache 2.0 license and is available for download at http://github.com/rise-maritime/liaison. While it depends on standard system libraries, third-party dependencies are bundled with the release, requiring no additional installation.

Liaison consists of a platform specific executable and a set of dynamic libraries that encapsulate the client logic for different platforms. The executable performs two main functions: 1) making a „Liaison FMU" from a standard FMU, and 2) serving an FMU. When making a Liaison FMU, the executable extracts the model description XML file from the source FMU, renames the dynamic libraries according to the model's name, and re-packages it all. In turn, when serving an FMU, the executable loads the source FMU and forwards to it all the incoming queries from the Liaison FMU as seen in Figures 1 to 3.

At the time of writing, only a subset of the FMI 3.0 standard is implemented. For an up to date list of the implemented functions the readers are referred to the project's online repository (Liaison, 2025).

3 Example: Software-In-The-Loop (SIL) for control of a Wind-Assisted Propulsion System

Consider the following made-up example: A ship owner is interested in retrofitting one of its vessels with a wind-assisted propulsion system (Laursen et al. 2023) and is particularly interested in rigid wings as suggested by a potential provider. But, prior to signing a deal, the shipowner wants to ensure the interoperability of the proposed wing sail system (and its controller) with the existing autopilot used on the ship. For that purpose, the ship owner demands a set of co-simulations to be performed to verify the interoperability in a Software-in-the-Loop (SIL) setup.

In this paper we will use the example above to exemplify the usefulness of Liaison in overcoming technical as well as organizational obstacles in sharing software, especially control systems, in the design process of ships. Further, we will use the same setup to investigate the performance penalty introduced by Liaison through different scenarios.

3.1 Simulation setup

Figure 7 presents the simulation setup with the following components:

- **Vessel FMU:** an FMU that encapsulates a hydrodynamic model of a cargo vessel as well as an aerodynamic model of a set of rigid wings mounted on said cargo vessel. The models used in this FMU are built using the RISE in-house ship dynamics solver SEAMAN (Ottosson and Byström, 1991).
- **Controller FMU**: an FMU that encapsulates the controller logic for the wing controller system.
- **Co-simulation engine:** A co-simulation engine that is responsible for stepping the co-simulation forward and handling connections between the two FMUs. The co-simulation engine used for these simulations is Ecos, for details see (Ecos 2025).

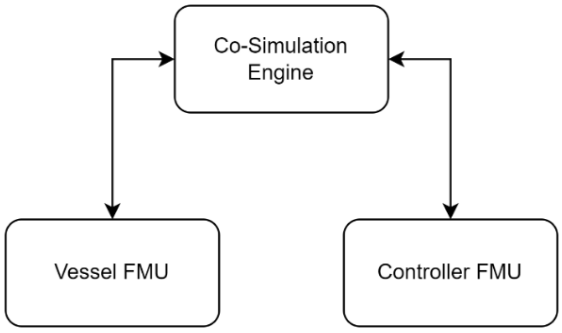


Figure 7. Simulation setup for the example.

The FMUs used in the simulation setup are deliberately kept at minimum level of complexity, using only the bare minimum of the FMI 3.0 specification, to ensure fair comparisons focusing on the performance penalty introduced by Liaison.

3.2 Scenarios

The simulated setup is used in the following scenarios:

- *Reference scenario:* the FMUs are loaded directly by the co-simulation engine, i.e. Liaison is not used.
- *Local scenario:* the FMUs and the co-simulation engine runs on the same physical computer, but Liaison is used as the means of communication.
- *LAN scenario*: the FMUs and the co-simulation engine run on different physical computers (3 computers in total), but in the same LAN.
- *WAN scenario:* the FMUs and the co-simulation engine all run on different physical computers (3 computers in total) on different LANs interconnected via a WAN (the Internet). The communication channels are secured using mTLS.

To investigate the performance penalty of using Liaison, each of the scenarios are run for 600 seconds with three different step sizes: 0.1, 1.0, and 10.0 seconds. To account for variations in execution time, each scenario and step size combination is simulated 10 times, and the average time is recorded.

4 Results

Figure 8 presents the performance penalty associated with using Liaison across various configurations and communication step sizes.

For a communication step size of 0.1 seconds:

- The reference scenario has an average execution time of about 25 seconds.
- The local scenario has an average execution time of 1.4 times that of the reference scenario.
- The LAN configuration scenario has an average execution time that is more than double (2.4 times) that of the reference scenario.
- The WAN scenario has a significantly higher average execution time, 7.6 times that of the reference scenario.

For a communication step size of 1.0 second:

- The reference, local, and WAN scenarios have similar execution times at about 25 seconds.
- The WAN scenario has a higher average execution time, about 1.7 times that of the other scenarios.

For a communication step size of 10.0 seconds:

- All cases exhibit similar average execution times, approximately 25 seconds.

DOI 10.3384/ecp218639

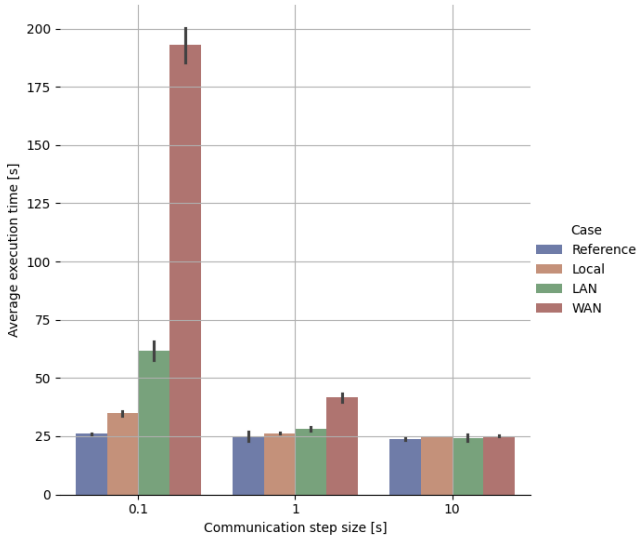


Figure 8. Performance results from different co-simulation scenarios and configurations.

These results indicate that the communication step size has a substantial impact on execution time, particularly in the WAN scenario. The performance is notably better in scenarios with larger communication step sizes and when the components are either on the same physical computer or within the same LAN.

5 Conclusion and future work

Liaison is a general open-source tool for distributed co-simulations and model exchange based on FMI 3.0. The workflow when using Liaison starts with an existing FMU. In the first step, Liaison is used to produce a "Liaison FMU", that can be included in any simulation tool that supports FMI 3.0. For the simulation tool, the Liaison FMU appears as a standard FMU. The Liaison FMU acts as a client and includes all information it needs to communicate with a Liaison server over LAN or WAN (such as the Internet). The communication protocol in Liaison is the Zenoh protocol, which ensures a simple communication configuration by the user and versatile network topologies.

Results presented in this paper shows that running simulations in a distributed fashions with Liaison can have a negative impact on simulation time due to communication events between the distributed Liaison FMUs that are part of the distributed co-simulation. However, this performance drawback can be mitigated by selecting appropriate communication step sizes.

Future work will focus on achieving full compatibility with FMI 3.0 and include support for FMI 2.0, as well as testing Liaison in various simulation software environments. Contributions are welcome!

Acknowledgements

This work is part of the project "Virtual Wind Ship", carried out within the Swedish Transport Administration's industry program Sustainable Shipping, operated by Lighthouse".

References

Blochwitz, Torsten; Martin Otter; Martin Arnold, et al. (2011-03). "The functional mockup interface for tool independent exchange of simulation models." In: *Proceedings of the 8th International Modelica Conference.* Linköping University Press, pp. 105-114.

Blochwitz, Torsten; Martin Otter; Johan R. Åkesson; et al. (2012-09). "Functional mockup interface 2.0: The standard for tool independent exchange of simulation models". In: *Proceedings of the 9th International Modelica Conference.* The Modelica Association, pp. 173-184.

Corsaro, Angelo; Luca Cominardi; Olivier Hecart; et al. (2023-09) "Zenoh: Unifying communication, storage and computation from the cloud to the microcontroller." In*: 26th Euromicro Conference on Digital System Design (DSD*). IEEE pp. 422-428.

Ecos (2025). Ecos co-simulation platform. Code repository. URL: *https://github.com/Ecos-platform/ecos (visited on 2025-07-30)*

Gomes, Cláudio; Casper Thule; David Broman; et al. (2018). "Co-simulation: a survey". In: *ACM Computing Surveys (CSUR)* 51(3):49, pp. 1-33.

Hatledal, Lars; Ivar Arne Styve; Geir Hovland and Houxiang Zhang (2019). "A language and platform independent co-simulation framework based on the functional mock-up interface". IEEE Access, Vol. 7 pp. 109328-109339. DOI: 10.1109/ACCESS.2019.2933275

Giese, Holger Giese; Dominique Blouin; Rima Al-Ali; et al. (2021). "Chapter 4 - An ontology for multi-paradigm modelling." Multi-Paradigm Modelling Approaches for Cyber-Physical Systems, Elsevier. ISBN; 978-0-12-819105-7

Junghanns, Andreas; Cláudio Gomes; Christian Schulze; et al. (2021-09). "The functional mock-up interface 3.0-new features enabling new applications." In: *Proceedings of the 14th Modelica Conference*, Linköping University Electronic Press, pp. 17–26. DOI: 10.3384/ecp2118117

Kolodziejski, Michal and Marcin Sosnows;ki (2025). "Review of Wind-Assisted Propulsion Systems in

Maritime Transport". Energies, 18(4):897. DOI: 10.3390/en18040897

Laursen, Rasmus; Hitesh Patel; Dimitra Sofiadi; et al. (2023). "Potential of Wind-Assisted Propulsion for Shipping." Tech. rep. *European Maritime Safety Agency (EMSA Report EMSA/OP/43/2020),* pp. 1-271

Liaison (2025). Liaison co-simulation tool. Code repository. URL: *https://github.com/RISE-Maritime/liaison (visited on 2025-07-30).*

Ottosson, Peter and Lars Byström (1991). "Simulation of the Dynamics of a Ship Manoeuvring in Waves". Transactions of the Society of Naval Architects and Marine Engineers (Trans. SNAME) 99, pp. 281–298

Palma, Steven (2024), "Transforming the robotics industry: A sustainable communication protocol". Control Engineering. URL: https://www.controleng.com/transforming-the-robotics-industry-a-sustainable-communication-protocol (visited on 2025-07-30).

A Tool for the Implementation of Open Neural Network Exchange Models in Functional Mockup Units

Michele Urbani[1] Michele Bolognese[1] Luca Pratticò[1] Matteo Testi[1]

[1]Hydrogen Technologies and Resilient Energy Systems Unit, Sustainable Energy Centre, Fondazione Bruno Kessler, Via Sommarive 18, 38123, Trento, Italy, {murbani,mbolognese,prattico,testi}@fbk.eu

Abstract

The Functional Mock-up Interface (FMI) standard is a flagship in the co-simulation and model exchange domain. However, the integration of graph-based computational models—particularly neural networks—into Functional Mock-up Units (FMUs) has remained a technical challenge due to interoperability and platform-specific limitations. To address this, we propose ONNX2FMU, a command-line Python tool that facilitates the deployment of Open Neural Network Exchange (ONNX) models into FMUs. According to FMI's good practices, ONNX2FMU generates C source code to wrap ONNX models in Functional Mockup Units, supports FMI versions 2.0 and 3.0, and provides multi-platform compilation capabilities. The tool simplifies the mapping process between model description and ONNX model inputs and outputs via JSON files, ensuring accessibility and flexibility. This paper presents the tool architecture and methodology and showcases its applicability through illustrative examples, including a reduced-order model powered by a recurrent neural network.

Keywords: ONNX, FMI, FMU, ML, AI

1 Introduction

Machine learning (ML) and deep learning (DL) have gained momentum in engineering and science in recent years. There are countless applications of inferential models in engineering: For example, in engineering and manufacturing (Malik et al. 2018), process system engineering (Lee, Shin, and Realff 2018), chemical engineering (Schweidtmann et al. 2021), smart energy systems (Lund et al. 2017), and fluid mechanics (Brunton, Noack, and Koumoutsakos 2020). Better and more efficient computational infrastructures, together with a never-seen-before amount of data, call for the exploitation of these sources of information in simulations (Rueden et al. 2020; Bertsch, Jarmolowitz, and Lenord 2024). However, incorporating ML models into FMUs requires specialised tooling. The existence of several ML frameworks, each with its own scope and peculiarities, advocates for a standard model exchange format. The Open Neural Network Exchange (ONNX) format (*Open Neural Network Exchange* 2025) enables one to describe an ML model through a set of standard operators and ships with its runtime. Several ML frameworks implement their methods to export a model to the ONNX format, thus partly bridging the gap of model interoperability. Therefore, the challenge is developing a tool to write the C code that maps the model's input and outputs to the FMI's C API specifications. Software licensing must guarantee the usability of the tool for many purposes to foster the widespread adoption of inferential models in simulations.

Commercial tools that can wrap ML models into FMUs already exist. Simcenter ROM (*Simcenter Reduced Order Modeling software* 2025) is a software for developing Reduced Order Models (ROM) that presents the FMU export feature, which allows a user to wrap an ML-based ROM in an FMU. Since ML is simply a model order reduction technique (among a few others) in Simcenter ROM, the choice of the ML framework is arbitrary and, to the eye of the user, it is irrelevant as far as the model's accuracy is satisfactory. The generation of an FMU is, therefore, purely functional and does not allow the user any customisation. SimuLink (*Simulink - Simulation and Model-Based Design* 2025) is a commercial software that exports simulation models as FMUs by generating the C code from a set of proprietary primitives. Neural networks, a type of ML model, can be included in a Simulink simulation workflow and shipped in an FMU. The software suite (*NEURECO: Automatic Parsimonious Neural Networks Generation* 2025) presents itself as the *automatic parsimonious neural networks factory*. It can produce neural network-based models and export them to C code, ONNX, and FMUs.

Software licensing is a key feature when selecting a tool. The availability of open-source, open-access software is always valuable to the community. Most open-source software is hosted on platforms like GitHub and GitLab, where we carried out a structured search for ML to FMU wrapping tools. The keywords used for the search were "ONNX" and "FMU", which returned only one relevant result on the GitHub platform. The search results are confirmed by a check of the "Tools" section of the FMI website. MLFMU (*mlfmu* 2025) is the only open-source, open-access tool that enables wrapping ML models to FMUs, and a team of contributors from DNV develops it. The MLFMU's workflow foresees the possibility of exporting an ONNX model with only one input and one output array, to which the physical model quantities need

to be mapped. Input and output mapping is declared using a JSON file, which plays the role of a user interface. The core functionality of MLFMU is the generation of the FMI model description and the C++ code generator to link the ONNX runtime to the FMU. For the generation of C++ code, MLFMU rely on another open source project called "cppfmu". Conan is used as a package manager.

Proposing a new tool for integrating ML models in FMU co-simulation is important because it enables the use of graph-based computational models, e.g., ML models and neural networks. Furthermore, the proposed tool would be a valuable open-access, open-source resource to the Modelica community.

We introduce ONNX2FMU, a command-line interface (CLI)-based Python tool that generates C source code for embedding ONNX models into FMUs, which can be compiled on any target platform. The tool bridges the gap between ML model deployment and FMI-compliant simulation, offering an efficient and user-friendly pipeline. Compared to the existing alternatives, ONNX2FMU:

- Directly generates C source code, improving performance and control over the generation process. Although the use of the C programming language is not mandatory, we preferred to stick with the FMI Standard's indication that the "FMI API is written in C, not C++, to avoid problems with compiler and linker dependent behaviour, and to enable the use of FMUs on embedded systems" (Modelica Association 2025).
- In addition to the FMI 2.0, it supports FMI 3.0, including extended data types.
- Handles models with multiple inputs and outputs mapped to the FMU model according to an arbitrary naming convention.

Generating C code directly exposes the user to the source code in a single file where FMI "get" and "set" functions are declared together with the link to the ONNX runtime. ONNX2FMU is designed for developers who want to bootstrap the creation of FMUs to deliver graph-based computational models, data scientists and researchers operating in model order reduction, and DevOps who need to automate the creation of pipelines of tools. Being distributed as open-source, the tool is highly customizable, and it allows the separation of the C code generation stage and the compilation stage, thus enabling the inspection and modification of the FMU source code. The tool comes with a set of unit tests and a pipeline for continuous integration that guarantees the integrity of the code and its cross-platform compatibility.

An analysis of requirements for the proposed tool and the adopted methodology for creating the tool is outlined in Section 2. Section 3 presents the results and shows two examples where ONNX2FMU is applied. Conclusions close the paper in Section 4.

2 Methodology

Requirements analysis The inspiring principle for the creation of ONNX2FMU was the simplicity of use. This means relieving the user from making many decisions during the tool usage, reducing the number of steps to the final result, and designing a clear interface for mapping ONNX model inputs and outputs to the FMI model description.

FMI standard compliance. The FMI Standard is now in its third major release (Junghanns et al. 2021), the FMI 3.0 version, which differs from the FMI 2.0 version in several aspects. In FMI 3.0, the *scheduled execution* interface type has been added to the *model exchange* and *co-simulation* types, allowing the activation of individual model partitions by an external scheduler. Model partitioning capabilities for concurrency on single-core CPUs are not of interest for executing ML models in a simulation; therefore, we excluded the possibility of generating the scheduled execution interface type. In general, the execution of ML models does not require the FMU to be equipped with a solver. A straightforward choice is hence to implement the model exchange interface type. Since a few modelling environments, e.g., Modelon Impact, allow us to import only co-simulation FMUs, the usability range of a tool for ML model encapsulation is significantly widened by implementing the co-simulation interface. The co-simulation interface brings the burden of having a solver within it that remains unused.

In the first version of ONNX2FMU, we focus on the FMI 2.0 and 3.0 Standard versions. The FMI 2.0 is the industry standard, supported by more than 90% of the tools published on the FMI Standard website[1]. Supporting the FMI 3.0 contributes to the state of the art in ML model implementation in the FMI standard. The two versions of the standard present meaningful differences in the data types that can be specified in the model description: The FMI 2.0 supports only four data types, i.e., float, integer, string, and boolean, whereas the FMI 3.0 extends these data types to support a broader spectrum of data precision and signature. Relevant data types to the scope of this project are the `float` type for the FMI 2.0 and the `Float32` and `Float64` types for the FMI 3.0. All other data types are considered according to the ONNX specifications discussed in the next paragraph. The variability and causality of variables are restricted to the combinations allowed by the standard. If not specified by the user, the causality of model entries is determined by their connections in the ONNX graph model, which divides entries into inputs and outputs. The available variability options for input and output causalities are "discrete" and "continuous". The FMU developer should carefully set the appropriate variability according to the behaviour of the model to be encapsulated; in the co-simulation mode, the capability flag `providesIntermediateUpdate` should be set accordingly. The combinations of platforms and operating

[1] https://fmi-standard.org/tools/

DOI 10.3384/ecp218645

systems in the following are generally supported:

- win32, win64, linux32, linux64, and aarch64-darwin for the FMI 2.0;
- x86_64-darwin, aarch64-darwin, x86-linux, x86_64-linux, aarch64-linux, x86-windows, x86_64-windows, and aarch64-windows for the FMI 3.0.

ONNX standard. The Open Neural Network Exchange (*Open Neural Network Exchange* 2025) is a standard for the interoperability and exchange of ML models. It defines a set of operators typically used in ML and DL models and a common format for moving models through different ML frameworks, tools, runtimes, and compilers.

Deploying an ML model to production requires replicating part of the ecosystem used to train the model. We refer to this ecosystem using the word *runtime*, that is, the code that enables running inference (and training) on a target hardware platform. In addition to guaranteeing interoperability, converting a model to the ONNX format allows developers to use optimised runtime libraries from different providers, see, e.g., those listed in (*Open Neural Network Exchange* 2025). A runtime can be defined in any language suitable for the production application, C, Java, Python, JavaScript, C#, WebAssembly, or ARM. We chose the ONNX runtime among the many available options because of its licensing, the large community of contributors to the project, and the availability of C code examples that allowed us to bootstrap the development of the tool.

Data types conversions. ML models require support for floating point numbers in single or double precision, and eventually, they may include other data types. An ML model supporting single-precision floating point numbers requires casting double-precision inputs/outputs, with a potential loss of accuracy of the model. The model data type dictates how the tool must manage data types internally: If a model is single-precision, a double-precision float must be converted before and after inference; otherwise, no casting operations are required. The same behaviour is adopted for integers.

Inputs, outputs, and naming conventions. A generic ML model can have multiple inputs and output nodes, whose elements must be mapped to the FMI model description. The FMI 2.0 version foresees only scalar variables in the model description. Therefore, arrays must be flattened, and a naming convention is required to map scalars to their original position. A logical choice seems to use the array name suffixed with the element indexes separated by a specific character. Conversely, the FMI 3.0 allows both the declaration of scalar and array variables in the model description. Declaring array variables produces a more compact and comprehensible model description, but poses a hurdle to mapping variables to their physical meaning.

User experience. The interaction with the tool should consist of two subsequent stages. The mapping of the ONNX model entries to the FMI model description occurs first. During this stage, the user can define the variability associated with ONNX model arrays, if different from `continuous`, and, optionally, a description of each element of an array. The latter will likely be shown by the graphical user interface (GUI) of the importer software, thus easing the connection of the model with other components of the simulation.

Once a model mapping scheme and an ONNX model are available, an FMU model can be generated. There are many possible approaches to FMU generation, ranging from a GUI to Python scripting. A GUI would consist of only a few buttons to trigger the generation process, and it would be more effective if it integrated the model mapping declaration. Interaction via CLI is simple to develop and effective in integrating within a pipeline of operations. FMU generation via Python scripting is always possible by calling the tool's functions directly.

3 Results

The result of this research is ONNX2FMU. The design choices are explained in the following with examples.

3.1 Getting started

ONNX2FMUis written in Python to ease code usability, interaction with ONNX models, and to exploit a suite of available open source tools. A Python interpreter is required, and ONNX2FMU has been tested for versions greater than or equal to 3.10. ONNX2FMU is distributed as a Python package available in PyPI (*PyPI, Python Package Index* 2025), and can be installed with

```
pip install onnx2fmu
```

from the command line. To generate FMUs, ONNX2FMU requires a C compiler for the target platform to be installed on the machine and added to the system `PATH` variable.

3.2 FMU model mapping

ONNX2FMU can handle ONNX models presenting multiple input and output nodes. The description of entry variables is organised hierarchically, including global model information and input/output node descriptions. The JSON file format has been chosen for model mapping due to its suitability in representing hierarchically organised information. In the input and output sections, nodes are identified using the same name used in the ONNX model; this allows ONNX2FMU to map the information correctly. A user can specify the variability of each node, which is set to continuous by default. Node variability should be handled carefully because it might cause sequence-based models to return wrong results when they are not called with the right time step. Array elements can be enriched with a description, which is shown in the FMI

model description and might be shown by the importer UI. Descriptions of entries help the user to link them with their physical meaning. If the list of descriptions is provided, it must match the elements in the array; otherwise, a warning is returned, and no description will be present for the elements of that array. The array notation provided by the FMI 3.0 Standard is left out of the scope of this work because it makes it difficult to connect array entries with their physical meaning. However, the authors acknowledge that array notation could be useful when an ONNX model presents several inputs and outputs; for this reason, it will be addressed in a future release of ONNX2FMU.

3.3 FMU build

The FMU build workflow starts with the target model to encapsulate and the model mapping JSON file. If not specified, the target platform is that of the system where the build process is carried out. A template of files and folders to be populated with ONNX model information was developed. The Python interpreter generates the C code by creating a copy of the FMU template and filling it using Jinja2 (*Jinja Project* 2025), a template engine that populates special placeholders in text files with context variables and renders the final document. The CMake build system (*CMake: A Powerful Software Build System* 2025) is used to build the project, download the latest ONNX runtime version, compile the library, and zip it to an FMI-compliant archive. For separation of concerns, ONNX2FMU takes care of template filling and ONNX model information retrieval using Python, whereas compilation issues and packaging are handled by CMake, which natively supports the build and compilation toolchain.

The Python CLI is realised using *typer* (FastAPI 2025), a Python package that creates simple command line applications with documentation, autocompletion, and typing support.

Validation and testing are the last two steps of ONNX2FMU's workflow, which is depicted in Figure 1. The automated validation of the FMU is made possible by the FMPy (*FMPy* 2025) package. Testing a generated FMU can hardly be automated because of the case-specific nature of ML models, which requires appropriate data.

3.4 Examples

Use cases are presented to show the functionalities and the potential of ONNX2FMU in the following paragraphs.

A simple model. To showcase the flexibility of ONNX2FMU, we present how a generic graph-based model can be wrapped in an FMU. The simple model takes three scalar inputs x_1, x_2, x_3 and returns a vector output y. In the model, two calculations occur: $x_4 = x_1 + x_2$ and $x_5 = x_2 - x_3$, and the results are concatenated in $y = [x_4, x_5]$ and returned as output. A graphical representation of the model is depicted in Figure 2, which was obtained using the Netron web app[2]. A Python pseudo-code example for creating the model using the PyTorch framework is reported in Listing 1.

Listing 1. Python code for the creation of a simple graph-based computational model using PyTorch.

```
class ExampleModel(nn.Module):

  def __init__(self):
    super(ExampleModel, self).__init__()

  def forward(self, x1, x2, x3):
    x4 = x1 + x2
    x5 = x2 - x3
    x = torch.cat([x4, x5], dim=0)
    return x
```

Exporting the model is easy in PyTorch as in other ML frameworks and usually, it can be done in a few lines of code. Models can be exported to the ONNX format with or without weights; ONNX aims to define the model structure primarily through its set of primitives. ONNX models are not necessarily shipped with weights either because they do not need them, as in the case of the model above, or because weights might be application-specific.

Once the ONNX model has been exported, inputs and outputs are mapped to the FMU model description. This is achieved through the JSON schema in Listing 2.

Listing 2. The JSON mapping between inputs and outputs of the simple model.

```
{
"name": "simple-model",
"description": "Example description.",
"FMIVersion": "2.0",
"input": [
    {"name": "x_1",
     "description": "A scalar input"},
    {"name": "x_2",
     "description": "A scalar input"},
    {"name": "x_3",
     "description": "A scalar input"}],
"output": [
    {"name": "y",
     "description": "The output array",
     "names": ["output1", "output2"]}]
}
```

Notice that the output array schema presents the `"names"` item, which allows users to describe each array entry in the FMI model description. Finally, FMU generation is triggered via the command line by providing the ONNX model and description as follows

```
onnx2fmu build \\
    model.onnx \\
    modelDescription.json
```

Time series prediction. Predicting temperature variations in a solid exposed to a heat source is a common engineering problem. A partial differential equation (PDE)-based model (Bergman 2011) can describe a temperature field variation in, e.g., an ideal 2D solid of interest. The

[2] https://netron.app/

DOI 10.3384/ecp218645

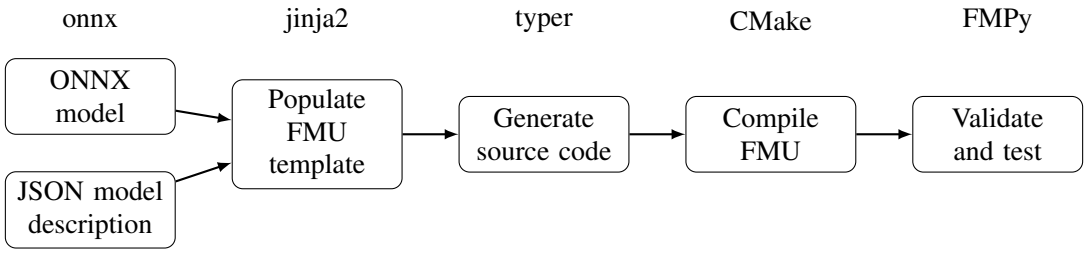


Figure 1. The workflow to embed an ONNX model in an FMU with ONNX2FMU.

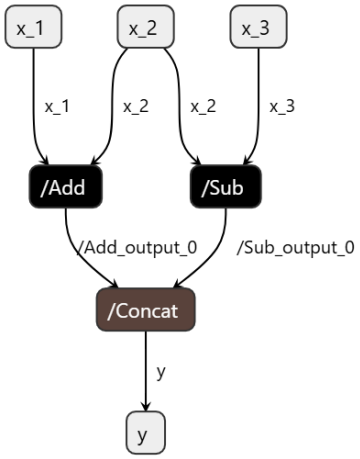


Figure 2. A simple graph-based model for summation and subtraction of inputs.

temperature field equation is

$$\frac{\partial T}{\partial t} = \alpha \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) + T_0, \tag{1}$$

where T is the temperature, t is the time, α is the thermal conductivity of the material, and x and y are the coordinates of a point in the solid. Equation 1 is discretised in time and space and resolved by a custom Python model that is reported in the ONNX2FMU documentation[3]. In the following, we define $\alpha = 0.9$, an initial temperature $T_0 = 100°\text{C}$, and $x, y \in [0, 1]$.

In some engineering applications, we may be interested in predicting the temperature variation at a point of the solid, e.g., $x_a = 0.5, y_a = 0.5$. A surrogate model of the studied solid would avoid resolving the full-scale model to know the temperature only in (x_a, y_a), thus saving a meaningful computational effort. A simple way to create a surrogate model for predicting temperature variations is to train a neural network using a time series of heat source variation $u(t)$ as inputs and temperature variations $y(t)$ as outputs. We use a Recurrent Neural Network (RNN) (Bengio, Goodfellow, Courville, et al. 2017) to extrapolate the temperature in (x_a, y_a). A dataset of temperature trajectories $y(t)$ in (x_a, y_a) is generated resolving (1) with the boundary condition

$$u(t) = (-1)^j 20j \sin(t/8 + j\pi/8) + T_0,$$

applied on the side with $x = 0.0$, where j is the index of a trial input profile, for $t \in [0, 25]$ seconds. An example of the applied temperature profile $u(t)$ and temperature profile $y(t)$ in (x_a, y_a) obtained by simulating the system using the custom Python script is depicted in Figure 3.

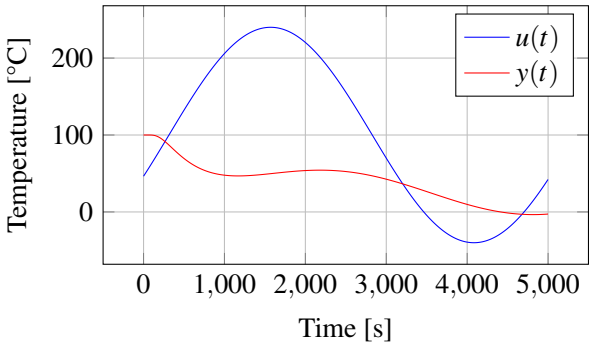


Figure 3. Input temperature profile and the temperature in the centre of the plate.

The model was trained on a dataset with $M = 40$ trajectories, each with 5000 time points sampled at a 1/200 s^{-1} frequency. Data were organised in tuples $(\mathbf{x}, y)$, where $\mathbf{x}$ was a sequence with $R = 200$ consecutive time points, and y was the $(R+1)$-th time point. The mean squared error

$$MSE = \frac{1}{N} \sum_{i}^{N} (y_i - \hat{y}_i)^2$$

measures the predictive power of the RNN, where $\hat{y}_i$ is the value predicted by the RNN, i is the sample index, and N is the number of samples. The efficiency of the RNN model improves if normalised data are used for training. However, it is impractical to deal with normalisation issues in a simulation environment; it would be more useful to provide unscaled data directly with the model. This issue can be easily resolved by adding normalisation and denormalisation steps to the network as shown in Listing 3.

Listing 3. Python code of the RNN model.

```
class HeatRNN(nn.Module):
    def __init__(self, input_size,
        hidden_size, output_size,
        num_layers, norm_params):
```

[3] github.com/HyRES-FBK/onnx2fmu/

```
        super(HeatRNN, self).__init__()
        x_min, x_max, y_min, y_max =
            norm_params
        self.register_buffer("x_min", torch
            .tensor(x_min))
        self.register_buffer("x_max", torch
            .tensor(x_max))
        self.register_buffer("y_min", torch
            .tensor(y_min))
        self.register_buffer("y_max", torch
            .tensor(y_max))
        self.rnn = nn.RNN(input_size,
            hidden_size, num_layers,
            batch_first=True)
        self.fc = nn.Linear(hidden_size,
            output_size)

    def forward(self, x, h=None):
        x = (x - self.x_min) / (self.x_max
            - self.x_min)
        out, _ = self.rnn(x, h)
        out = self.fc(out)
        out = out * (self.y_max - self.
            y_min) + self.y_min
        return out
```

In the code Listing 3, the normalisation parameters are passed as inputs when the network is initialised and stored as *buffers*, that is, they will be part of the network state when exported to the ONNX format, and they will remain untouched by PyTorch optimization algorithms during training. The RNN is initialised with 2 hidden layers, an input size of R, and a 64-unit hidden layer. The RNN output is further processed by a linear layer that maps the RNN output to the output scalar. Figure 4 shows the behaviour of the average value of *MSE* calculated over the training and validation data sets with the number of training epochs. The effectiveness of the trained model was satisfactory for this example after 1000 epochs using the Adam optimiser with a learning rate of 10^{-5}.

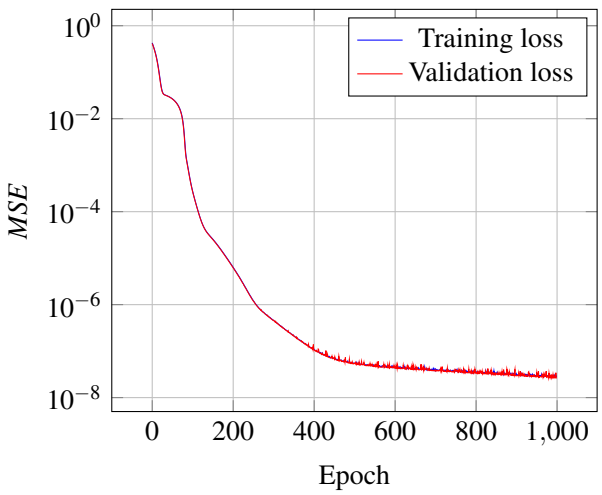


Figure 4. The *MSE* behavior through training epochs.

Finally, the trained RNN model is exported in ONNX format and encapsulated in an FMU for testing. The FMU was tested using FMPy (*FMPy* 2025), a Python toolkit for FMU validation and simulation. An example of the prediction produced by the RNN model is shown in Figure 5. In FMPy, a model can be simulated by providing inputs

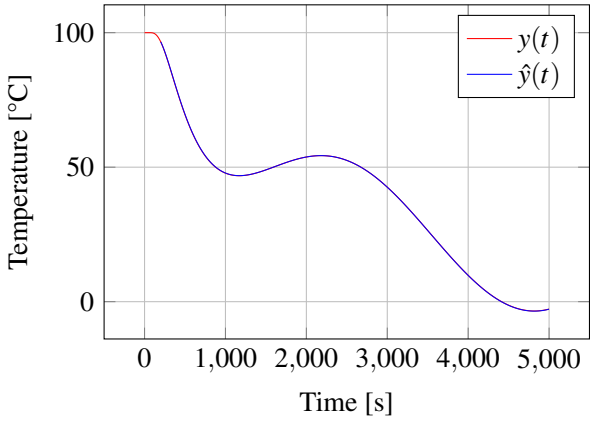


Figure 5. An example of the predicted and analytically obtained temperature profiles in (x_a, y_a).

directly from a text (CSV) file, which simplifies simulation setup when the RNN model presents $R = 200$ inputs. The recursion in the RNN model is carried out by providing R times the number of model variable values at each time step. However, we acknowledge that:

- the length of the input array, $R = 200$, makes the initialisation of such a model impractical and time-consuming in a GUI-based simulator, and
- the repetition of values along rows is memory-wise inefficient.

This aspect will be improved in the next major release of ONNX2FMUby implementing feedback loops declared in the model description.

In this simple example, we showed how a powerful model such as an RNN can be wrapped in an FMU with a few lines of code. The code of the example is released in GitHub[4].

4 Concluding remarks

We presented ONNX2FMU, a tool for converting ONNX models into FMI-compliant FMUs. The tool supports multiple FMI versions and cross-platform compilation and simplifies the deployment of ML models in simulation environments. Future work includes optimising runtime performance, integrating a GUI to favour a broader tool adoption, implementing FMI 3.0 array notation, and develop support for stateful models using feedback loops directly through the model declaration file.

[4] github.com/HyRES-FBK/onnx2fmu/blob/master/tests/example3/generate-example-model.ipynb

DOI 10.3384/ecp218645

References

Bengio, Yoshua, Ian Goodfellow, Aaron Courville, et al. (2017). *Deep learning*. Vol. 1. MIT press Cambridge, MA, USA.

Bergman, Theodore L (2011). *Fundamentals of heat and mass transfer*. John Wiley & Sons.

Bertsch, Christian, Fabian Jarmolowitz, and Oliver Lenord (2024). "Using the Functional Mock-up Interface (FMI) for AI applications". In: *Center for Model-Based Cyber-Physical Product Development* 33.18. DOI: 10.13140/RG.2.2.35730.47043.

Brunton, Steven L., Bernd R. Noack, and Petros Koumoutsakos (2020). "Machine Learning for Fluid Mechanics". In: *Annual Review of Fluid Mechanics* 52.Volume 52, 2020, pp. 477–508. ISSN: 1545-4479. DOI: 10.1146/annurev-fluid-010719-060214.

CMake: A Powerful Software Build System (2025). URL: https://cmake.org/ (visited on 2025-03-26).

FastAPI (2025). *Typer: A library for building CLI applications*. URL: https://typer.tiangolo.com/. (accessed: 28.04.2025).

FMPy (2025). URL: https://fmpy.readthedocs.io/en/latest/ (visited on 2025-03-27).

Jinja Project (2025). URL: https://jinja.palletsprojects.com/en/stable/ (visited on 2025-03-26).

Junghanns, Andreas et al. (2021). "The functional mock-up interface 3.0-new features enabling new applications". In: *Modelica conferences*, pp. 17–26. DOI: 10.3384/ecp2118117.

Lee, Jay H., Joohyun Shin, and Matthew J. Realff (2018). "Machine learning: Overview of the recent progresses and implications for the process systems engineering field". In: *Computers & Chemical Engineering* 114. FOCAPO/CPC 2017, pp. 111–121. ISSN: 0098-1354. DOI: 10.1016/j.compchemeng.2017.10.008.

Lund, Henrik et al. (2017). "Smart energy and smart energy systems". In: *Energy* 137, pp. 556–565. ISSN: 0360-5442. DOI: 10.1016/j.energy.2017.05.123.

Malik, Hasmat et al. (2018). "Applications of artificial intelligence techniques in engineering". In: *Sigma* 1. DOI: https://doi.org/10.1007/s10845-021-01771-6.

mlfmu (2025). URL: https://github.com/dnv-opensource/mlfmu (visited on 2025-04-06).

Modelica Association (2025). *Functional Mock-up Interface Specification*. URL: https://fmi-standard.org/docs/3.0/. (accessed: 24.03.2025).

NEURECO: Automatic Parsimonious Neural Networks Generation (2025). URL: https://www.adagos.com/neureco (visited on 2025-03-27).

Open Neural Network Exchange (2025). URL: https://onnx.ai/ (visited on 2025-04-06).

PyPI, Python Package Index (2025). URL: https://pypi.org/ (visited on 2025-03-26).

Rueden, Laura von et al. (2020). "Combining Machine Learning and Simulation to a Hybrid Modelling Approach: Current and Future Directions". In: *Advances in Intelligent Data Analysis XVIII*. Ed. by Michael R. Berthold, Ad Feelders, and Georg Krempl. Cham: Springer International Publishing, pp. 548–560. ISBN: 978-3-030-44584-3.

Schweidtmann, Artur M. et al. (2021). "Machine Learning in Chemical Engineering: A Perspective". In: *Chemie Ingenieur Technik* 93.12, pp. 2029–2039. DOI: 10.1002/cite.202100083.

Simcenter Reduced Order Modeling software (2025). URL: https://plm.sw.siemens.com/en-US/simcenter/integration-solutions/reduced-order-modeling/ (visited on 2025-03-27).

Simulink - Simulation and Model-Based Design (2025). URL: https://www.mathworks.com/products/simulink.html (visited on 2025-03-27).

DOI
10.3384/ecp218645

Input Smoothing for Faster Co-Simulation using FMI

Erik Henningsson[1] Christian Schulze[2] Manuel Gräber[3] Elmir Nahodovic[1] Dag Brück[1]
Julius Aka[4] Oliver Lenord[5]

[1]Dassault Systèmes AB, Sweden, {Erik.Henningsson,Elmir.Nahodovic,Dag.Brueck}@3ds.com
[2]TLK-Thermo GmbH, Germany, c.schulze@tlk-thermo.com
[3]TLK Energy GmbH, Germany, manuel.graeber@tlk-energy.de
[4]University of Augsburg, Germany, julius.aka@uni-a.de
[5]Robert Bosch GmbH, Germany, oliver.lenord@de.bosch.com

Abstract

We present two methods for speeding up co-simulations under the FMI standards. By smoothing the input signals inside each FMU, the internal integrator may avoid re-initialization. This can significantly reduce the number of model and Jacobian evaluations. To further help the integrator we also propose a predictor compensation technique tailored to the input smoother. The main benefit of our methods is the ease-of-use, requiring no model manipulations, nor any special co-simulation master algorithms. The methods are implemented in Dymola 2025x and validated with both an academic mechanical model as well as industrial thermo-fluid examples where we often observe over 10 times faster co-simulation and, in the best cases, up to 400 times faster. One of these thermo-fluid examples is used in the *OpenSCALING* research project to generate training data for constructing surrogate models, for which the input smoothing is especially important to speed up the dataset creation. *Keywords: FMI, Efficient co-simulation, Input smoothing, Thermo-fluid*

1 Introduction

To facilitate simulation of complex and large systems spanning over multiple technical domains, co-simulation techniques are commonly employed. This may, for example, include the integration of several domain-specific tools into one single simulation.

In this article we present two methods for speeding up co-simulation under the FMI standards, versions 2 and 3 (Modelica Association 2022; Modelica Association 2023). Co-simulation of multiple FMUs face several numerical challenges. Among others, these involve stability, accuracy, efficiency, and step-size control. A wide range of methods for handling these challenges can be found in the literature. For stabilization techniques see for example Benedikt and Drenth (2019) for non-iterative co-simulation and Arnold (2010) for iterative co-simulation. For adaptive co-simulation with step-size control we refer to Schierz, Arnold, and Clauß (2012).

Here we consider the efficiency challenge by proposing two methods to speed up the numerical integration inside each co-simulation FMU. More precisely, our techniques are useful for FMUs using variable-step solvers. Firstly, we apply *input smoothing* to create continuous inputs to the FMUs. Thus, we may avoid restarts of the integrators inside the FMUs. To further help the integrator we also present a *predictor compensation* method.

Input smoothing methods have previously been discussed in the literature, see e.g. Andersson (2016) and Busch (2016) where the smoothing is achieved by the master. These references also include stability and consistency analyses of related smoothing techniques.

However, the main benefit of the input smoothing and predictor compensation presented in this article is the ease-of-use. No special co-simulation master is required. Nor do we require any changes to the models that are exported to FMUs.

Indeed, in our implementation in Dymola 2025x you only need to set the corresponding options before exporting your model to an FMU and the input smoother, with or without predictor compensation, will then be included in the generated code. The resulting FMU can then be used with most co-simulation masters, in particular versions of the classical non-iterative, parallel master.[1] Further, FMUs using input smoothing may be co-simulated with FMUs not using it.

Avoiding integrator re-initialization at communication points may improve performance for any variable step-size solver. But this is especially true for multistep methods as they have to restart at order one, with a tiny internal step size. While ramping up both the order and the step size, several costly model and Jacobian evaluations are required. In this article we consider this class of integrators. More precisely the implementations CVODE and IDA from SUNDIALS (Gardner et al. 2022) and DASSL from Brenan, Campbell, and Petzold (1996).[2] In the context of models exported from Dymola these are arguably the most

[1]The exceptions involve masters that set the input derivatives. Further, since the input to an FMU using input smoothing is delayed, such FMUs cannot be used in co-simulation masters which rely on setting an input and then immediately reading the new output without first performing a `doStep`.

[2]The input smoothing does not depend on which solver is used, but the predictor compensation requires modifications tailored to the solver implementation.

widely used variable-step solvers in co-simulation FMUs. Note that fixed-step solvers, e.g. the commonly used Dymola inline methods, do not benefit from input smoothing.

However, even with smoothed inputs, continuing the integration over communication points comes with additional challenges and problems. We identify and address these issues. To alleviate the main problem caused by avoiding integrators restarts, we arrive at the aforementioned predictor compensation. In Andersson (2016), this technique was discussed in a similar context for integrators with a Nordsieck history representation (e.g. CVODE). In this article we extend the idea to the DAE solvers DASSL and IDA and tailor it to work optimally in combination with input smoothing.

To validate these methods we study their application in both academic models and thermo-fluid examples. One important example is how input smoothing can benefit the creation of datasets for training surrogate models. For this purpose we examine the effect of smoothing inputs on a demonstrator of the *OpenSCALING* research project, delivered by the industry partner *Bosch*.

Finally, we note that FMI 3 comes with the powerful capability *Intermediate Update Mode*, where inputs can be communicated between FMUs also during the communication intervals. This allows for more advanced co-simulation masters and can also be used to reduce the number of integrator restarts. As an example we mention the *Transmission Line Method* (R. Braun and Krus 2013; Robert Braun and Fritzson 2022) where physically motivated time-delays are introduced. While our methods are not motivated by physical considerations, their benefit is that they do not require any special modeling methodology or corresponding co-simulation master.

2 Co-simulation with input smoothing

Co-simulation of multiple FMUs can be performed using a variety of methods. For example iterative methods, higher-order input extrapolation, or non-iterative stabilization techniques. However, we consider this to be out-of-scope for the current presentation. Rather, the purpose of this article is to investigate techniques for speeding up the simulation inside each FMU. Therefore, we here only consider the classical parallel co-simulation algorithm (also known as non-iterative or weak-coupling co-simulation).

2.1 Classical co-simulation

During classical co-simulation the FMUs communicate with each other at *communication points*. This means that the input to each FMU may be computed from outputs of other FMUs. The interval between communication points (the *communication interval*) may vary in size. To advance the time from one communication point to the next, a `doStep` call is made in each FMU. The `doStep` computations may be done in parallel.

The classical approach for handling inputs inside each FMU is to keep any inputs constant throughout the communication interval (zero-order hold). Then, at the next communication point a jump is made to the newly received input values.

We refer to the above algorithm as *classical co-simulation* with each FMU using *classical input handling*.

During a `doStep` call an integrator is employed inside the FMU. The here considered integrators CVODE, IDA, and DASSL are all variable step-size solvers. That is, they have their own step-size sequence. Normally, but not necessarily, there are several integrator steps during each communication interval.

When using classical input handling (zero-order hold) the input signal (as seen by the integrator) is piecewise constant with jumps at the communication points. Therefore, the integrator needs to restart at each such point. This is costly for any variable step-size solver, but especially so for multistep methods.

2.2 Input smoothing

In contrast to classical input handling, the purpose of the proposed input smoothing is to construct input signals which are continuous over communication points. The idea is to do this inside the FMU such that no modifications to the co-simulation master or the exported models are required, cf. Figure 1.

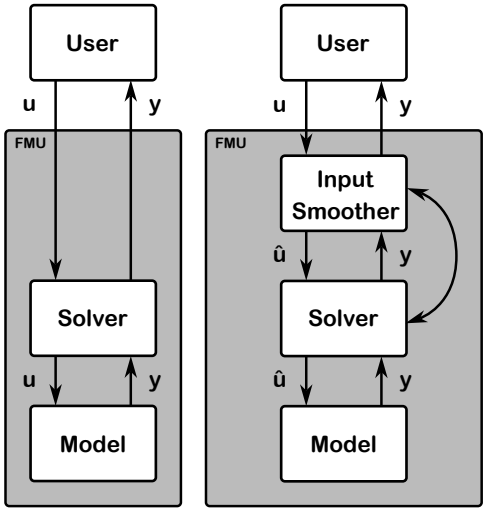


Figure 1. On the left is the abstract structure of a Classical FMU, on the right the modified variant. The input smoothing can be visualized as an additional block between user and model processing the inputs. Input u provided by the user is converted to the interpolated $\hat{u}$ signal. Additionally, an interaction of the input smoothing block and the solver has to be implemented.

Therefore, at each communication point, the input values provided from the master are not immediately set. Instead, we ramp up to these values such that they are reached at the end of the communication interval. In this way, continuous, piece-wise linear input signals are achieved, c.f. Figure 2. This must be done with care to also handle the case of variable communication interval lengths.

DOI 10.3384/ecp218653

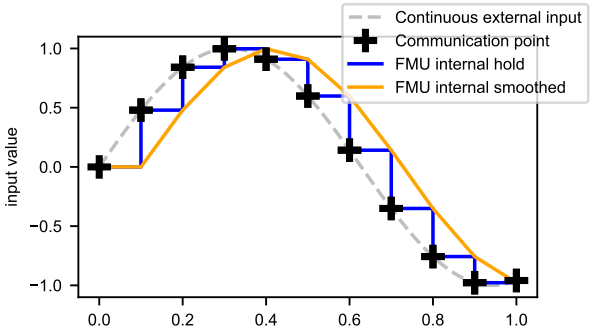


Figure 2. The orange smoothed input signal instead of the blue hold signal is used by the model

As the integrator inside the FMU only sees this smoothed input, then we can allow the integrator to continue integration without restarts at communication points.

An important assumption here is that the integrator does not step past any communication points before receiving new inputs. Otherwise, it would use the previous piecewise linear segment of the input interpolation in the new interval. This is incorrect and also gives a discontinuity in the input signal at the end of the integrator step. An implementation of the input smoothing must make sure that the integrator, at each communication point, has a coinciding internal step. This restriction may lead to an increased number of successful solver steps and Jacobian updates.

The input smoother is enabled in Dymola by setting the flag

`Advanced.FMI.UseInputSmoother`,

and then translating your model to a co-simulation FMU. As algorithm select either CVODE, IDA, or the Dymola solver DASSL. Input smoothing for FMI 2 is supported from Dymola 2025x. FMI 3 support is available from Dymola 2025x Refresh 1.

We have identified two potential problems that may be encountered when using the proposed input smoothing. Firstly, the smoother introduces additional input delays. Secondly, when the integrator uses orders higher than one, mere continuity of the input signals is not enough for optimal performance. These problems are discussed in the upcoming subsections together with proposals for how to handle them.

2.3 Delayed inputs

Already classical input handling introduces input delays as the inputs are set at the communication points and then held throughout the communication interval. The input smoothing makes the situation worse by further delaying the input. This additional delay is effectively half a communication interval as we ramp up to the new inputs, reaching them at the end of the interval. Therefore, input smoothing must be used with care when the co-simulation setup is sensitive to delays.

For such setups, a potential solution may be to slightly reduce the communication interval to compensate for the additional delays. For co-simulation problems where the majority of the CPU time is spent inside the FMUs, then using the input smoother may still be faster, even with the smaller communication interval. This idea is further pursued with the mechanical example model in Section 3.1.

2.4 Discontinuous input derivatives

The input smoothing guarantees that all smoothed inputs seen by the integrator are continuous. But, already the first input derivative is generally discontinuous. When the solver uses any order higher than one, these discontinuities may still cause problems for the integrator.[3] This may result in reduced step size, reduced order, and Jacobian re-evaluations, meaning that we lose some of the benefits from the input smoothing.

The main problem is that the integrator makes bad predictions. The multistep methods considered in this article use old solution values to predict the next step. The predictor can generally be described by a polynomial of the same order as the current order of the BDF method used. This polynomial is evaluated at the time of the next step to get the prediction. For details see Brenan, Campbell, and Petzold (1996) and Radhakrishnan and Hindmarsh (1993).

The prediction is then used as a start guess in the corrector iteration that solves for the actual next step. A bad start guess makes it more difficult for the quasi-Newton method to solve the corrector equations, which may cause extra model evaluations. In the worst case, the Newton method fails, meaning reduced step size, or even Jacobian re-evaluations.

Further, the multistep methods use their predictor to estimate the local error. The estimate is based on the difference between the predictor and the solution of the corrector iterations. When the predictor is bad the error estimate will typically be far too large. The result is that steps are often discarded when they should not be. Even if no steps are discarded, the large error estimates typically result in reduced step sizes and reduced order while the integrator recovers and improves its error estimates.

2.5 Predictor compensation

To alleviate the problem of discontinuous input derivatives, we tailor the predictor compensation of Andersson (2016, Chapter 7) to our input smoother and extend it to the IDA and DASSL solvers. The zeroth- and first-order terms of the predictor polynomial are already reasonable, the first-order term due to the continuity of the input achieved by the input smoothing. Thus, the predictor compensator implemented in Dymola 2025x targets the second-order term. Higher order terms are not considered as they would be computationally too complex to handle with the employed technique, offsetting any benefits.

The predictor compensation is employed as needed at each communication point. It requires an approximation of how the second state derivative is affected by jumps in

[3] In contrast, note that this is not a problem for classical input handling due to the re-initialization done at every communication point.

the input derivative. As the dependency has to be traced through the model, we need to approximate the Jacobian of the state derivatives with respect to the inputs. If the FMU supports directional derivatives (Modelica Association 2022), these can be used for this task.

Dymola FMUs support directional derivatives and require two model evaluations to compute them. This means that the predictor compensation comes with an additional cost that offsets some of the benefits.

To enable the predictor compensation in Dymola, follow the steps in Section 2.2 to enable the input smoothing. In addition, enable the flag

```
Advanced.FMI.UsePredictorCompensation.
```

As algorithm select either CVODE, IDA, or the Dymola solver DASSL. Predictor compensation is supported in the same Dymola versions as input smoothing, except for DASSL where support starts with Dymola 2026x.

3 Applications

We present a simple mechanical model along with two different thermo-fluid examples to highlight the benefit of input smoothing. As mentioned in Section 2.3, the delayed inputs can introduce errors in the simulation. The mechanical example is especially sensitive to delays, and a method to handle it is presented in Section 3.1. The thermo-fluid models studied in Section 3.2 and Section 3.3 are also affected by delays, but for these examples the focus lies on the performance gain of the input smoother.

3.1 Simple mechanical example

To illustrate the effects of input smoothing and predictor compensation we start with a simple mass-spring-damper system. The example is selected not only to showcase the improved simulation statistics, but also to see the effects of input smoothing on a model famously sensitive to co-simulation delays.

3.1.1 Setup

Consider the mass-spring-damper system in Figure 3. We split the model at the connection between the mass component and the spring-damper component. Then we export each submodel into an FMU. From the mass FMU, position and velocity is sent to the spring-damper FMU. The latter sends the force to the former. Please see Table 1 for non-default physical and numerical parameters.

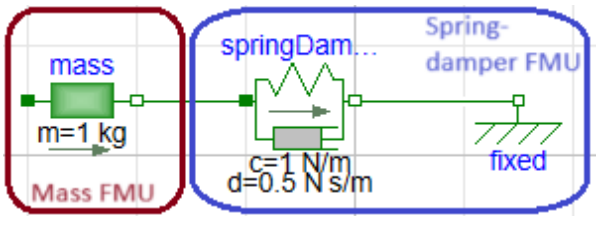


Figure 3. Simple mass-spring-damper model split into two co-simulation FMUs.

Table 1. Physical and numerical parameters of the mass-spring-damper co-simulation problem. Only non-default values are listed.

Parameter	Value
Mass start position	1 m
Mass (m)	1 kg
Spring constant (c)	1 N/m
Damping constant (d)	0.5 Ns/m
Simulation stop time	100 s
Communication interval	0.2 s

The co-simulation is performed using a classical non-iterative master. For the mass FMU we include either CVODE, IDA, or DASSL as solver inside the FMU to integrate its two states: position s and velocity v. For this FMU we either use classical input handling or input smoothing with or without predictor compensation. The spring-damper FMU has no states so it uses no integrator and only uses classical input handling.

3.1.2 Result and analysis

Select simulation statistics for all combinations of solvers and input options are listed in Table 2. In each case, the number of f-evaluations is significantly reduced by input smoothing. The additional evaluations required by the predictor compensation is more than repaid by the further reductions in f-evaluations made by the solvers. Especially for CVODE and IDA. On top of that, the more expensive Jacobian-evaluations are significantly fewer.

When using classical input handling, CVODE constructs one Jacobian at the start of each communication interval (due to the integrator restart) and then keeps it throughout the interval. Enabling input smoothing reveals that these Jacobians actually can be kept for a longer time, resulting in only 40 Jacobian evaluations throughout the co-simulation. Also enabling the predictor compensation helps CVODE's nonlinear solver and reduces the number of required Jacobians to only 14.

For IDA, the reduction in Jacobian-evaluations given by the predictor compensation is not significant. However, when only using input smoothing, then IDAs nonlinear solver fails to converge seven times during the co-simulation. With the help of predictor compensation to improve the start guesses, all of these failures are eliminated revealing the numerical soundness of this technique.

The position of the mass (s) is plotted over time in Figure 4 for the co-simulations using IDA in the mass FMU. A direct simulation of the full system is provided as reference. As expected the classical non-iterative co-simulation attenuates the oscillations. This is made worse by the extra delays introduced by the input smoothing. As previously discussed, the predictor compensation has no direct effects on the simulation results, but rather only helps the solver. Since the result of the input smoother with predictor compensation coincides with that without compensation, we do not include it in the figure.

DOI
10.3384/ecp218653

Table 2. Simulation statistics for the mass FMU from the mass-spring-damper co-simulation problem. Three different solvers are tested with classical input handling or input smoothing (IS) with or without predictor compensation (PC). The statistics are the number of times the model is evaluated by the solver (f-evals), the number of times the solver Jacobian is approximated (Jac-evals), and the number of times the predictor compensation evaluates the model (PC f-evals).

	CVODE			IDA			DASSL		
Option	f-evals	Jac-evals	PC f-evals	f-evals	Jac-evals	PC f-evals	f-evals	Jac-evals	PC f-evals
Classical	5471	499		5958	4960		5042	4543	
IS	2708	40		4715	1030		2653	534	
IS & PC	853	14	996	2692	916	996	1481	259	996

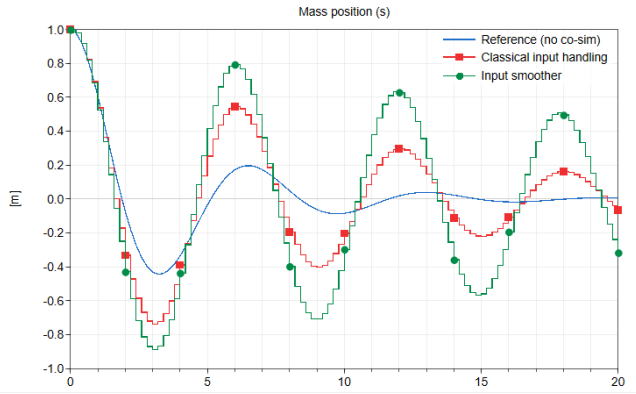


Figure 4. The mass position (s) over the first 20 s of simulation of the mass-spring-damper co-simulation problem with communication interval $H = 0.2$ s, with or without input smoothing. The reference solution is given by simulating the full model directly in Dymola.

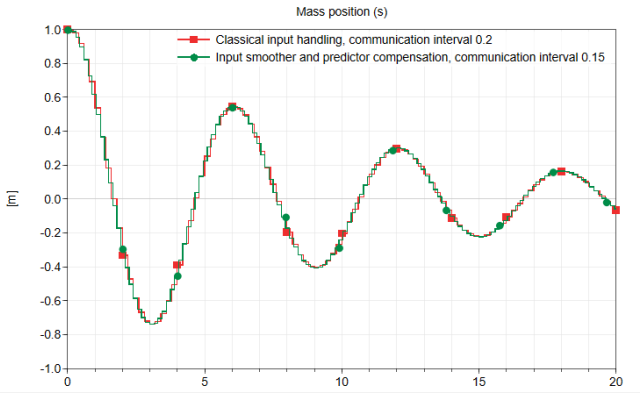


Figure 5. The mass position (s) over the first 20 s of simulation of the mass-spring-damper co-simulation problem. The simulation with input smoothing and the predictor compensation has a reduced communication interval to compensate for the additional co-simulation delay. Under these settings the results are similar.

Table 3. Simulation statistics for the mass FMU from the mass-spring-damper co-simulation problem. The co-simulations are run using IDA. The simulation statistics for communication interval $H = 0.2$ s is as in the previous experiments, cf. Figure 4 and Table 2. To compensate for the additional delay, the interval has been reduced to $H = 0.15$ s for the simulations with input smoothing, cf. Figure 5.

		IDA		
Option	H [s]	f-evals	Jac-evals	PC f-evals
Classical	0.2	5958	4960	
IS	0.2	4715	1030	
IS & PC	0.2	2692	916	996
IS	0.15	2380	471	
IS & PC	0.15	1506	357	1330

As a final experiment we reduce the communication interval to $H = 0.15$ s for the simulations with input smoothing to compensate for the additional delay. With this choice of H, the result is similar to the result when using classical input handling with $H = 0.2$ s, cf. Figure 5. From Table 3 we can see that even with this smaller communication interval the simulations with input smoothing are cheaper in terms of f-evaluations and Jacobian-evaluations inside the mass FMU. Indeed, the statistics are even better compared to Table 2.

A disadvantage is that the master has to handle inputs and outputs more often. Thus, in a larger system, reducing the communication interval could increase the number of function evaluations in other part of the coupled system. Further, in practice, it is not always possible to arbitrarily change the communication interval, e.g. due to real-time requirements.

3.2 HIL simulation of a heat pump

In this section the model of a heat pump as used for Hardware-in-the-Loop (HiL) is examined. On the HiL environment the model is simulated with a (usually) fixed synchronization rate. The input signals of the model are updated in each step. The input signal often has a small noise. In this section we only replicate the boundary conditions by applying a constant input signal with a small noise. The example uses a single FMU, the surrounding environment is represented by the input signals. This example is focussed on the overall performance gain, it does not examine the effect on the results due the modified input trajectory, or the introduced delay.

3.2.1 Setup

The domestic heat pump used in this example is based on the HiL use case of (Schulze, Gräber, and Huhn 2011) and uses the TIL-Suite (TLK-Thermo GmbH 2024). It is a simple vapor compression cycle with efficiency-

based compressor, Bernoulli expansion valve, two 1D-discretized TubeAndTube HX (heat exchangers) and an ideal separator model. The refrigerant is R-407C.

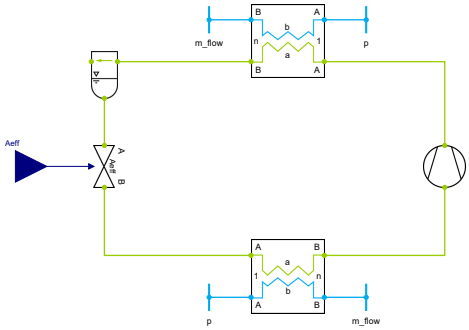


Figure 6. The valve opening area is input of this domestic heat pump model.

A large simulation study with almost 17000 single simulations was conducted to analyze the CPU time performance. The cumulative CPU time exceeded 5 days and 16 hours. The model is exported as FMU 2.0. The solver tolerance is set to 10^{-5}. Five independent "dimensions" have been varied: The solver (CVODE, DASSL and IDA), the smoothing method (Classical, IS, and IS & PC), model variant, the boundary condition (constant, small noise, large perturbation), the `doStep` interval length.

The model variants are:

- Baseline (1 s): Low number of HX volumes with num. ODE-Jacobian and 1 s stop time, 83 continuous-time states, Helmholtz fundamental equation of state.
- Baseline (100 s): Low number of HX volumes with num. ODE-Jacobian and 100 s stop time, 83 continuous-time states, Helmholtz fundamental equation of state. Used to display what happens if the time constants of the model are relatively smaller compared to the `doStep` interval lengths.
- High-Resolution (1 s): High number of HX volumes with num. ODE-Jacobian and 1 s stop time, 403 continuous-time states, Helmholtz fundamental equation of state. Used to display the influence of the model complexity.
- Analytical Jacobian (1 s): Low number of HX volumes with analytical ODE-Jacobian and 1 s stop time, 83 continuous-time states, spline-based fluid properties. Used to display the influence of the computation of the ODE-Jacobian.

There are three types of boundary conditions for the effective valve opening area:

- Constant input: 1mm^2
- Small noise: 1mm^2 plus a small random number varying at each time step with an amplitude of 0.1%, simulating signal noise.
- Large stochastic perturbation: 1mm^2 plus a large random number varying at each time step with an amplitude of 50%, replicating abrupt controller output changes.

The time constants of the Baseline model with 83 continuous-time states range from 0.1 s to 65 s with a mean of 2.1 s and a median of 0.58 s. The time constants of the High-Resolution model with 403 continuous-time states range from 0.023 s to 52 s with a mean of 0.6 s and a median of 0.1 s.

3.2.2 Result and analysis

Figure 7 shows the performance gain of CPU time relative to the reference case plotted against the number of `doStep` calls. The Classical FMU is the fastest scenario if the inputs are constant (best-case scenario). If the input of the Classical FMU is updated, then the solver is reset and the CPU time increases significantly (worst-case scenario). The IS variant has a limited solver step size, and an increased number of Jacobian evaluations compared to the Classical variant with constant input signal, as mentioned in Section 2.2.

The Baseline (100 s) demonstrates smaller performance gains if the input is subject to large perturbations. The model responds comparatively quickly because the `doStep` interval length is larger compared to the model time constants. The model has to be brought closer to the steady state, and hence the computational effort is higher. The effect of input smoothing diminishes. Using constant input values (dotted line, best-case scenario) is significantly faster, but also the results are significantly different.

The High-Resolution model demonstrates that as the number of continuous-time states of the model increases, the computational cost of the f- and Jacobian-evaluation also rises. Consequently, the speed-up factor of the IS and IS & PC method becomes more significant. The model variant with analytic Jacobian is generally faster than the Baseline with num. Jacobian. In all cases, the speed-up factor with analytic Jacobian is at a lower level, but the overall picture is unchanged.

The minimum and maximum speed-up summarized in Table 4 of the variants shown in Figure 7 illustrate that input smoothing is beneficial, if inputs have to be updated and the synchronization step size is low.

Augmenting the input smoother with predictor compensation is occasionally very beneficial. In Figure 7 we see the benefits when the input is altered by large perturbation and IDA or DASSL is used. When only the input smoother is used, these represent the cases where the solver runs into severe problems due to the large changes in the derivative of the smoothed input signal. Investing a few extra model evaluations at communication points to help the solver predictor is repaid many times over. In the best cases, we see about 10 times faster simulation compared to only using input smoothing (about 100 times faster compared to the reference). The most striking is that the number of Jacobian evaluations is reduced from about 1000 (IS) to

DOI
10.3384/ecp218653

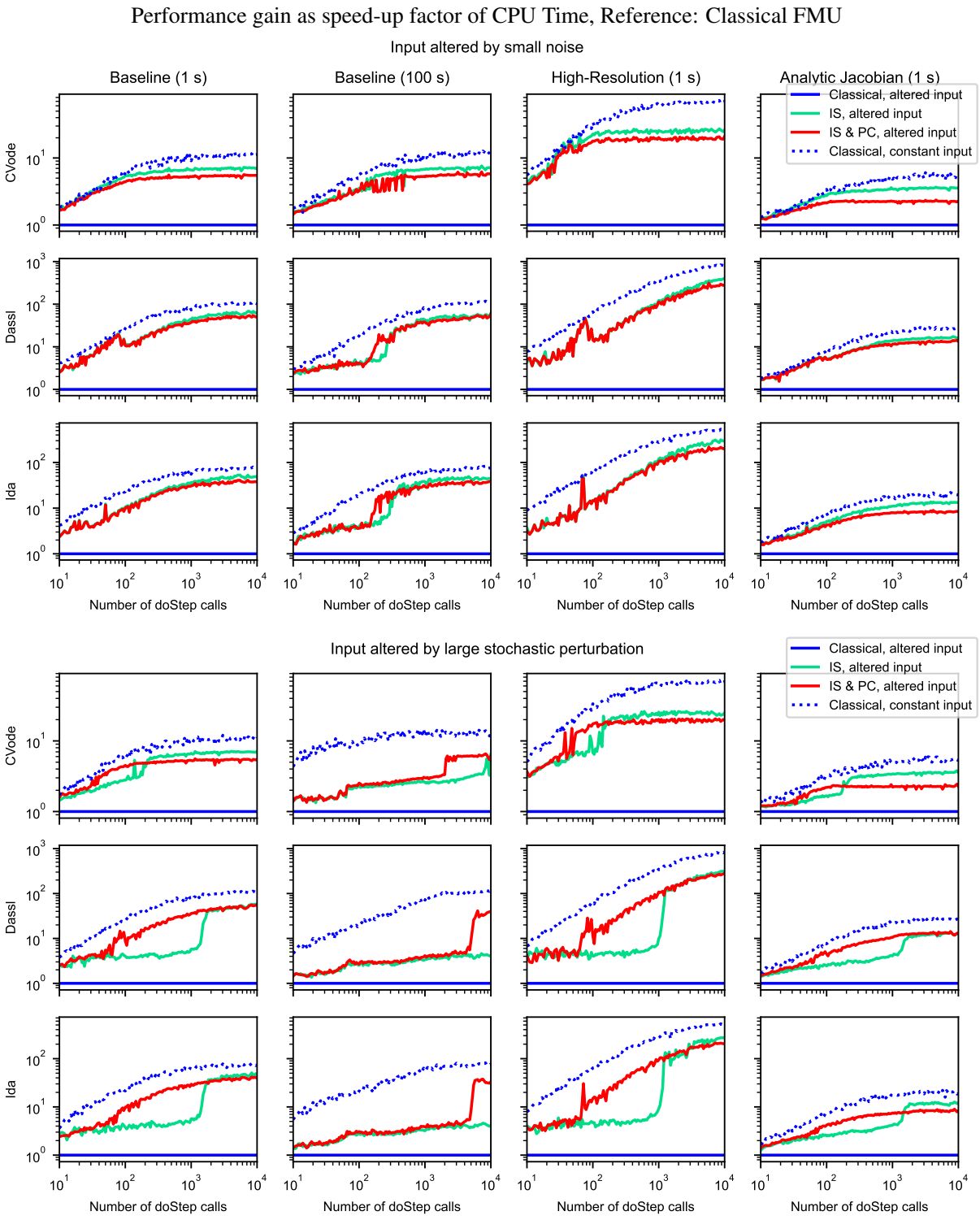


Figure 7. The figure illustrates the CPU time performance gain across various simulations, represented as a speed-up factor relative to the solid blue reference scenario. The x-axis indicates the number of `doStep` calls required to reach the fixed stop time. The upper half of the figure represents simulations with small noise in the boundary condition, while the lower half shows results under large stochastic perturbations. The reference scenario, defined as the Classical FMU with altered input (regardless of perturbation amplitude), represents the worst-case in terms of performance. Conversely, the Classical FMU with constant input exhibits the fastest execution and serves as the best-case scenario. As the number of `doStep` calls with the same stop time increases, the performance gain improves in all plots. The Baseline (100 s) shows smaller performance gains under large perturbation, as the model responds comparatively quickly. With increasing model complexity (High-Resolution), the performance gain becomes more pronounced. Utilizing an analytic Jacobian reduces the impact of input smoothing while preserving the overall trend. Adding predictor compensation is sometimes very beneficial with large perturbation boundary conditions using DASSL or IDA, reaching about a factor of 10 times faster simulations compared to when only using input smoothing.

Table 4. The table provides a summary of Figure 7, presenting the minimum and maximum speed-up factors across selected number of `doStep` calls. The range reflects the differences among model variants and boundary conditions. The rows compare the IS and IS & PC performance with updated inputs to the best case scenario. Across all scenarios, IS consistently improved performance. However, the speed-up factor is highly influenced by time constants, model complexity and the input trajectory.

		Speed-up factor			
Solver	**Variant**	Number of doStep calls			
		10	100	1000	10000
CVODE	Classical with constant input	1.3 - 5.5	3.1 - 34.0	4.8 - 61.4	5.2 - 70.1
	IS, altered input	1.2 - 4.5	1.7 - 22.8	2.6 - 25.7	3.6 - 26.2
	IS & PC, altered input	1.2 - 4.1	2.1 - 18.0	2.2 - 20.3	2.3 - 21.0
DASSL	Classical with constant input	1.8 - 7.8	7.6 - 66.5	21.3 - 338.9	25.8 - 847.2
	IS, altered input	1.4 - 4.3	2.4 - 14.6	3.4 - 122.0	4.1 - 403.0
	IS & PC, altered input	1.5 - 4.9	2.7 - 14.6	4.1 - 113.2	13.1 - 282.1
IDA	Classical with constant input	1.8 - 8.8	7.1 - 67.8	17.2 - 272.5	19.0 - 532.2
	IS, altered input	1.3 - 4.1	2.5 - 14.9	3.2 - 119.6	4.1 - 299.2
	IS & PC, altered input	1.4 - 3.8	3.1 - 14.9	3.8 - 105.9	8.3 - 206.2

about 50 (IS & PC), showcasing the numerical soundness of the predictor compensation.

With that said, we can also see that predictor compensation is not always needed. When the input is altered by small noise, then the changes in input are small enough for the integrator to function well, even without the help of predictor compensation. This is evident by looking at the internal step-size sequence of the integrator inside the FMU. Very often the integrator takes the largest possible step, i.e. the length of the communication interval. In these cases, the predictor compensation only contributes its additional f-evaluations and therefore worsens performance. This is also the case in the experiments with CVODE when the input is altered by large perturbation and the number of `doStep` calls is large.

It is important to note that the results of the simulations performed are not identical. The user must evaluate whether the results are still within tolerance, even with the altered input trajectory effectively causing additional time delay.

3.3 Heat pump simulation data for training surrogate models

To show the possible improvements for industry-relevant applications, we investigate the application for a detailed model of an air-source heat pump. The used model is a demonstrator of the *OpenSCALING* research project, delivered by the industry partner *Bosch*. It consists of a refrigeration cycle with controller and is modeled using the TIL-library of *TLK-Thermo*. The outdoor heat exchanger is especially detailed; apart from the liquid-vapor behavior of the R-290 refrigerant, it models the moist air flow including condensation and frost formation on the heat exchanger surface. The diagram view of the R-290 refrigeration cycle (or just model) can be seen in Figure 8.

In practice, system models including refrigerant cycles can be computationally expensive, due to the detailed representation of the physical phenomena (evaporation, condensation, ice formation, ...). Required simulation times

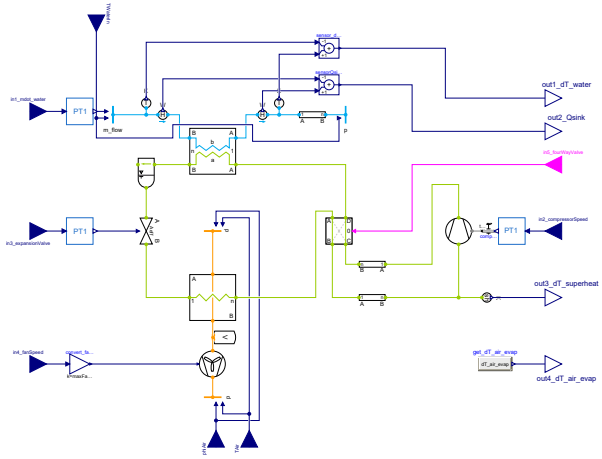


Figure 8. Diagram of the R-290 refrigerant with 7 inputs.

and computational resources set limits to repeated evaluation of the models. This restricts how the model can be used, for example in virtual testing, controller design or setpoint optimization. Here, data-driven surrogate modeling comes into play: By sampling input signals to the physical simulation model, a dataset is created, with which surrogate models are trained. This means that one has to accept a high computational upfront cost for generating the surrogate model; but once it is trained, its evaluation becomes really cheap and can be easily used for millions of evaluations. For example, *University of Augsburg* examines in the research project *OpenSCALING* Balanced Neural ODEs for generating surrogates. This method has been demonstrated to decrease computation times by a factor up to 100 (Aka et al. 2025).

3.3.1 Operating points

Evaluation times of the heat pump model can considerably differ depending on the operation point; it is especially high for reverse-cycle operation to defrost the outdoor heat exchanger, or for oscillating controller behavior, that frequently arises during virtual controller devel-

DOI
10.3384/ecp218653

opment. On the other hand, if steady state operating conditions are reached, adaptive-step size solvers quickly solve the model. To evaluate the impact of the input smoother for these different cases, we include it with (1) tuned controllers that avoid oscillating behavior and (2) controller with high gains to enforce oscillations. Additionally, the model is operated at air-conditions that avoid frost formation.

3.3.2 Setup

The heat pump system consisting of the refrigeration cycle and controllers is exported as a co-simulation FMU 2.0 with DASSL as solver. The inputs are defined in a co-simulation master. Moreover, the heat-pump model with setup for controllers is also implemented in pure Modelica and simulated for comparison.

3.3.3 Result and analysis

The results for the simulation statistics related to the solver are found in Table 5, while computational time measured is presented in Table 6.

Table 5. Simulation statistics of the solver for the heat pump co-simulation FMU exported with DASSL, from the controller model for the two different operation points. Simulated to $T = 500$ s with communication interval $H = 1$. In addition the model is simulated using pure Modelica, with applicable statistics for f-evals and Jac-evals. The model with tuned controllers that avoid oscillating behavior in the output is labeled *Low Oscillations* while the model with controllers with high gains to enforce oscillations in the output is labeled *High Oscillations*.

Low Oscillations				
Option	f-evals	Jac-evals	Events	PC f-evals
Classical	8251	4546	506	
IS	1950	209	6	
IS & PC	1910	222	6	998
Modelica	1038	140		

High Oscillations				
Option	f-evals	Jac-evals	Events	PC f-evals
Classical	71939	9073	514	
IS	70734	7011	16	
IS & PC	70442	6936	16	926
Modelica	35343	6113		

When the controller is tuned to avoid oscillating behavior (Low Oscillations) the input smoother performs well, reaching over 40 times faster co-simulation. This is clear from looking at the reduced number of events in Table 5. The Classical FMUs generate events every time the integrator re-initializes. Thanks to the continuity of the input these re-initializations are not needed, drastically speeding up the simulation. On the other hand, adding predictor compensation slightly worsens performance for the same reason as described in Section 3.2.2 when the input is altered by small noise, c.f. Figure 7.

Although still having a positive impact, input smoothing is less effective for the model containing high oscillations. The reason is that the simulation inside the heat pump FMU is challenging for the integrator given the used communication interval. The work is dominated by a lot of internal steps in each interval, overshadowing the performance gained by avoiding restarts at communication points. Adding predictor compensation improves performance slightly.

Table 6. CPU-time statistics of the solver for the heat pump co-simulation FMU exported with DASSL, from the controller model for the two different operation points. CPU-time gain is calculated with classical FMU as baseline. Simulated to $T = 500$ s with communication interval $H = 1$. In addition the model is simulated using pure Modelica. The model with tuned controllers that avoid oscillating behavior in the output is labeled *Low Oscillations* while the model with controllers with high gains to enforce oscillations in the output is labeled *High Oscillations*.

Low Oscillations		
Option	CPU-time [s]	CPU-time gain
Classical	129	1
IS	3.11	41.5
IS & PC	3.27	39.5
Modelica	2.35	54.9

High Oscillations		
Option	CPU-time [s]	CPU-time gain
Classical	321	1
IS	273	1.18
IS & PC	255	1.26
Modelica	183	1.75

In all cases it is most efficient to simulate the model using pure Modelica, but employing input smoothing techniques decreases computational effort to be more in level with the Modelica implementation.

4 Conclusion

In this article we have introduced techniques that may speed up co-simulation by orders of magnitude, namely input smoothing and predictor compensation. The main benefit is their ease-of-use. No special co-simulation master is required, nor any manipulations of the models that are being exported to FMUs. Rather, the options are easily enabled in Dymola 2025x and the logic is handled entirely by the generated code.

By smoothing the input to the FMU, integrator re-initializations at communication points can be avoided. For multistep solvers like CVODE, IDA, and DASSL, this may heavily reduce the number of model and Jacobian evaluations. The benefits of smoothing the input were observed for the examples studied in this article, with some configurations seeing over 400 times faster co-simulation.

However, the smoothed input may still have discontinuous derivatives. To handle this problem we proposed the use of predictor compensation to help the integrator inside

the FMU. When enabled, additional model evaluations are performed to improve the guess for the next internal state. In the best cases we observed performance gains of a factor 10 compared to only using input smoothing.

Based on the experiments with the thermo-fluid applications we conclude that input smoothing gives the greatest benefits when the co-simulation FMU is computationally heavy but the communication interval is small compared to the dynamics of the model inside the FMU. This is a commonn case in applications as the communication interval often is set by external factors, such as stability and accuracy of the overall co-simulation problem or real-time requirements, c.f. Sections 3.2.

Predictor compensation is most beneficial when, additionally, the integration with input smoothing is not too trivial. Thereby, predictor compensation performs best within a sweet spot where the integration inside the FMU is not too simple, nor too difficult. In our experiments IDA and DASSL are more often helped by predictor compensation.

Although, the results presented in this article are very promising, further work is needed to study the effect of input smoothing and predictor compensation for co-simulation setups containing several coupled FMUs.

5 Acknowledgments

The authors would like to thank Vaillant and Airbus for their invaluable contributions in developing and testing the input smoothing technology.

Research for the input smoother was funded by the German Federal Ministry for Education and Research (BMBF) under the research project “MODAK” (Grant 01IS17006C).

The inclusion of the air-source heat pump example for training surrogate models has been supported by VINNOVA (grant number 2023-00969) and by the ITEA 4 project OpenSCALING (Open standards for SCALable virtual engineerING and operation) N°22013 under grant number 01IS23062H.

SPONSORED BY THE



References

Aka, Julius et al. (2025). “Balanced Neural ODEs: nonlinear model order reduction and Koopman operator approximations”. In: *The Thirteenth International Conference on Learning Representations*.

Andersson, Christian (2016). “Methods and Tools for Co-Simulation of Dynamic Systems with the Functional Mock-up Interface”. Doctoral Thesis (monograph). Mathematics (Faculty of Engineering, Lund University). ISBN: 978-91-7623-698-7.

Arnold, Martin (2010-05). “Stability of Sequential Modular Time Integration Methods for Coupled Multibody System Models”. In: *Journal of Computational and Nonlinear Dynamics* 5.3, p. 031003. ISSN: 1555-1415. DOI: 10.1115/1.4001389.

Benedikt, Martin and Edo Drenth (2019). “Relaxing Stiff System Integration by Smoothing Techniques for Non-iterative Co-simulation”. In: *IUTAM Symposium on Solver-Coupling and Co-Simulation*. Ed. by Bernhard Schweizer. Cham: Springer International Publishing, pp. 1–25. ISBN: 978-3-030-14883-6.

Braun, R. and P. Krus (2013). “Tool-Independent Distributed Simulations Using Transmission Line Elements And The Functional Mock-up Interface”. In: *53rd SIMS conference on Simulation and Modelling*. Linköping University Electronic Press.

Braun, Robert and Dag Fritzson (2022). “Numerically robust co-simulation using transmission line modeling and the Functional Mock-up Interface”. In: *SIMULATION* 98.11, pp. 1057–1070. DOI: 10.1177/00375497221097128.

Brenan, Kathryn E., Stephen L. Campbell, and Linda R. Petzold (1996). *Numerical Solution of Initial-Value Problems in Differential–Algebraic Equations*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics.

Busch, Martin (2016). “Continuous approximation techniques for co-simulation methods: Analysis of numerical stability and local error”. In: *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik* 96.9, pp. 1061–1081.

Gardner, David J et al. (2022). “Enabling new flexibility in the SUNDIALS suite of nonlinear and differential/algebraic equation solvers”. In: *ACM Transactions on Mathematical Software (TOMS)*. DOI: 10.1145/3539801.

Modelica Association (2022-11). *Functional Mock-up Interface for Model Exchange and Co-Simulation Version 2.0.4*. Tech. rep. Modelica Association. URL: https://fmi-standard.org.

Modelica Association (2023-07). *Functional Mock-up Interface for Model Exchange and Co-Simulation Version 3.0.1*. Tech. rep. Modelica Association. URL: https://fmi-standard.org.

Radhakrishnan, K and A C Hindmarsh (1993-12). “Description and use of LSODE, the Livemore Solver for Ordinary Differential Equations”. In: DOI: 10.2172/15013302.

Schierz, Tom, Martin Arnold, and Christoph Clauß (2012). “Co-simulation with communication step size control in an FMI compatible master algorithm”. In: *Proceedings of the 9th International Modelica Conference*. Linköping University Electronic Press, pp. 205–214. DOI: 10.3384/ecp12076205.

Schulze, Christian, Manuel Gräber, and Michaela Huhn (2011-06). “Real-Time Simulation of Vapour Compression Cycles”. In: DOI: 10.3384/ecp1106348.

TLK-Thermo GmbH (2024). *TIL Suite*. Version 2024.1. URL: https://www.tlk-thermo.com/software/til-suite.

Comparing the Predictive Event Handling Algorithm LookAhead to Rollback and Early Return

Felix Tischer[1] Simon Genser[1] Daniel Watzenig[1, 2] Martin Benedikt[1, 3]

[1]Virtual Vehicle Research GmbH, Inffeldgasse 21a, 8010 Graz, Austria,
{Felix.Tischer,Simon.Genser,Daniel.Watzenig,Martin.Benedikt}@v2c2.at
[2]Institute of Visual Computing, Graz University of Technology, Inffeldgasse 16/2, 8010 Graz, Austria,
daniel.watzenig@tugraz.at
[3]SETLabs Research GmbH, Elsenheimerstraße 55, 80687 Munich, Germany, Martin.Benedikt@setlabs.de

Abstract

LookAhead is a lightweight algorithm that improves event handling in co-simulation by predicting events and adjusting the communication step size beforehand. It operates without requiring subsystem event handling capabilities. This paper compares LookAhead with other event handling methods, namely Rollback and Early Return, from the perspective of performance and applicability. Results from the presented example show that LookAhead performs on par with iterative co-simulation methods and is particularly well-suited for handling shared state events.
Keywords: co-simulation, non-iterative co-simulation, hybrid simulation, event handling

1 Introduction

In co-simulation, the continuousness of signals is critical for performance and accuracy. The coupling error at coupling time steps can introduce discontinuities in a signal that might lead to slower computation, as a subsystem's solver algorithm might have to reset or use smaller steps. Methods like anti-aliasing can be used to smoothen input signals and improve co-simulation results. Such signals usually represent physical quantities that are continuous in nature. In other cases, a signal can actually be discrete, which can pose a particular challenge for co-simulation. This is a common challenge in the simulation of cyber-physical systems, where physical systems interact with or are being controlled by discrete control signals. A discrete change in signals, state or behaviour in a simulation model in general is called an event.
Events can be the result of different types of causes (Cellier and Kofman 2006). An event that is triggered at a certain moment in time, irrespective of the system's state (within limitations), is called a time event. An event that is directly caused by the system reaching a certain state on the other hand is called a state event.
A challenge of discrete signals unique to co-simulation are shared events, where events are being triggered in multiple subsystems by the same cause; if such an event occurs in only some subsystems or at different times, it can lead to incorrect and unphysical behaviour like violating boundary conditions or conservation of energy.
Due to the evolving sophistication of co-simulation techniques and the increasing importance of simulation and co-simulation in industrial development, proper event handling has become more and more important and has been a major focus in the development of the fmi3 standard.

2 Algorithm Description

2.1 LookAhead

For systems that lack sophisticated event handling support, LookAhead can improve the results of a co-simulation by predicting events and adapting the communication step size (Tischer et al. 2023). It does so by defining event indicators, which are sets of functions of a subsystem's inputs, outputs, and parameters. Each function z in an event indicator (called an event condition) is defined in a way such that $z < 0$ means that this condition of the event is fulfilled. An event occurs when all conditions are fulfilled, which is equivalent to when the last condition function crosses zero. Via extrapolation of these event condition functions, an event can be predicted, and the communication time step can be reduced and adapted in advance to hit the event precisely. For a more thorough description, see (Tischer et al. 2023).
As LookAhead only relies on a subsystem's interface, it can be used with any subsystem regardless of its own event handling capability and whether it is a black-box or not. On the other hand, knowledge about its events is needed to a degree that their conditions can be modelled; additionally, the event conditions also need to be able to be modelled in this manner instead of relying on internal variables.

2.2 Rollback

Rollback capability means that a system's state can be reset to a prior point in simulation, usually the previous time step. FMUs with the *canGetAndSetFMUstate* attribute are able to store their state at any point in a simulation and thus support Rollback.
To improve event handling in our use case, the subsystems store their state at every communication time step.

This save state is being stored externally (meaning, outside of the FMU by the importer) for one communication time step, when it is being overwritten by the new one if the step finished successfully. If the step fails, both FMUs in the example system (see following section) are being reset to the beginning of the step, and a new step size is chosen. In our example, if an event occurs in the middle of a communication time step, the simulation is being rolled back to the last time step and then a new communication step size is chosen so that the event occurs at the communication point.

2.3 Early Return

Early Return is an fmi3 functionality (Modelica Association 2024) where an FMU can do a simulation step and, for example when encountering an event, stop and return early from the step. This is different from Rollback in that the step was still finished successfully and does not have to be repeated. The importer has to declare to the FMU that it supports Early Return and can handle subsystems not finishing a whole step. In our example, the FMUs return early when encountering an event to allow for additional exchange of signals.
Early Return can be requested from the outside by the importer as well via callback functions, but this has not been utilised in this setup.

3 Example Setup

3.1 Example System

The example system that the algorithms are being tested on consists of two connected spring-damper-mass systems. The upper mass is connected to a fixed ceiling at height $x = 0$ while the lower mass is connected to the upper one. Both connections have the same stiffness and damping coefficients, $k = 100, d = 1$. This system was introduced in (Dejaco and Benedikt 2017) and was slightly adapted from there.
The system's state is described by four variables, the positions and velocities of both blocks, which can be described by the following system of first order differential equations:

$$\begin{aligned} \dot{x}_1 &= v_1 \\ \dot{x}_2 &= v_2 \\ \dot{v}_1 &= -\frac{2k}{m_1}x_1 - \frac{2d}{m_1}v_1 + \frac{k}{m_1}x_2 + \frac{d}{m_1}v_2 - g \\ \dot{v}_2 &= -\frac{k}{m_2}x_2 - \frac{d}{m_2}v_2 + \frac{k}{m_2}x_1 + \frac{d}{m_2}v_1 - g \end{aligned} \tag{1}$$

with $i \in 1,2$ denoting the upper and lower block and x, v their position and velocity, respectively. The half height of each block is $\Delta x = 0.55$. When splitting the system up into two subsystems, each block takes the other one's position and velocity as inputs and provides its own as outputs.
This simple, continuous-state system is being complicated by two collision events. The upper block can collide with the ceiling, and both blocks can collide with each other. The former is a private event that only affects the upper block, while the latter is a shared event between both blocks.

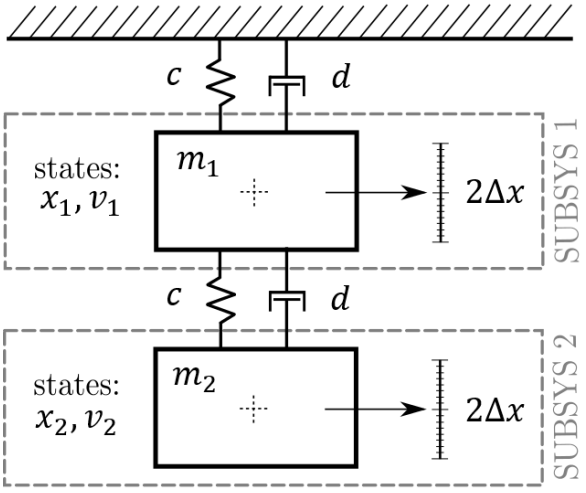


Figure 1. Schematic of the spring-damper-mass system. Figure was taken from (Dejaco and Benedikt 2017) and adapted.

The conditions for the wall block's private event, the collision with the ceiling, are

$$\begin{aligned} -v_1 &< 0 \\ -x_1 &< \Delta x. \end{aligned} \tag{2}$$

The ceiling collision is handled internally and simply by subsystem 1 by flipping its velocity:

$$v_1' = -v_1. \tag{3}$$

Primed variables denote variables after the event. The conditions for the shared event, the collision between the blocks, are

$$\begin{aligned} v_1 - v_2 &< 0 \\ x_1 - x_2 &< 2\Delta x. \end{aligned} \tag{4}$$

The binary collision is handled internally (and seperataly) in both blocks. Each block does this by using the formulas for elastic collisions to calculate the post-collision velocities. For calculating the time steps between collision and the next communication step with the other block, not only the block's own velocity is being calculated but also the other one's to prevent the possibility of false succeeding collisions.
The calculation is being done by these formulas:

$$\begin{aligned} v_1' &= \frac{m_1 - m_2}{m_1 + m_2}v_1 + \frac{2m_2}{m_1 + m_2}v_2 \\ v_2' &= \frac{m_2 - m_1}{m_1 + m_2}v_2 + \frac{2m_1}{m_1 + m_2}v_1. \end{aligned} \tag{5}$$

Even though the equations are the same in both blocks, the results can differ, as each block can only access the other's velocity as an input (that is dependent on

DOI 10.3384/ecp218663

coupling step size and method, and is therefore affected by coupling error), and not the exact state value.

Similarly to how the event prediction works in LookAhead, the FMUs use event indicators to trigger their events internally. For an event E_i, we call its event indicator $M_i(z)$, analogously to the naming conventions of the fmi standard. Then, the event E_i occurs if and only if $M_i(z) < 0$. To use the event conditions Equation 2 and Equation 4, we just define the $M_i(z)$ for each event as the minimum of its event conditions:

$$M_i(z) = min_j z_{ij}, \tag{6}$$

where z_{ij} is the j-th event condition for event E_i.

3.2 Co-simulation Setup

The subsystems have been compiled as FMUs, using the Modelica Reference FMUs as a basis. Three versions have been compiled of each subsystem: one without any additional event handling methods, one implementing Rollback, and one implementing Early Return. The simulated time is 20 seconds, the longest time in which the described system shows interesting behaviour. After that, the system reaches a somewhat stable configuration and no more events occur.
The co-simulations have been carried out in Python, using the fmpy package to import and simulate the FMUs. The Early Return FMUs are equipped with Intermediate Update (Modelica Association 2024) support, which is used to log the FMUs' states at every microstep.
In total, simulations with five different setups have been carried out.

- The first, the monolithic simulation, serves as ground truth for our analysis; this was not a co-simulation, but was in fact a simulation of the total system of four equations that the two subsystems consist of. The results have been saved with a time step of $\Delta t = 0.001s$, which is the time step of the fixed step solver integrated in the FMUs.

- The second setup is the standard co-simulation, where no event handling has taken place. This means that collisions are only being handled internally by the subsystems, and the master is only exchanging information between them without any additional handling such as inter- or extrapolation, event triggering, or time step adaptations. It has been carried out with a fixed communication step size of $\Delta t = 0.08s$, which was also the default for the following setups.

- The third setup is similar to the second, except the addition of the LookAhead algorithm turned on, evaluating the event indicators once per simulation loop and adapting the communication step size when an event is being predicted. The minimum step size for LookAhead was equal to the solver step size of $\Delta t = 0.001s$.

- The fourth setup used Rollback. The Rollback FMUs were equipped not only with the capability to reset their state, but also with two additional output signals to indicate if (called EventSensor) and when (called EventTime) an event occurred. If an event occurred at a time that was too far away from a communication point, the co-simulation algorithm would reset the last simulation step and repeat it with the event time as target time.

- The final setup utilises Early Return. The FMUs have been adapted in such a way that, when they are being initialised with $earlyReturnAllowed = True$, they return early immediately before an event occurs. For this, the event indicators $M(z_i)$ have been slightly changed by subtracting a small threshold $m = 0.05$ from their final value, but also changing the condition for triggering the event to $M(z_i) < m$. This makes it so that the event trigger remains unchanged, but the FMU returns early right before the event to allow for exchange of signals before it.

4 Analysis

4.1 Criteria

To compare and analyse the different co-simulation methods, the position error is used as a key metric. This error is calculated as the absolute difference between the position and a reference or ground truth over time, as given in Equation 7, where x_i is the position of block i in the co-simulation and $\hat{x}_i$ is its counterpart in the reference solution. By examining the magnitude and trend of this error, we can assess the accuracy and performance of each co-simulation method.
By taking the root mean square of it, we can get an average error as well. The RMS provides a single scalar value that reflects the overall magnitude of the error, smoothing out noise and highlighting consistent deviations. By evaluating the RMS error up to different points in time, we can observe how the accuracy of the simulation evolves and compare the performance between the different methods. This time-resolved analysis helps to identify when and where the simulation performs well and when it diverges.

As the performance of a hybrid system can depend heavily on how well the events are being handled, the temporal accuracy of events (especially the shared event) is also an important metric that has a direct effect on the position error. Temporal accuracy refers to how closely the timing of key events in the simulation aligns with those in the reference solution. To evaluate temporal accuracy, timestamps of the collision events are compared across simulations, and any deviations are measured as time offsets.

This is defined in Equation 8, analogously to the position error, summing up the time differences of all n events that occur during simulation. To get a meaningful result for the time error, all the events in the reference solution have to have a counterpart in the co-simulation. If the co-simulation has events that do not occur in the reference, or any events of the reference do not occur in the co-simulation (or, if it is a shared event, do so only in some of the partaking subsystems) the time error can not be calculated.

$$e_{pos} = |x_1 - \hat{x}_1| + |x_2 - \hat{x}_2| \tag{7}$$

$$e_{time} = \sum_{i}^{n} |t_i - \hat{t}_i| \tag{8}$$

4.2 Results

In Figure 2 we can see the results of the five simulations described in subsection 3.2. One can see that the shared events are the main contributor to the position error; after each collision the deviation from the reference increases.
Table 1 shows the RMS position error for each simulation co-simulation method evaluated at three different time points: 2 seconds, 10 seconds, and at stop time of 20 seconds. The standard co-simulation already has a large error at $t = 2s$ of 0.609 m after just 2 events, and increases by a factor of ten at $t = 10s$. At this point, the simulation is already far off from the reference solution.
In contrast, LookAhead and Rollback maintain consistently low errors across the entire simulation, indicating strong and stable accuracy. Finally, Early Return starts off with the lowest error at $t = 2s$ ($0.071m$), but experiences a significant jump to $2.036m$ at $t = 10s$ and a slightly lower $1.936m$ at $t = 20s$. This accumulating error leads to a diverging behaviour and indicates that the events have not been handled well enough. Overall, LookAhead and Rollback demonstrate the most stable and accurate performance over time, while Early Return is accurate at first, but diverges later in the simulation.
One can see that the Rollback and LookAhead co-simulations performed better than the standard co-simulation and the one using Early Return; we split the result analysis into these two groups.

When looking at the results for the standard and Early Return variants (Figure 3), we can see that not only the trajectories diverge, but also the event times do not match and, at a simulation time of about 12 seconds, an additional collision takes place which changes the following trajectory so much that the position error is no longer a meaningful metric. The issue with Early Return is that, unlike Rollback, it is not possible to reset a simulation, so when one subsystem returns early but the other has completed its step (or stopped at a later time during the step), it is no longer possible to reset the simulation to the point of the event. The early returned subsystem can be fed with updated inputs, but the other one not. Two possible ways to solve this problem is to either schedule the subsystems sequentially, which works when one of them tends to stop earlier, or to allow the FMUs to be stopped externally, which introduces non-deterministic behaviour (Broman 2017). If there is an untreated discrepancy in event times between the involved subsystems, an additional error is introduced, possibly leading to a bigger time discrepancy at the next shared event and explaining the increasing position error.

On the other hand, the Rollback and LookAhead methods (Figure 4) performed much more accurate, with no additional events occurring. From a first impression, one might assume that Rollback and Early Return perform the same; in both cases, the simulation introduces an additional communication step at the event time, and calculating exactly the same outside of any events. The difference is that, as Rollback is an iterative co-simulation technique, the system has more accurate information during subsequent repeated calculation steps, resulting in reduced error — independent of how the event is handled. Unlike Early Return, both subsystems in Rollback calculate to the same event time as their communication step is governed by the co-simulation algorithm, preventing any time discrepancy errors.
LookAhead handles shared events as well as Rollback does, even though it is non-iterative. It accomplishes this by reducing the communication step size just before the event, introducing additional communication steps. Since collisions significantly contribute to position error, the co-simulation improves accuracy by increasing signal exchange when it matters most, while maintaining larger step sizes elsewhere to optimize performance.

Table 1. Root mean square position error at different times.

Variant	RMS@2s	RMS@10s	RMS@end
Standard	0.609	6.060	4.578
LookAhead	0.146	0.100	0.132
Rollback	0.139	0.178	0.159
Early Return	0.071	2.036	1.936

5 Summary, Conclusion and Future Work

We have introduced the LookAhead algorithm, a lightweight event handling algorithm that works by predicting upcoming state events and reducing the communication step size, and compared it to two other event handling methods, the iterative Rollback mechanism and fmi3's Early Return functionality. Based on an example system that has proven itself to be a hard task to simulate correctly due to the shared events having such a large influence on the error, we compared each method's

DOI 10.3384/ecp218663

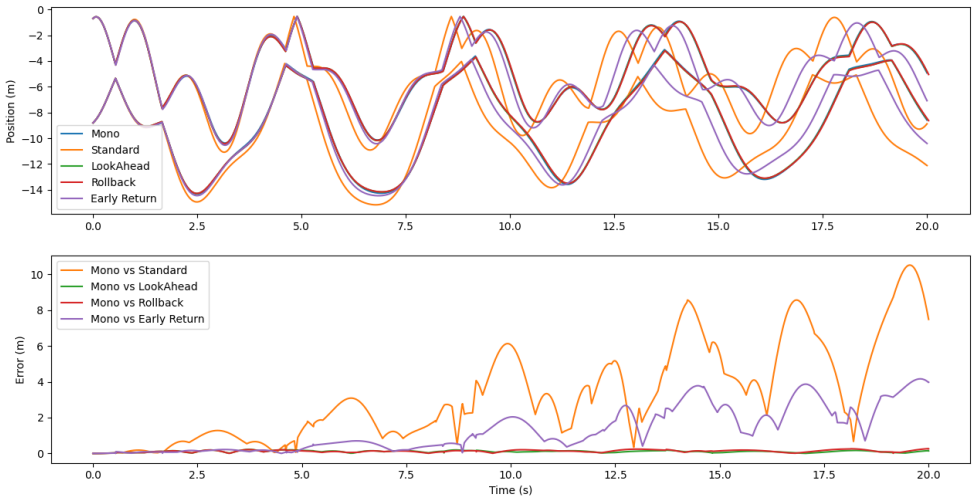


Figure 2. An overview over all 5 co-simulation results. The upper panel shows the trajectories of both blocks, while the lower panel shows the position error e_{pos} as calculated in Equation 7.

performance to the monolithic reference solution, a co-simulation algorithm without any event handling, and to each other. All three methods have their strengths, each one having the high ground in different systems.

Rollback can significantly enhance the accuracy of a co-simulation with comparatively small programming effort. This is achieved by enabling a subsystem to access and restore its internal state, and by providing a mechanism to notify the importer when such operations should occur. (We assume here that the importer supports this functionality for all three methods.) However, the main drawback lies in the potential computational cost: for complex models, saving and restoring system states can be computationally expensive. If states are stored thousands of times for an event that occurs only a few times, the overhead may outweigh the benefits. This limitation can be mitigated by storing the system state less frequently and, if a reset is required, reverting to an earlier point in time. Furthermore, if not all subsystems support Rollback, sequential coupling must be employed. In this case, the Rollback-capable subsystems are executed first, while the remaining components proceed only after a successful communication step has been completed.

Early Return has its strengths in handling private events. It not only enables recording the exact time and state at which an event occurs, but also allows this information to be propagated to other system components, thereby improving overall accuracy. To maximise this advantage, similarly to Rollback, Early Return FMUs can be calculated first in a sequential coupling scheme. The problem that we encountered in this setup is that two components utilising Early Return can give contradicting information on when an event takes place, leading to time discrepancies. This problem arises only under specific circumstances, but can lead to uncertain credibility in systems with events in more than one subsystem. Another big advantage of Early Return is its versatility in how it can be used, as aspects such as the ability to trigger Early Return from the outside (Modelica Association 2024) have not been explored in this setup.

LookAhead operates at the importer level, managing the communication step size of the entire co-simulation. While this global approach may be less optimal for a single component, it is unaffected by multiple concurrent events. When LookAhead predicts an event, it sets the next communication point and thus ensures that all subsystems are synchronized. This method remains effective regardless of the number of events or event sources, making it robust in managing both shared and simultaneous events. However, its main limitation lies in the types of events it can handle. Events must be predictable from the component's interface (its inputs and outputs) which can be challenging for events that occur deep in complex subsystems. Even though LookAhead can be applied to any black-box subsystem, knowledge about the system's events is necessary to implement its event detection. Additionally, the event must be reliably predictable: sudden state events triggered by rapidly evolving conditions can be difficult to anticipate and may indicate the need to reduce the communication step size of the continuous portion of the model.

Future work aims to improve usability by supporting internal event indicators and devising a method to automatically associate signals with events. On a technical level, LookAhead is being expanded to work with other coupling schemes than the standard parallel coupling, such as sequential coupling and parallel or sequential coupling for different communication step sizes. LookAhead is be-

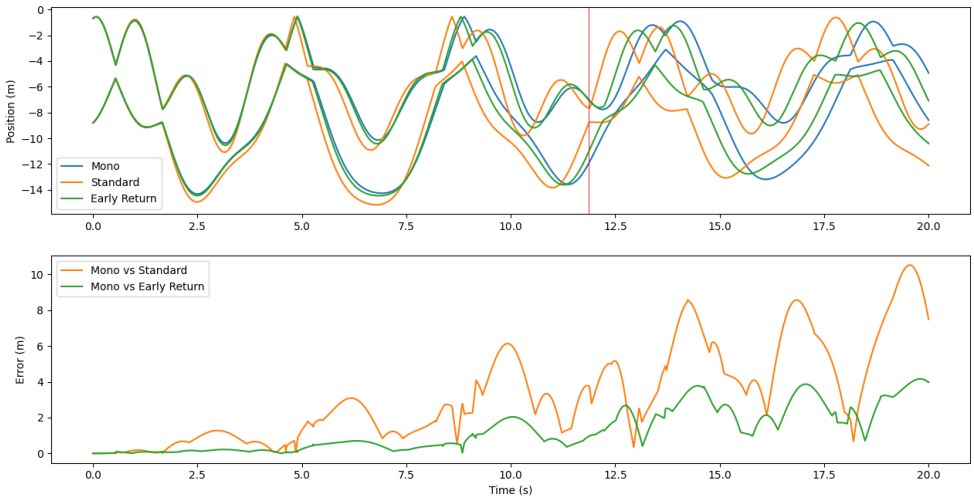


Figure 3. Comparison of the standard and Early Return co-simulation results. Although both systems diverge from the monolithic reference solution, Early Return keeps the position error much lower.

ing implemented in the co-simulation tool ICOS, developed at Virtual Vehicle Research GmbH and included in the co-simulation platform Model.CONNECT™by AVL List GmbH (AVL 2023).

Acknowledgment

The publication was written at Virtual Vehicle Research GmbH in Graz, Austria. The authors would like to acknowledge the financial support within the COMET K2 Competence Centers for Excellent Technologies by the Austrian Federal Ministry for Innovation, Mobility and Infrastructure (BMIMI), Austrian Federal Ministry for Economy, Energy and Tourism (BMWET), the Province of Styria (Dept. 12) and the Styrian Business Promotion Agency (SFG). The Austrian Research Promotion Agency (FFG) has been authorised for the programme management.

References

AVL (2023). *Model.CONNECT™, Co-simulation tool for the coupled simulation of multiple models and different tools as well as the integration with real-time systems.* URL: https://www.avl.com/en/simulation-solutions/software-offering/simulation-tools-z/modelconnect (visited on 2023-08-24).

Broman, David (2017). *Hybrid Simulation Safety: Limbos and Zero Crossings*. arXiv: 1710.06516 [cs.CY]. URL: https://arxiv.org/abs/1710.06516.

Cellier, François E and Ernesto Kofman (2006). *Continuous system simulation*. Springer Science & Business Media.

Dejaco, Daniela and Martin Benedikt (2017). "A Novel Approach for Handling Discontinuities in Non-Iterative Co-Simulation". In: *Proceedings of the 7th International Conference on Simulation and Modeling Methodologies, Technologies and Applications*. SIMULTECH 2017. Madrid, Spain: SCITEPRESS - Science and Technology Publications, Lda, pp. 288–295. ISBN: 9789897582653. DOI: 10.5220/0006440402880295. URL: https://doi.org/10.5220/0006440402880295.

Modelica Association (2024). *Functional Mock-up Interface Specification*. URL: https://fmi-standard.org/docs/3.0.2/ (visited on 2025-04-10).

Tischer, Felix et al. (2023). "LookAhead - A New Approach for Event Handling in Co-Simulation by Predicting State Events". In: *SIMUL 2023 Proceedings*. SIMUL 2023. Valencia, Spain.

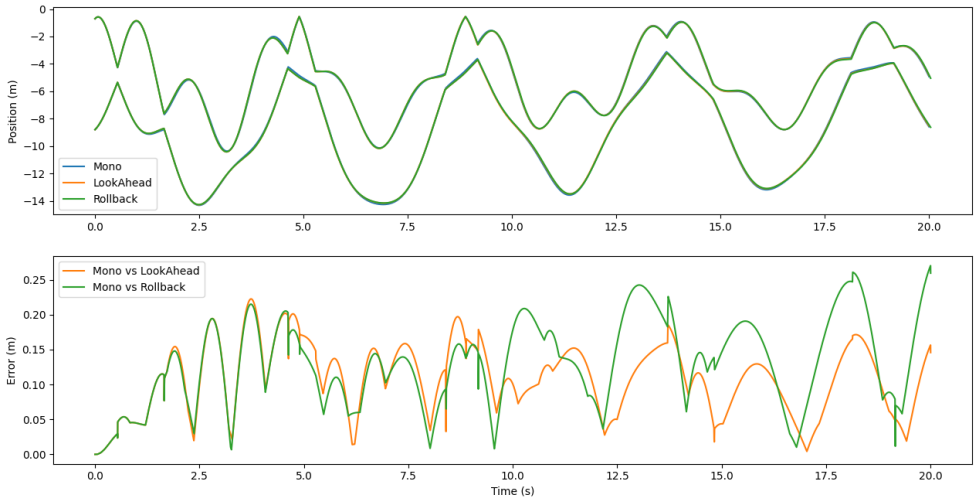


Figure 4. Comparison of the Rollback and LookAhead co-simulation results. They follow the trajectory of the reference solution very closely and keep the error at a constant magnitude. Note that the scale of the error graph is much smaller than in Figure 2 and Figure 3.

DOI
10.3384/ecp218663

Towards a Common Standard for Uncertainty Quantification

Amin Bajand[1] L. Viktor Larsson[2] Lena Buffoni[1] Elmir Nahodovic[4] Robert Hällqvist[3] Oliver Lenord[5] Hans Olsson[4] Martin Otter[6] Antoine Vandamme[5] Adrian Pop[1]

[1]Linköping University, Sweden, {amin.bajand, lena.buffoni, adrian.pop}@liu.se
[2]Parker Hannifin Manufacturing, Sweden, viktor.larsson@parker.com
[3]Saab Aeronautics, Sweden, robert.hallqvist@saabgroup.com
[4]Dassault Systèmes, {elmir.nahodovic, hans.olsson}@3ds.com
[5]Robert Bosch (GmbH), {oliver.lenord, antoine.vandamme}@de.bosch.com
[6]German Aerospace Center (DLR-FK), martin.otter@dlr.de

Abstract

Uncertainty Quantification (UQ) studies allow us to determine whether a model is fit for a particular purpose, as well as the operational domain in which it can be used. Standardising the UQ analysis setup and result summary enables the iterative composition of UQ information, which is a crucial step in evaluating model credibility. In this paper, we present an initial attempt to specify UQ information as a cross-layer standard for Modelica-, FMI-, and SSP-based workflows subject to two essential restrictions: (a) uncertainties can only be described in terms of parameters, and (b) analysis is limited to forward uncertainty propagation and sensitivity analysis of nonlinear models. More analysis features are planned for the future. The approach is illustrated using both a simple example and an industrial use case.

Keywords: Uncertainty Quantification, Credibility, Standardization, Operational Domain, FMI, SSP, Modelica

1 Introduction

Simulation models provide an approximation of reality with different degrees of accuracy, which has a direct impact on the context in which these models are usable. Through use case analysis in the ITEA OpenSCALING research project[1], we have identified the following workflows that rely on UQ:

- choosing a model with the appropriate level of representativeness and complexity for the specific application context,
- evaluating the model's overall uncertainty based on the propagation of uncertain parameters,
- performing model validation,
- conducting global sensitivity analysis of parameters, and
- improving the accuracy of data measurements.

In this paper, we propose a data model for capturing uncertainty quantification in a unified way across different workflows and thus facilitating the exchange of models, including metadata, between different stakeholders.

Furthermore, we show how this data model can be integrated into Modelica[2], Functional Mock-up Interface (FMI)[3], and System Structure and Parametrization (SSP)[4] based workflows.

The paper is structured as follows; Section 2 provides the motivation for the standard. Section 3 presents a simple forward UQ study while Section 4 introduces an industrial use case. The data model is outlined in Section 5, and Section 6 describes integration mechanisms with Modelica Association (MA) standards. Finally, Section 7 outlines future directions, and Section 8 concludes the paper.

2 Background and Motivation

Machine-interoperable traceability is vital for transparent model validation and reduces reliance on implicit knowledge transfer (Rosenlund et al. 2025; Otter et al. 2022). The *SSP Traceability standard* draft (SSP LS Traceability)[5] incorporates Simulation Resource Meta Data (SRMD) to capture verification activities, intended use, and operational domain, supporting automated model suitability evaluations. Embedding such structured metadata, either via SRMD or directly in models, streamlines validation, supports long-term credibility assessments, and aligns with industry standards like NASA STD-7009 [6]and LOTAR[7]. The recent Credible Simulation Process (CSP) initiative formalizes continuous simulation credibility processes using SSP-based layering (Ahmann et al. 2022; Heinkel and Steinkirchner 2021).

A model's credibility depends on how accurately it represents reality. Ensuring credibility requires rigorous verification, validation, and UQ frameworks, see, e.g., *the NASA Handbook for Models and Simulations* [8] (Roy and

[1] https://itea4.org/project/openscaling.html.

[2] https://modelica.org/language/

[3] https://fmi-standard.org/

[4] https://ssp-standard.org/

[5] https://github.com/modelica/ssp-ls-traceability

[6] https://standards.nasa.gov/standard/nasa/nasahdbk-7009

[7] https://lotar-international.org/

[8] https://standards.nasa.gov/standard/nasa/nasahdbk-7009

Oberkampf 2011; Riedmaier et al. 2021). UQ systematically evaluates confidence in model predictions. Techniques such as probabilistic analysis and sensitivity studies help quantify and propagate uncertainties, guiding confident interpretation of simulation results (Hällqvist et al. 2023; Otter et al. 2022).

Predictive capability extends validation by assessing model performance across the entire feasible input space, not just tested scenarios. Hallqvist et al. (Hällqvist et al. 2023) propose *entropy-based* (Shannon 1949; Kullback and Leibler 1951) and *coverage-based* (Atamturktur et al. 2015) metrics to quantify a model's numerical representativeness in both interpolation and extrapolation scenarios. These metrics provide a quantitative foundation for evaluating whether a model remains valid when applied outside its validated input and parameter space.

3 Simple Use-Case: DC Motor

This section presents a forward UQ study propagating uncertainties in a simple DC motor model. This kind of study can be used to answer engineering questions, such as whether at least 95% of the produced DC motors meet the requirement of reaching the declared idle speed. In other words, the aim is to assess the impact of production and material tolerances on the DC motor's steady-state idle speed.

It is assumed that a standardized procedure and supporting tooling are in place to enable this type of analysis independently of the chosen modeling tool. The task of developing a functional behavioral model of the DC motor is assigned to a simulation engineer, who is free to select the most appropriate modeling environment. The resulting model is delivered in executable form, accompanied by a specification of the experiment setup. This setup defines the static boundary conditions, parameter values, estimated distributions for uncertain parameters, and the quantities of interest to be analyzed. In addition, all metadata required to perform the analysis in the target environment in a repeatable and automated manner must be provided in a machine-readable format.

An investigation of the physical behavior of the DC motor found that the relevant dynamics can be captured by the balance equations for electrical voltage and mechanical torque, resulting in the two differential Equations (1) and (2) implemented in Modelica.

$$V = R \cdot I + L\frac{dI}{dt} + c_m \cdot \omega \tag{1}$$

$$J \cdot \frac{d\omega}{dt} = c_g \cdot I - d \cdot \omega + T_{load,mech} \tag{2}$$

In both equations, the following variables are defined:

- V $[V]$: Supply voltage,
- I $[A]$: Winding current,
- R $[\Omega]$: Winding resistance,
- L $[H]$: Winding inductance,
- c_m $[V \cdot s/rad]$: Electrical motor constant,
- $T_{load,mech}$ $[Nm]$: Mechanical load,
- ω $[rad/s]$: Motor velocity,
- $J[kg \cdot m^2]$: Rotor inertia,
- c_g $[N \cdot m/A]$: Mechanical motor constant,
- d $[N \cdot m \cdot s/rad]$: Mechanical friction coefficient.

To keep the investigations simple, only two physical parameters are defined as *uncertain parameters* and are described with the following two stochastic distributions:

- R: $\mathcal{N}(\mu = 0.1\ [\Omega], \sigma = 0.01\ [\Omega])$
- L: $\mathcal{U}([0.105,\ 0.115]\ [mH])$

The Probability Density Functions (PDFs) related to R and L are displayed in Figure 1, assuming a Gaussian normal distribution[9] $\mathcal{N}$ and resp. a uniform distribution[10] $\mathcal{U}$.

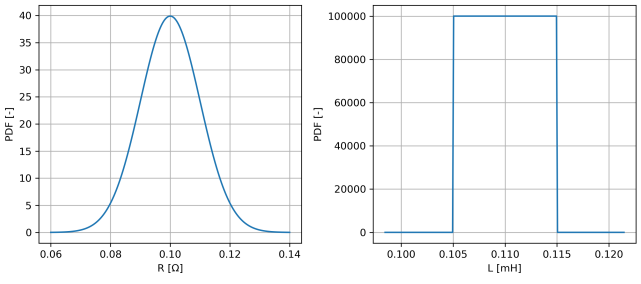


Figure 1. Representation of the uncertain parameters R and L as PDF

The remaining physical parameters are kept constant along with the voltage at the terminals and the mechanical load.

The motor is initially at rest ($I(t=0) = 0\ [A], \omega(t=0) = 0\ [rad/s]$) and accelerates in response to the input voltage V, assuming zero mechanical load torque ($T_{load,mech} = 0$), until it reaches steady-state. The objective is to quantify the uncertainty in the steady-state motor speed, expressed as $\omega_\infty[\mathrm{rad/s}]$ or equivalently $n_{\mathrm{rpm},\infty} = \omega_\infty \cdot \frac{30}{\pi}[\mathrm{rpm}]$, based on the uncertainties in the electrical parameters R and L. To this end, the uncertain parameters are sampled using Latin Hypercube Sampling, and a simulation is carried out for each sample. The resulting data is used to estimate the Probability Density Function (PDF) and Cumulative Distribution Function (CDF) of the steady-state motor speed.

All the information required to configure the forward UQ experiment, such as the problem description, model reference, parameter definitions, observed variables, experiment settings, and desired results, is listed in a corresponding `.xml` file. In this file, a reference to an `.ssv` file might be present, which is the standard way in SSP to store parameters and their values. The UQ XML file can be accessed via scripting during all stages of a forward UQ workflow, including pre-processing, simulation,

[9] https://en.wikipedia.org/wiki/Normal_distribution

[10] https://en.wikipedia.org/wiki/Continuous_uniform_distribution

DOI 10.3384/ecp218671

and post-processing. External references to result files or summaries are included where needed, see Figure 2.

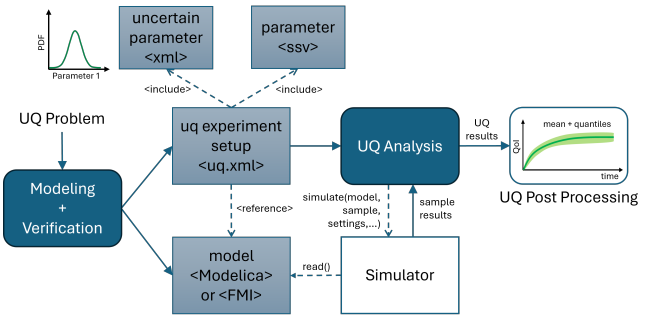


Figure 2. Forward-UQ workflow: Artifacts and processing steps

For the simple DC motor example, the following data is provided in the `.xml-file` defining the forward UQ experiment:

```
<!-- ... -->
 <uq:ParameterSet> <!-- Super set of SSP -->
  <uq:Parameters>
    <uq:UncertainParameter name="R">
      <uq:Normal mu="0.1" sigma="0.01"
        unit="Ohm"/>
    </uq:UncertainParameter>

    <uq:UncertainParameter name="dcMotor.L">
      <uq:Uniform minimum="1.05e-4"
       maximum="1.15e-4" unit="H"/>
    </uq:UncertainParameter>
  </uq:Parameters>

  <uq:Units> <!-- SSP Units -->
    <!-- ... -->
  </uq:Units>
</uq:ParameterSet>

<uq:ObservedVariables>
  <uq:ObservedVariable name ="omega"/>
</uq:ObservedVariables>

<uq:DesiredResults
  data = "dc_motor.mat"
  summary = "dc_motor_result_summary.xml"
  scope = "Trajectory"
  mean = "true"
  pdf = "true"
  cdf = "true">
  <uq:Percentiles level="0.05 0.95"/>
  <uq:SobolIndices order="2" total="true"/>
</uq:DesiredResults>

<uq:SimulationSettings
  stopTime = "0.5"
  interval = "0.01"
  tolerance = "1e-4"
  solver = "DASSL"/>

<uq:SamplingMethod>
  <uq:LatinHypercube numberOfSamples="1000"/>
</uq:SamplingMethod>
```

Note, for every uncertain parameter, a *nominal* value is required that can, for example, be used if only a single simulation run shall be performed. From this data, it is possible to perform the forward UQ experiment computations and to display typical outcomes like:

- The time response of the spreading motor speed $n_{rpm}(t)$, see Figure 3,
- The PDF of the idle motor speed in steady-state, see Figure 4 on the left,
- The CDF of the idle motor speed in steady-state, see Figure 4 on the right,
- The first-order global sensitivity analysis related to the motor speed trajectory over time, see Figure 5.

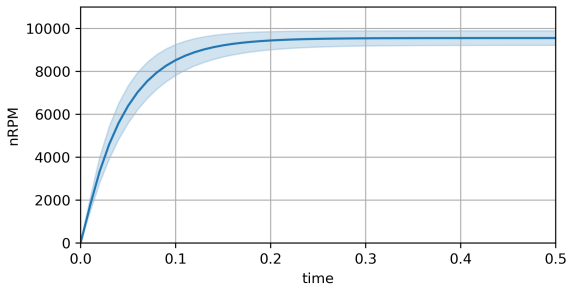


Figure 3. Mean, 5th and 95th percentile of the motor speed time response

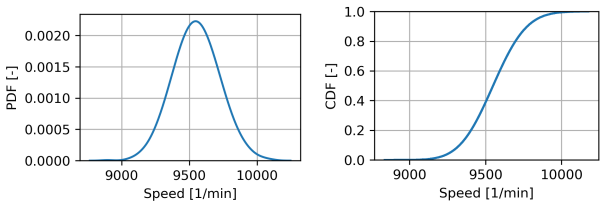


Figure 4. PDF and CDF of the motor speed at steady-state

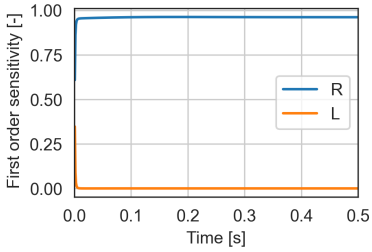


Figure 5. First order sensitivities of the motor speed w.r.t. the uncertain parameters R and L over time until reaching steady-state. The interpretation is that the uncertainty of L is of relevance only during the early start-up phase, while the uncertainty of R dominates the uncertainty of the motor speed at steady-state.

In these investigations,

- The UQ analysis of the DC motor is evaluated for only one operational point (supply voltage $V = 12\ [V]$ and constant mechanical load $T_{load,mech} = 0\ [N \cdot m]$),

- Python[11] has been used to evaluate the forward UQ experiment based on a Modelica model of the DC motor and the experiment definitions from the .xml-file, generating the samples, evaluating the simulations through OMPython[12] and OpenModelica[13], and performing the post-processing to produce the desired results.

This illustrates how the Modelica model, together with the estimated distributions of the uncertain parameter values and the .xml experiment settings based on the draft UQ standard, was used to perform Forward UQ analysis with open-source tools. The complete setup of the open-source tools is available in the draft specification[14]. In the future, the UQ .xml file will most likely be supported by many other tools, such as Modelica-based tools[15] or FMI-based tools[16].

4 Industrial Use Case: Mobile Valve

Parker Hannifin's Hydraulic Valve Systems Division Europe[17] manufactures fluid power valves for a wide range of applications. The factory in Borås, Sweden, specializes in directional valves for mobile working machines. Recently, an increased demand for valve simulation models has been noticed both from customers and internally at Parker. Simulation models are, for instance, useful in control strategy development, energy loss minimization, and functionality verification. To meet these demands, a process for auto-generation of valve models, based on valve specification data, has been developed. During this development, several challenges emerged:

- ensuring simulation tool independence,
- protecting company Intellectual Property (IP),
- shipping model credibility information, such as UQ experiments results, with the models.

The initial solution was to use the FMI standard and ship the models as Functional Mock-up Units (FMUs) providing credibility metadata in the `index.html` file inside the model's FMU directory. However, this limits the reusability of the credibility information in the simulation tool and has a negative impact on traceability, hence the need for a standardized way of shipping model credibility information with simulation models. Within the ITEA OpenSCALING project, Parker has identified a representative use case where a UQ experiment is carried out on a valve model. The UQ results are then packaged along with the model in the standard format proposed in this paper.

[11] https://www.python.org/

[12] https://pypi.org/project/OMPython/

[13] https://openmodelica.org/

[14] https://gitlab.liu.se/openscaling/work-products/d2.1-uq-layered-standard

[15] https://modelica.org/tools/

[16] https://fmi-standard.org/tools/

[17] https://www.parker.com/de/en/divisions/motion-systems-group-europe/resources/about-us/hvse.html

4.1 Valve Model

The valve model is based on the orifice equation, where the relationship between volumetric flow q and pressure drop, $\Delta P = P_1 - P_2$, across the orifice is described according to the time-invariant relationship:

$$q = A\sqrt{\frac{(P_1 - P_2)}{K}} \tag{3}$$

where the parameter K lumps the discharge coefficient and fluid density, see e.g. (Merritt 1967; Miller 1990). The orifice area, A, and K may be identified as uncertain parameters. A model of such an orifice can be defined as having two time-dependent inputs, P_1 and P_2, and two time-dependent outputs $q_1 = q_2 = q$ under the assumption of incompressible flow. The Operational Domain (OD) of this model could be viewed as a space containing all the different feasible combinations of the two inputs P_1 and P_2. The two-dimensional area, encompassing all feasible inputs, can be summarized through a set of points that define (for example) a convex hull.

The valve used in the UQ study is one work section of a proportional, load-sensing, pre-compensated mobile directional control valve in the Parker L90LS series. The core functionality of the valve is to generate a flow[18] that is proportional to an input control current, independently of pump and load pressure variations.

4.2 Scenario

A typical valve test scenario is to connect its work ports to a volume in a closed loop, feed it with constant pump pressure, and investigate the flow and counter-pressure characteristics for different input currents. Manufacturing tolerances lead to uncertainties in valve discharge coefficients, so UQ experiments must be performed to investigate the effect of these uncertainties on flow and counterpressure.

4.3 Experiment Setup

Figure 6 shows the schematics of the model used to carry out the UQ experiment. The UQ model contains the valve model, a hydraulic volume, and other valve model boundaries (e.g., constant pump supply). The valve contains Meter-In (MI) and Meter-Out (MO) orifices that are modelled according to Equation (3). The MI and MO areas (A_{MI}, A_{MO}) are determined by the main spool geometry and controlled by the main spool position via current-controlled pilot valves. In the UQ experiment described here, direction PABT is considered, which means that the main spool position is controlled by input current u_A. Pump (p_p) and load (p_A) pressure-independent flow is realized with a pressure compensator which controls the pressure drop over the MI orifice. Briefly explained, the MI orifice determines the flow (q_A), while the steady-state counterpressure (p_A) is determined by the resistance of the MO orifice for a given flow (q_A).

[18] For example, to a cylinder that controls the boom function of a forestry crane.

DOI 10.3384/ecp218671

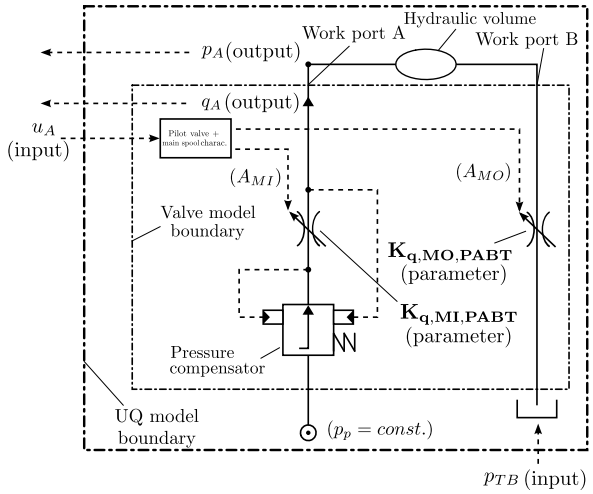


Figure 6. Schematics of the model used for UQ in the Mobile Valve use case. Variables within parentheses represent internal states.

Table 1. UQ model variables.

Variable	Causality	Description
u_A [mA]	input	Valve input A current
p_{TB} [bar]	input	Valve port TB pressure
q_A [l/min]	output	Valve work port A flow
p_A [bar]	output	Volume port A pressure
$K_{q,MI,PABT,gain}$	parameter	MI orifice discharge coefficient (rel.)
$K_{q,MO,PABT,gain}$	parameter	MO orifice discharge coefficient (rel.)

The model is packaged as an SSP-file, where the valve model FMU is connected to another FMU, exported from Hopsan [19], which contains the volume and other valve boundaries. The UQ model has the variables listed in Table 1.

According to the specification, the current $u_A \in [0, 900]$ mA, while the tank pressure $p_{TB} \in [0, 20]$ bar. The model's OD may thus be represented as a rectangle with corner points as listed in Table 2 and illustrated in Figure 7. Note that Figure 7 also shows the UQ model's Domain of Uncertainty Quantification (DoUQ), which is described in the next section.

Table 2. Valve UQ model operational domain coordinates.

Point nr.	u_A [mA]	p_{TB} [bar]
1	0	0
2	0	20
3	900	20
4	900	0

4.4 UQ Definition

For this example, UQ experiments are carried out at points whose coordinates are listed in Table 3, representing locations within the model's OD.

[19] https://liu.se/en/research/hopsan

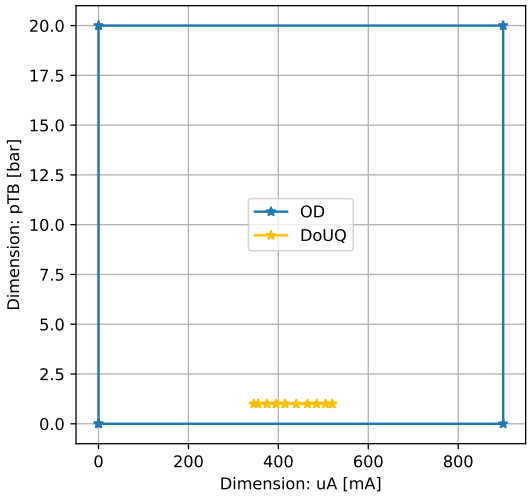


Figure 7. Operational domain (OD) and domain of uncertainty quantification (DoUQ) of the valve UQ model.

Table 3. Valve UQ model domain of uncertainty quantification points.

Point nr.	u_A [mA]	p_{TB} [bar]
1	345	1
2	355	1
3	375	1
4	395	1
5	415	1
6	440	1
7	465	1
8	485	1
9	505	1
10	525	1

Within the UQ experiment, uncertain model parameters are defined explicitly, while the resulting model output uncertainties represent the experiment outcomes. The relative discharge coefficients of the MI and MO orifices are assumed to be uncertain and follow truncated normal distributions, as specified in Table 4.

The experiments conducted here constitute forward UQ, where the uncertainties in the relative discharge coefficients are propagated through to the port A flow and pressure (q_A and p_A). For this example, the relationships between the uncertain parameters and these observed variables can be described by the simplified expressions given in Equation (4).

$$\begin{aligned} q_A &= A_{MI}\sqrt{\frac{\Delta p_{MI}}{K_{q,MI,PABT}}} \\ p_A &= p_{TB} + \left(\frac{A_{MI}}{A_{MO}}\right)^2 \Delta p_{MI}\frac{K_{q,MO,PABT}}{K_{q,MI,PABT}}. \end{aligned} \tag{4}$$

Table 4. Valve UQ model domain of uncertainty quantification coordinates. The uncertainties of the two parameters are estimated and are described with truncated normal distributions with parameters μ, σ. *: The valve model's name in the SSP-file is "valve_model".

Uncertain Parameter	Minimum	Maximum	μ	σ
valve_model.KqMI_PABT_gain*	0.8	1.2	1	0.1
valve_model.KqMO_PABT_gain*	0.8	1.2	1	0.1

4.5 UQ study in FMI

A MonteCarlo-sampling with 1000 samples was done to generate samples of the two observed variables q_A, p_A, using the UQPy python package (Olivier et al. 2020). Figure 8 shows histograms of the sample sets.

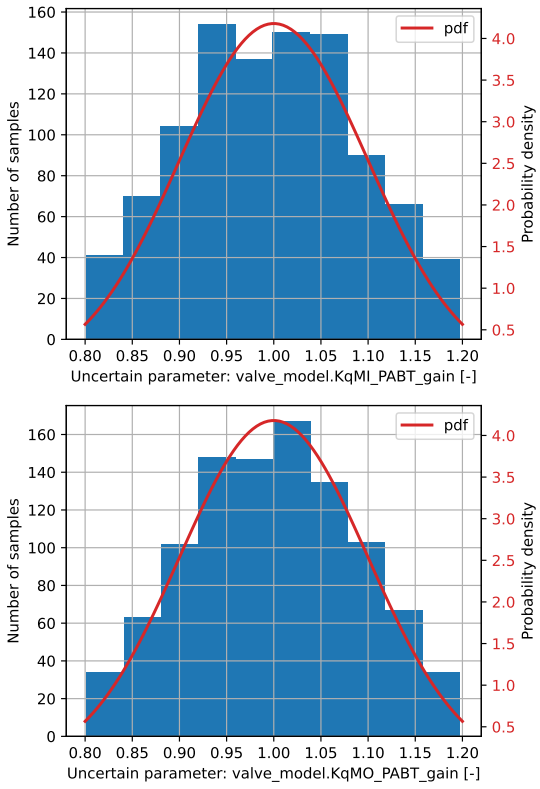


Figure 8. Histogram of 1000 samples of each of the uncertain parameters.

The UQ model was simulated with the Python API of OMSimulator. The simulation was executed for all samples at each coordinate of the DoUQ. Each simulation was run until a steady-state was reached on all model outputs. This resulted in one sample set for each output at each operating point, with 1000 samples in each set. These sample sets were then fitted to truncated normal distributions with minimum, maximum, mean (μ), and standard deviation (σ). Figure 9 shows an example of the distribution of the two observed variables at one operating point. Table 5 and 6 show the resulting fits at all experiment points for the observed variables q_a and p_A, respectively.

Figure 10 shows violin plots of the data in Tables 5 and 6. Based on these results some observations can be made:

- Uncertainty in flow q_A due to uncertainty in discharge coefficient is higher at higher current (u_A) levels, as the MI orifice area (A_{MI}) is higher at higher currents. This, in turn, results in a greater amplification $K_{q,MI,PABT} \rightarrow q_A$ (see Equation (3)). A similar reasoning may be applied to explain the larger uncertainty in p_A at higher current levels,
- Uncertainty in orifice discharge coefficients has a higher impact on the uncertainty of p_A than that of q_A. This may be illustrated by the simplified analytical expressions of q_A and p_A in equation (4); q_A primarily depends on $K_{q,MI,PABT}$, while p_A depends on both discharge coefficients.

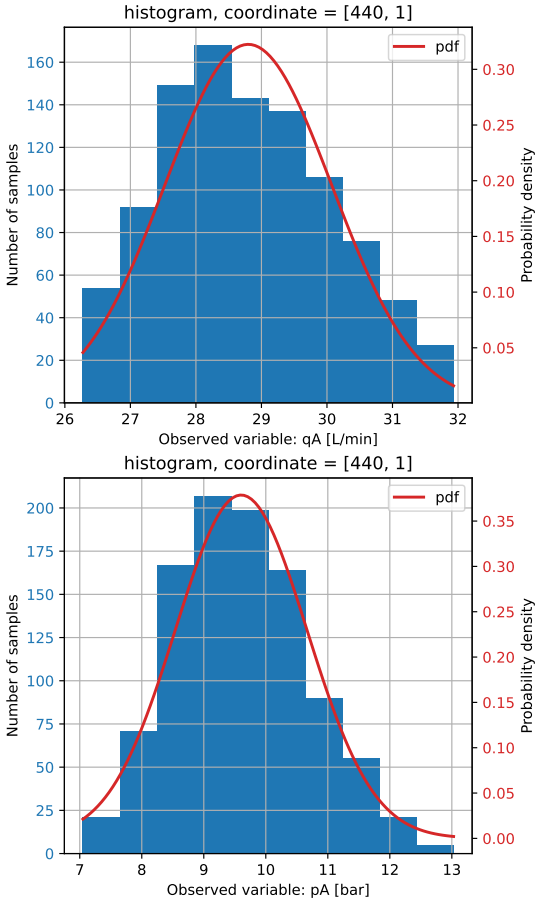


Figure 9. Histogram of 1000 samples from one Monte Carlo simulation run for each of the observed variables at operating point $u_A = 400$ mA, $p_{TB} = 1$ bar.

The .xml definition of the Forward UQ Experiment and the results are shown in Listing 1 and Tables 5-6, respectively.

Table 5. Statistical data of the observed variable q_A [L/min] for all UQ experiments fitted to truncated normal distributions.

		q_A [L/min]			
ID	Points [uA [mA], pTB [bar]]	Minimum	Maximum	μ	σ
1	[345, 1]	0.013	0.016	0.015	0.00067
2	[355, 1]	0.41	0.5	0.45	0.021
3	[375, 1]	2.6	3.2	2.9	0.13
4	[395, 1]	6.1	7.5	6.7	0.3
5	[415, 1]	12	15	13	0.6
6	[440, 1]	26	32	29	1.3
7	[465, 1]	42	51	46	2
8	[485, 1]	54	65	59	2.5
9	[505, 1]	59	71	64	2.7
10	[525, 1]	59	71	64	2.7

Table 6. Statistical data of the observed variable p_A [bar] for all UQ experiments fitted to truncated normal distributions.

		p_A [bar]			
ID	Points [uA [mA], pTB [bar]]	Minimum	Maximum	μ	σ
1	[345, 1]	1.3	1.4	1.3	0.02
2	[355, 1]	3.3	4.2	3.7	0.17
3	[375, 1]	5.0	7.3	5.8	0.36
4	[395, 1]	5.7	9.7	7.2	0.77
5	[415, 1]	6.3	12.0	8.6	0.94
6	[440, 1]	7.1	13.0	9.6	1.1
7	[465, 1]	7.3	13.0	9.9	1.1
8	[485, 1]	7.4	14.0	10.0	1.1
9	[505, 1]	7.4	14.0	10.0	1.1
10	[525, 1]	7.4	14.0	10.0	1.1

Listing 1. The `ForwardUncertaintyQuantification` element from the mobile valve use case

```
<!-- ... -->
  <uq:ParameterSet> <!-- Super set of SSP -->
    <uq:Parameters>
      <uq:UncertainParameter
         name="KqMI_PABT_gain">
        <uq:Normal mu="1" sigma="0.1"
           minimum="0.8" maximum="1.2"
           unit="-"/>
      </uq:UncertainParameter>
      <uq:UncertainParameter
         name="KqMO_PABT_gain">
        <uq:Normal mu="1" sigma="0.1"
           minimum="0.8" maximum="1.2"
           unit="-" />
      </uq:UncertainParameter>
    </uq:Parameters>
    <uq:OperationalDomain>
      <uq:Axes>
        <uq:Axis name="uA" unit="mA" />
        <uq:Axis name="pTB" unit="bar" />
      </uq:Axes>
      <uq:Boundary>
        <uq:ConvexHull>
          <uq:Point coordinates="0 0" />
          <uq:Point coordinates="0 20" />
          <uq:Point coordinates="900 20" />
          <uq:Point coordinates="900 0" />
        </uq:ConvexHull>
      </uq:Boundary>
      <uq:ExperimentPoints>
        <uq:PointSet id="PS1">
          <uq:Points>
            <uq:Point id="P1"
               coordinates="345 1" />
            <uq:Point id="P2"
               coordinates="355 1" />
            <!-- ... -->
            <uq:Point id="P10"
               coordinates="525 1" />
          </uq:Points>
        </uq:PointSet>
      </uq:ExperimentPoints>
    </uq:OperationalDomain>
    <uq:Units> <!-- SSP Units -->
      <!-- ... -->
    </uq:Units>
  </uq:ParameterSet>
  <uq:ObservedVariables>
    <uq:ObservedVariable name="qA"
     distributionApproximation="Normal" />
    <uq:ObservedVariable name="pA"
       distributionApproximation="Normal" />
  </uq:ObservedVariables>
  <uq:DesiredResults
    data="mobile_valve.csv"
    summary="mobile_valve_result_summary.xml"
    scope="FinalTime"
    histogram="true">
  </uq:DesiredResults>
  <uq:SamplingMethod>
    <uq:PseudoRandom numberOfSamples="1000" />
  </uq:SamplingMethod>
```

4.6 UQ study in Dymola

The proposal is that a Modelica tool will have the option to import the .xml-file and provide the user the choice to both modify, run, and verify the forward uncertainty quantification from the provided information. The tool should also be able to export any new UQ results and setups.

For this example, the mobile valve model is packaged as an SSP and imported into Dymola. This study will concern only one point, with coordinates $(u_A, p_{TB}) = (415\ \text{mA}, 1\ \text{bar})$.

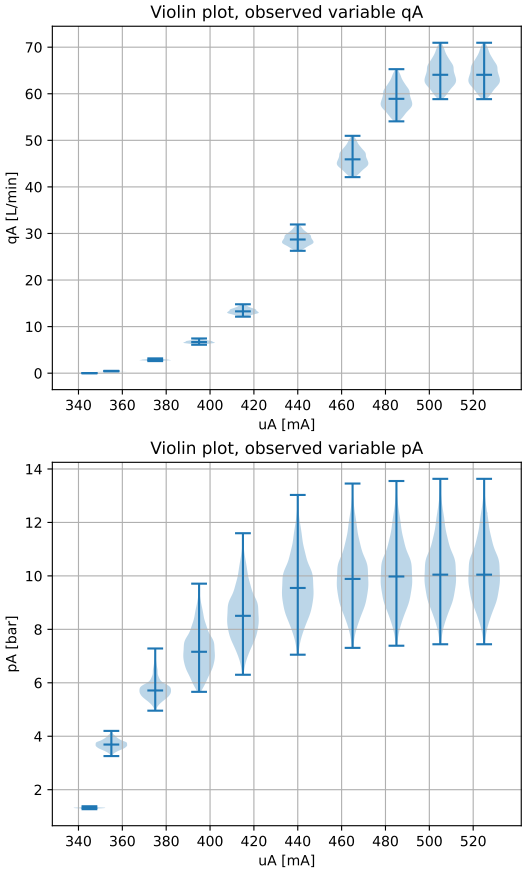


Figure 10. Violin plots of the data in Tables 5 and 6.

4.7 Verification of UQ

Using the data provided from the .xml-file, an UQ analysis with 1000 Monte Carlo samples can be performed to validate the result of previous UQ experiments. The results show that statistics such as mean value and standard deviation agree with those presented in Table 5.

In the previous step, the uncertain parameters $K_{q,MI,PABT}$ and $K_{q,MO,PABT}$ were propagated to the observed variables p_A and q_A. In addition, it can be useful to quantify exactly how much specific parameters contribute to the output variance of p_A and q_A. This helps to isolate the most important variables in a complex model with multiple parameters. The idea is to append the .xml file with additional attributes for the sensitivity analysis, such as the sampling method and the choice of sensitivity indices.

In this case, the experiment performs *global* variance-based sensitivity analysis using Sobol indices. The experiment is conducted on both p_A and q_A at the point $(u_A, p_{TB}) = (415\ \text{mA}, 1\ \text{bar})$. Three separate Monte Carlo runs with 10000 samples each are performed to evaluate q_A and p_A, where the uncertain parameters are sampled using the Latin Hypercube method. These 10000 samples are then bootstrapped 1000 times to provide confidence intervals for the first and total order Sobol indices.

The results are presented in Table 7. For the observed variable q_A, the first and total order indices are approximately equal, indicating that there are no interactions.

The Sobol indices for the parameters on the outcome p_A show that most of the variability comes from the parameters varying on their own, but there is no statistical significance in higher order interactions.

5 Data Model

The data model for UQ is designed to be tool-agnostic and compatible with multiple standards, including Modelica, FMI, and SSP. This ensures seamless integration across diverse simulation environments while maintaining flexibility to support both simple and complex UQ workflows. To achieve this, the model employs a modular structure where many elements are optional, allowing users to specify UQ data either inline within the model or in separate files as needed. The data model reuses existing SSP elements, such as `Units` and `ParameterSet`.

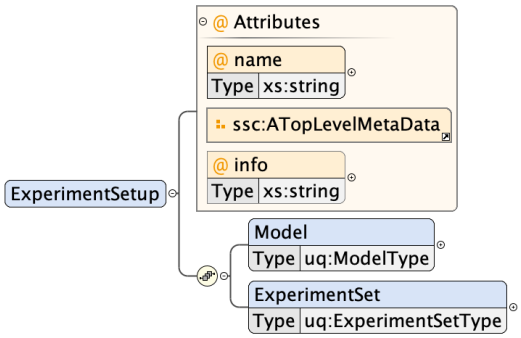


Figure 11. The structure of the `<ExperimentSetup>` element. The **`Model`** contains optional details about the simulation model, such as `name`, `file`, `model`, and `modelResourceMetaData`, the latter enabling linkage to an SRMD file for SSP traceability integration. **`ExperimentSet`** defines the experiments to be conducted on the model

The top element of the UQ model is `ExperimentSetup`, see Figure 11.
The setup can contain different types of experiments, such as `ForwardUncertaintyQuantification` (see Figure 12):

- **`ParameterSet`**: Extends the SSP `ssv:ParameterSet` by enabling the description of stochastic distributions for uncertain parameters alongside deterministic values. A reference to a standard SSP parameter file (SVV) can be used to include a standard (non-UQ) set of parameter values.
- **`ObservedVariables`**: Specifies the model variables for which results shall be computed and stored. Optionally, the attribute `distributionApproximation` may be used to indicate that the result at a given time point should be approximated by the defined distribution.
- **`DesiredResults`**: Defines which results are to be computed for all observed variables and where the resulting data shall be stored. The `data` attribute typically points to a tool-specific result file (e.g.,

DOI 10.3384/ecp218671

Table 7. The mean values and 95% confidence intervals for the Sobol Indices of the valve model, at coordinates $(u_A, p_{TB}) = (415\text{ mA}, 1\text{ bar})$, estimated using a Monte Carlo method with 10000 Latin hypercube samples bootstrapped 1000 times.

Observed variable	Uncertain parameter	First Order	Confidence	Total Order	Confidence
pA	KqMI	48.79%	+/-0.56%	50.40%	+/- 0.41%
	KqMO	49.5%	+/-0.57%	50.37%	+/-1.32%
qA	KqMI	100.00%	+/- 0%	99.44%	+/- 2.01%
	KqMO	0.01%	+/-2.01%	-0.10%	+/-4.06%

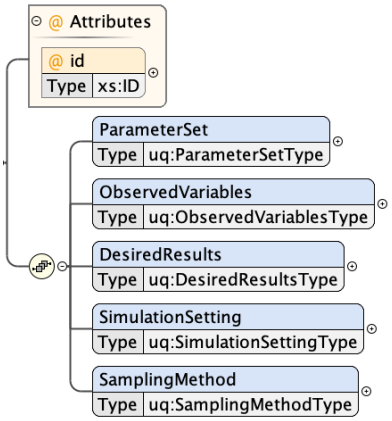


Figure 12. Structure of the `ForwardUncertainty Quantification` element

`dc_motor.mat`) and can also refer to a directory if results are stored in multiple files. The optional `summary` attribute is used to reference a separate file that contains aggregated or post-processed statistical information.

- **`SimulationSetting`**: Specifies the most important parameters for one simulation run. Optionally, tool-specific settings can be added.
- **`SamplingMethod`**: Describes the sampling method to be used in the Monte Carlo simulation, along with all required parameters for generating samples. Tool specific parameterization can included if needed.

6 Integration with MA Standards

Uncertainty data can come from multiple sources, and it needs to be easy to export and import between tools in a unified way. For instance, a component manufacturer might evaluate the accuracy of the model for a single component in Modelica and would wish to deliver this information together with the model as an FMU. This information could then be used as input to a larger UQ study for the entire system. The suggested approach is a Cross Layered Standard that uses the same definition of UQ concepts across Modelica, FMI, and SSP models.

6.1 Integration with FMI and SSP

FMI and SSP have general extension mechanisms. Every extension must keep an FMU or SSP component still compliant, so tools can ignore elements of an extension that they do not know or support.

The extension mechanism in FMI is called *layered standard* and is described in section 2.6 of FMI 3.0[20]. It is used for the UQ standard by storing the UQ XML file in a new folder inside the FMU extra-folder and by adding the required `fmi-ls-manifest.xml` in this folder.

From the SSP Traceability layered standard perspective (i.e., which information is involved related to the modeling and simulation activity, where does it come from and where shall it be propagated), the current concept would consist of referring to the *uq.xml* file containing the UQ data e.g., from further *.xml*-files that are hosted in the different activities (i.e., Analysis, Requirements, Design, Implementation, Execution, Evaluation, Fulfillment) of the Credible Simulation Process (CSP). Each file would provide information related to a given CSP activity that could be easily read and processed by a dedicated tool (e.g., *easySSP*[21] for the Requirements, *Dymola*[22] for the Execution).

6.2 Mapping to Modelica

The Modelica Language has annotations with modifier syntax that can be used to represent the information extracted from the XML UQ file in a *backwards compatible way*. A benefit of storing the information inside the model is that references to variables are syntactically variable references and not strings, and that the information is naturally moved together with the model. The Modelica Language also supports a hierarchical `Resources` folder inside a Modelica package where the different parts are referenced using Modelica URIs. In this folder, large tables could be stored, e.g., the coordinates of the experiment points in the operational domain, including the values (that can be reused for different experiments).

On the other hand, the UQ analysis supported by a tool is typically tool-specific, and a tool could provide many more features and options as standardized by the draft specification proposed in this paper. A Modelica tool would store its UQ setup in tool-specific annotations. For this reason, currently the conservative approach is used, that which no mapping of the XML-file to Modelica annotations is defined. Instead, (a) the UQ XML file can be imported in a Modelica tool and the tool stores its UQ setup in tool specific annotations, and (b) the tool can export its UQ setup and store it in the proposed UQ XML file format (tool specific settings can be stored in tool-specific

[20] `https://fmi-standard.org/docs/3.0.2/#VersioningLayered`
[21] `https://www.easy-ssp.com/`
[22] `https://www.3ds.com/products/catia/dymola`

annotations of the XML file).

7 Outlook

The data model describes a container, which can be applied on any system level, in an iterative composition-based process. This paper showcases applications in forward uncertainty propagation and sensitivity analysis of nonlinear models. The next step is to extend the specification with model validation against in-situ measurements as presented in (Hällqvist et al. 2023).

The standard presented in this paper is driven by the analysis of existing use cases and supports common concepts used in the UQ domain. In the future, if the need for additional elements is identified, the definitions can be expanded. For instance, while the current schema includes Latin Hypercube and pseudo-random sampling, and supports basic statistical outputs (e.g., mean, percentiles), future versions may incorporate additional sampling strategies, surrogate modeling techniques, sensitivity measures, and probabilistic inference methods.

The aim is to support simple as well as advanced workflows. The presented use-cases have the modeler's role at the center of the workflow and all the information is centralised in a single file; in the future, the UQ data should be traceable via the SSP Traceability Layered Standard. This will enable quality tracking and more efficient information reuse.

Another aspect to be considered is the scalability of the approach and enabling the definition of large data quantities through references to external resources, as well as being able to associate different parameterisations to the same experiment and maximize data reuse.

8 Conclusions

Storing UQ, sensitivity analysis and validation, and verification experiments in a standardized manner is an important step toward ensuring credibility and qualifying whether a model is fit for a particular purpose.

These kinds of computations require evaluations of multiple uncertain variables, which in larger models can be a very computationally complex task. An iterative approach, where the results of a previous study for a single component can be exported and composed with the results of experiments for other components, allows for managing this complexity.

A cross-layered standard is an enabler for interoperability, where studies can be performed in different simulation tools and packaged in FMI or SSP format.

The next steps are the validation of this approach on larger use-cases with composition of UQ information from several sources and the implementation of tool support for import, export, update, and analysis of UQ data in Modelica, FMI and SSP tools.

Acknowledgements

The work in this project has been conducted within the scope of the ITEA4 OpenSCALING 22013 project, with the funding of the Swedish Innovation Agency and the German Federal Ministry of Education and Research under the grant number 01IS23062.

References

Ahmann, Maurizio et al. (2022). "Towards Continuous Simulation Credibility Assessment". In: *Proceedings of Asian Modelica Conference 2022*, pp. 171–182. DOI: 10.3384/ecp193171.

Atamturktur, Sezer et al. (2015). "Defining coverage of an operational domain using a modified nearest-neighbor metric". In: *Mechanical Systems and Signal Processing*, pp. 349–361. DOI: https://doi.org/10.1016/j.ymssp.2014.05.040.

Hällqvist, Robert et al. (2023). "Toward Objective Assessment of Simulation Predictive Capability". In: *Journal of Aerospace Information Systems* 20.3, pp. 152–167. DOI: 10.2514/1.I011153.

Heinkel, H.M. and K. Steinkirchner (2021). *Credible Simulation Process Simulation-based Engineering and Testing of Automated Driving*. URL: https://setlevel.de/assets/forschungsergebnisse/Credible-Simulation-Process.pdf (visited on 2025-02-12).

Kullback, S. and R. A. Leibler (1951). "On Information and Sufficiency". In: *Ann. Math. Statist.* 22.1, pp. 79–86. DOI: 10.1214/aoms/1177729694.

Merritt, Herbert E (1967). *Hydraulic Control Systems*. John Wiley & Sons, Inc.

Miller, Donald (1990). *Internal Flow Systems*. Cranfield, Bedford: BHRA Information Services. ISBN: 978-0956200204.

Olivier, Audrey et al. (2020). "UQpy: A general purpose Python package and development environment for uncertainty quantification". In: *Journal of Computational Science* 47. DOI: https://doi.org/10.1016/j.jocs.2020.101204.

Otter, Martin et al. (2022). "Towards Modelica Models with Credibility Information". In: *Electronics* 11.17. DOI: 10.3390/electronics11172728.

Riedmaier, Stefan et al. (2021). "Unified Framework and Survey for Model Verification, Validation and Uncertainty Quantification". In: *Arch Computat Methods Eng* 28.4. DOI: 10.1007/s11831-020-09473-7.

Rosenlund, Erik et al. (2025). "Objectively Defined Intended Uses, a Prerequisite to Efficient MBSE". In: *Proceedings of the American Modelica Conference 2024*, pp. 29–42. DOI: 10.3384/ecp20729.

Roy, Christopher J. and William L. Oberkampf (2011). "A comprehensive framework for verification, validation, and uncertainty quantification in scientific computing". In: *Computer Methods in Applied Mechanics and Engineering* 200, pp. 2131–2144. DOI: 10.1016/j.cma.2011.03.016.

Shannon, C. E. (1949). "A Mathematical Theory of Communications". In: *The Bell system technical journal* 27.3, pp. 379–423.

LS-SA: Developing an FMI layered standard for holistic & efficient sensitivity analysis of FMUs

Tobias Thummerer[1] Hans Olsson[2] Chen Song[3] Julia Gundermann[4] Torsten Blochwitz[4] Lars Mikelsons[1]

[1]Chair of Mechatronics, University of Augsburg, Germany, {tobias.thummerer,lars.mikelsons}@uni-a.de
[2]Dassault Systemes AB, Sweden, {hans.olsson}@3ds.com
[3]ABB Corporate Research Center, Germany, {chen.song}@de.abb.com
[4]Keysight Technologies Deutschland GmbH, Germany,
{julia.gundermann,torsten.blochwitz}@keysight.com

Abstract

The Functional Mock-up Interface (FMI) is *the* standard for exchanging industrial simulation models in a variety of different applications. Although sensitivity analysis for continuously differentiable systems is directly supported by the standard, for systems with state discontinuities, it is only possible to determine correct sensitivities to a limited extent. In this position paper, we investigate how sensitivity analysis for discontinuous Functional Mock-up Units (FMUs), i.e. including state and time events, works in theory and which additional steps are required to obtain correct results in practice. We further investigate that these steps are unnecessarily computationally intensive from a mathematical point of view, but cannot be implemented in a more efficient way under the current restrictions of the standard. We therefore make a concrete proposal for the new *layered standard sensitivity analysis* (LS-SA) that remedies the current deficits of FMI in the sensitivity analysis of discontinuous systems. In this way, *LS-SA* opens FMI towards a variety of next-level applications — including (scientific) machine learning and optimal control — by providing fully differentiable FMUs under high computational performance.

Keywords: Functional Mock-up Unit, Functional Mock-up Interface, Discontinuous, Events, Machine Learning, Scientific Machine Learning, Neural FMU, Layered Standard

1 Introduction

We start by motivating the actual topic, followed by a brief introduction to the relevant basics to understand the paper.

1.1 Motivation & Goal

The FMI standard was created to allow the exchange of standardized models within and outside of company boundaries. The topic of Sensitivity Analysis (SA) has already played a role in the development of the standard, for example, to be able to perform uncertainty assessments on models. The groundbreaking developments in the field of machine learning in recent years have led to an increasing fusion of classical engineering disciplines such as scientific computing with machine learning, establishing the research field of *Scientific Machine Learning* (Baker et al. 2019; Rackauckas, Ma, et al. 2021). The backbone of (scientific) machine learning is efficient SA. Therefore, in our view, enhancing SA capabilities for FMUs is a crucial step toward advancing the standard for future applications in research and industry. This improvement will facilitate the use of FMI in machine learning, advanced dynamic optimization, and other fields.

Today, many industrial optimization platforms are used to improve energy efficiency, reduce operational costs, and improve sustainability in various sectors, for example, ABB OPTIMAX@ (Franke et al. 2008) and TLK-Thermo (Petr, Tegethoff, and Köhler 2017). Such platforms often operate based on FMUs because of the seamless integration of physics-based models and efficient handling of complex multi-domain systems. Enabling proper SA in FMUs becomes more and more important, because many industrial optimization problems are highly nonlinear, nonconvex and discontinuous.

While FMI already offers possibilities for SA, for example for integration of FMUs into frameworks for Automatic Differentiation (AD) (s. Sec. 3.4), these are only sufficient to perform SA correctly for continuously differentiable systems. However, for discontinuous systems, further measures are necessary for a correct SA, as we will investigate further. This is of crucial relevance for industrial applications, where most models include discontinuities.

In this position paper, we want to show what is currently possible in FMI and what should be made possible to be open to current and future research fields, e.g., neural FMUs (Thummerer, Mikelsons, and Kircher 2021), the combination of FMUs and Arifical Neural Networks (ANNs). We start by investigating two applications of current research that we believe should be made solvable with FMI. We then examine the current interface offered by FMI and collect requirements with respect to unsolved tasks within these applications. Based on these requirements, we make concrete proposals for an extension of

FMI on the basis of a Layered Standard (LS), i.e. without adjustments to the actual standard release. In parallel, we are already working on a prototype implementation, i.e. the suggestions made are already being tested.

In the following, we provide a brief introduction to the foundational concepts necessary to understand the paper.

1.2 Functional Mock-up Interface (FMI)

The Functional Mock-up Interface (FMI) is an open standard that defines an interface to exchange dynamic models in the form of a ZIP container which stores the model as a combination of XML files, binaries, and C code. There are three major releases, the most recent version is 3.0.2 (Modelica Association 2024b), while version 2.0.5 might still be the release with the broadest implementation (Modelica Association 2024a). Motivated by different application scenarios for simulation models, FMI supports three different simulation types, while we focus only on the most relevant for SA, which is Model-Exchange (ME) and is further introduced in the next paragraph. Because ME requires external implementation of the solver and event handling routines, incorporation of SA is more straightforward, for example compared to Co-Simulation (CS). Throughout the paper, for readability reasons, we only use the FMI3 command terminology, and are also focusing specifically on the development of the LS for FMI3. However, the proposed adjustments should also be compatible with FMI2 and we try to highlight the necessary adjustments if not.

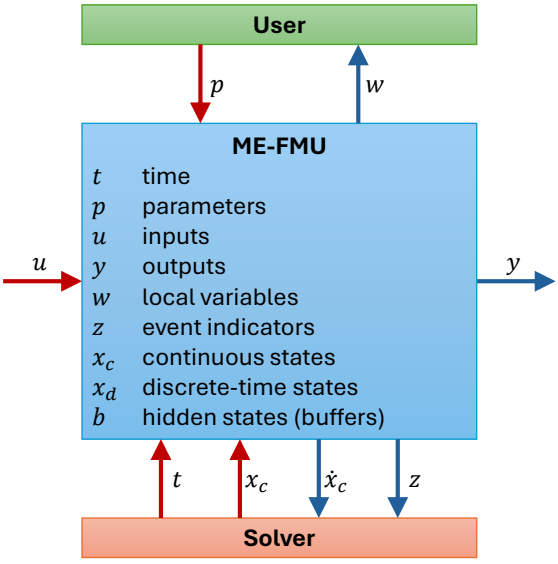


Figure 1. The ME-FMU and its interface signals. Figure adapted from the official standard document (Modelica Association 2024b).

ME-FMUs provide an interface heavily motivated by Ordinary Differential Equations (ODEs) (as further introduced in 1.4) and are therefore paired with ODE solvers to obtain a solution (referred to as *simulation*), see Fig. 1.

The FMI standard facilitates the transfer and exchange of models in simulation tools. However, it has limitations, when the calculation of a model's sensitivity is required.

Layered Standard (LS) The development of the FMI standard is subject to long release cycles to ensure quality and safety of investments, as the standard is widely used in industry. On the other hand, to allow innovation, in FMI 3.0 the concept of Layered Standard (LS) was put in place to be able to introduce new features in an optional and backward compatible way (Bertsch et al. 2023). A LS can add standardized annotations, standardized extra files, and/or new C-API functions.

1.3 Sensitivity Analysis (SA)

Sensitivity Analysis (SA) in the context of FMUs involves studying the influence on the quantities computed by the model (e.g., outputs and state derivatives) w.r.t. variations in quantities that are input to the model (like parameters, states, and inputs). Among others, SA is especially useful for parameter estimation, uncertainty quantification, and optimization.

For example, in optimal chemical production scheduling problems, a mixed-integer linear programming (MILP) solver may be often trapped in suboptimal solutions due to strict reactor start-up time constraints. One can use SA to prioritize the most influential constraints to prevent suboptimal schedules. Moreover, SA can support uncertainty quantification and risk assessment, as industrial systems operate under various uncertainties. By evaluating the range of possible outcomes w.r.t. input variations, SA enhances confidence in optimization results. A further practical example is in hydrogen plant sizing, where SA can assess the impact of load fluctuations, future demand uncertainties, and thermal constraints on transformer efficiency and longevity. This reduces the risk of over- or under-sizing, ensuring that transformers operate within safe margins while optimizing cost and reliability (Gutermuth, Primas, and Bitta 2023).

Last but not least, efficient computation of SA is essential for many modeling and analysis tasks, including hybrid modeling (combining physical and machine learning models) and scientific machine learning. Supporting SA for FMUs is therefore a key enabler for integrating large-scale industrial simulation models into data-driven workflows, es for example required in *neural FMUs* (Thummerer, Mikelsons, and Kircher 2021).

1.4 Ordinary Differential Equations (ODEs)

ME-FMUs without events can be interpreted straightforward as ODEs. The ODE within this work is denoted with

$$\dot{x} = f(x, p, t), \tag{1}$$

where the function f is referred to as *right-hand side*, x the system state, $\dot{x}$ the system state derivative, p parameters and t the time. Note that external inputs to the system could be implemented by augmenting x or p. In this work,

DOI 10.3384/ecp218681

we use this formalism for a simplified description of ME-FMUs without events.

To solve an ODE, a variety of numerical integration schemes exist, that are often incorporated into an abstract definition for arbitrary ODE solvers:

$$x(t) = ODESolve(f, x_0, p, t_0, t), \tag{2}$$

where x_0 and t_0 are the initial state and time to solve the ODE, (Hairer, Nørsett, and Wanner 1992). To obtain a time-discretized solution for f, *ODESolve* can be called recurrently, leading to a *simulation loop* as in Alg. 1.

Algorithm 1 *ODESim*: Simple simulation loop without events.

Input: right-hand side f
Input: time points t_i for $i \in \{0,...,N\}$
Input: initial state x_0, parameters p
$n = 0$ ▷ initialize step counter
while $t_n < t_N$ **do** ▷ final time not reached
  $x_{n+1} = ODESolve(f, x_n, p, t_n, t_{n+1})$
  $n = n + 1$
end while
Output: states $X = \{x_0, ..., x_N\}$

In addition, this concept can be extended to ODEs including *events*. Events are points in time where the system is allowed to change discontinuously. The time point of an event can be known in advance or depend on the system state by defining an event indicator (Modelica Association 2024b). However, this requires a different interface for *ODESolve*, because the termination time for the solver is now not known beforehand (Hairer, Nørsett, and Wanner 1992) and (Chen, Amos, and Nickel 2020):

$$t^*, x(t^*) = ODESolveEvent(f, c, x_0, p, t_0, t). \tag{3}$$

Here, t^* is the event time and the event function c is defined as zero at any event time:

$$c(x(t^*), p, t^*) = 0. \tag{4}$$

This is a simplified view, and in practice a common approach is to monitor changes in signs of a vector containing multiple event indicators, instead of an event function stopping exactly at zero. An event often corresponds to a discontinuous change in the function f. In essence, processing events (so-called *event handling*) captures the following steps: The event time point is determined (by investigation of the event function zero crossing), the solver is stopped, the discontinuous state change is performed (here, by calling the event affect function a) and a fresh integration is started from the new state after the event. The event affect function a computes a new state for after the event $x(t^+)$ based on the quantities at time t^- before the event, and is defined as:

$$x(t^+) = a(x(t^-), p, t^-) = 0. \tag{5}$$

In the actual implementation, it is distinguished between events that solely depend on time (*time events*) or also on states (*state events*). A more detailed introduction on the topic of event handling is not relevant for this work and is omitted for brevity.

Algorithm 2 *ODESimEvent* : Simple simulation loop with events.

Input: right-hand side f
Input: condition c, affect a
Input: time points t_i for $i \in \{0,...,N\}$
Input: initial state x_0, parameters p
$n = 0$ ▷ initialize step counter
$\tilde{t} = t_0$ ▷ init. current time
$\tilde{x} = x_0$ ▷ init. current state
while $t_n < t_N$ **do** ▷ final time not reached
  $\tilde{t}, \tilde{x} = ODESolveEvent(f, c, \tilde{x}, p, \tilde{t}, t_{n+1})$
  while $\tilde{t} \neq t_{n+1}$ **do** ▷ next sol. time point not reached
    $\tilde{x} = a(\tilde{x}, p, \tilde{t})$ ▷ handle event
    $\tilde{t}, \tilde{x} = ODESolveEvent(f, c, \tilde{x}, p, \tilde{t}, t_{n+1})$
  end while
  $x_{n+1} = \tilde{x}$
  $n = n + 1$
end while
Output: states $X = \{x_0, ..., x_N\}$

Further, in the FMI specification, the concept of *superdense* time is applied, allowing events to be ordered even if they are triggered for the same t. For reasons of clarity, this was simplified within this work. In general, the FMI specification (Modelica Association 2024a; Modelica Association 2024b) provides a valuable source for this topic in detail.

2 Related Work

We now turn to related work on SA for FMUs, with a particular focus on discontinuous systems.

FMISensitivity.jl The Julia library *FMISensitivity.jl*[1] implements fully differentiable FMUs using built-in FMI functions as far as available, while sensitivities not considered in FMI are approximated by finite differences. Further, SA relevant to practice, in terms of reasonable computing effort, requires additional but optional FMI features, like `fmi3Get/SetFMUState` for getting and setting the FMU memory state. The implemented workarounds are investigated in more detail in Sec. 3. Before porting the functionality to *FMISensitivity.jl* in 2023, these features were implemented as part of *FMIFlux.jl*.

Time-freezing This approach reformulates non-smooth systems with state jumps into systems that appear smooth from the perspective of a numerical solver (Nurkanović, Sartor, et al. 2021). It introduces a pseudo-time variable and clock state to enforce the discontinuities to lay

[1] `https://github.com/ThummeTo/FMISensitivity.jl`.

in the derivative rater than in the state itself. During the frozen phases, an auxiliary dynamic system evolves the state smoothly in pseudo-time to mimic the effect of the jump. As a result, the original non-smooth problem is more suitable to conventional ODE solvers without adding integer variables. However, constructing such an auxiliary differential equation is not trivial, and introducing pseudo-time variables may increase the computational effort.

Finite Elements with Switch Detection (FESD) This discretization scheme embeds switch detection directly into the time discretization process (Nurkanović, Sperl, et al. 2024). Essentially, it detects state jumps using complementarity constraints and the collocation method. Hence, FESD is able to pose the time points exactly where events happen without relying on the event handling process. The main challenges remain the implementation overhead of a robust switch detection mechanism within a finite element framework and the computational complexity due to the increase of optimization variables and constraints. An implementation of this, as well as the time-freezing approach, can be found as part of the NOSNOC software package (Nurkanović and Diehl 2022).

CasADi In (J. Andersson and Goppert 2024), the integration of SA for discontinuous systems within the *CasADi* tool (J. A. Andersson et al. 2019) is investigated. The paper also gives a rough outlook on how to make discontinuous SA available for FMUs under similar considerations to those we make in the course of this work. However, the authors omitted existing work regarding SA for FMUs: Integrate-then-differentiate, as well as differentiate-then-integrate approach, is available since 2021[2] as part of *FMIFlux.jl* and since 2023 as part of *FMISensitivity.jl*, both providing seamless integration with *SciMLSensitivity.jl* (Rackauckas, Ma, et al. 2021) and *DiffEqCallbacks.jl* (Rackauckas and Nie 2017) that provide implementations for a multitude of SA approaches.

Finite Differences The most obvious approach to SA would be to perform numerical differentiation of the output of *ODESolve*. There are two main disadvantages: a high amount of numerical noise which can be avoided by freezing the step-sizes and number of iterations or alternatively by internally differentiating (Hairer, Nørsett, and Wanner 1992), and that the cost increases linearly with the number of parameters — using adjoint equations avoids this second issue. On the other hand, there is the great advantage that even systems with discontinuities can be analyzed, which is why this method can be found in practical application despite its poor computational performance.

However, all the investigated solutions are suboptimal from a mathematical perspective and are either not generally applicable or scale insufficiently with the size of the problem due to limitations in FMI. This strongly motivates the derivation of an addition to the standard (Layered Standard), that is able to provide access to computational efficient SA, while still being fully backward compatible with FMI.

[2]Experimental 2021, stable release in 2022 (Thummerer, Stoljar, and Mikelsons 2022).

3 Requirements Analysis

In this section, we want to derive requirements for the LS. The essential part of the requirements is the availability of the various Jacobians that are needed for a correct SA. However, it is important to remember that in many cases we are actually interested in *results* of operations *including* the Jacobian, rather than the Jacobian itself. This especially captures Jacobian-Vector Products (JVPs) and Vector-Jacobian Products (VJPs). Note that JVPs are not only useful for SA, but also for solving large ODEs (Ascher and Petzold 1998) with implicit solvers using iterative approaches (e.g., Krylov subspace methods).

We start collecting requirements by investigating two use cases that should enabled with the new LS. Both applications share the question posed by SA, but look at the actual problem from two different perspectives: while a *parameter optimization* task (s. Sec. 3.1) is defined by the problem of identifying sensitivities for variables *within* the FMU, the *neural FMU* approach (s. Sec. 3.2) requires determination of sensitivities for variables *outside* the actual FMU. As a result, different sensitivities are required to solve both tasks.

3.1 Parameter Optimization

One of the most common applications of SA in the field is the optimization of ODE parameters within a model, often referred to as *model calibration*. When optimizing ODEs, an objective l is defined that is either minimized (e.g., *loss* in supervised machine learning) or maximized (e.g., *reward* in reinforcement learning). Typically, this loss is not defined on the right-hand side of the ODE itself, but on the *solution* of the ODE.

The gradient of a loss function defined on the ODE solution X, can be derived by applying the chain rule:

$$\frac{dl(X(p),p)}{dp} = \frac{\partial l(X(p),p)}{\partial X(p)} \frac{\partial X(p)}{\partial p} + \frac{\partial l(X(p),p)}{\partial p}, \quad (6)$$

where the ODE solution X can stem from *ODESim*, as well as *ODESimEvent*.

SA for *ODESim* As can be investigated in algorithm 1, the consecutive evaluation of *ODESolve* leads to the determination of X. On derivative level, this results in a chain of Jacobians containing

$$\frac{\partial x_{n+1}}{\partial x_n} = \frac{\partial ODESolve(f, x_n, p, t_n, t_{n+1})}{\partial x_n}, \quad (7)$$

growing with every step of *ODESolve* that needs to be performed to determine the required Jacobian $\partial X / \partial p$. Most

DOI 10.3384/ecp218681

ODE solvers, for example explicit and implicit Runge-Kutta methods, determine x_{n+1} by one or more evaluations of f. This results in differentiation of f with respect to state and parameters, so $\partial f/\partial x_n$ and $\partial f/\partial p$. These Jacobians can also be found by investigating the sensitivities required for the continuous adjoint method (Céa 1986; Serban and Hindmarsh 2003; Sapienza et al. 2024). If the right-hand side depends on time, further the time gradient $\partial f/\partial t$ is required. Of course, to obtain a complete derivative chain, also the arithmetic operations inside the ODE solver need to be differentiated. However, the detailed chain of Jacobians depends on the type and implementation of the numerical solver and is therefore omitted.

SA for *ODESimEvent* Very similar to *ODESim*, sensitivities for *ODESimEvent* can be collected by investigation of algorithm 2. However, there are differences: First, *ODESimEvent* also determines the event time t^* besides the system state, therefore differentiation over *ODESolveEvent* results in determination of the Jacobian $\partial x^*/\partial x_n$ and the additional time gradient $\partial t^*/\partial x_n$. Second, the event time t^* depends on the zero crossing of the event condition c, which involves differentiation of the event condition with respect to its arguments, so $\partial c/\partial x_n$ and $\partial c/\partial t$. These sensitivities can also be found by investigating the continuous adjoint method for event-ODEs (Chen, Amos, and Nickel 2020; Pfeiffer 2008). Finally, also the derivative of the new state after the event is needed by differentiation of the *affect* function a, in particular the Jacobian of the state after the event w.r.t. to the state before the event $\partial x^+/\partial x^-$.

SA for calculated Parameters In practice there are often also calculated parameters $b = g(p)$ that are computed at the start of the simulation. These calculations may be trivial (e.g., calculating the mass based on the density and volume parameters), or complicated, as, for example, calculating polynomial coefficients based on curve fitting. The set of parameters can be extended to calculated parameters to accumulate sensitivities with respect to b and then only compute the VJPs for $\partial g/\partial p$ once. Differentiating these functions should be straightforward even if they are not needed for analytic Jacobians. Parameter subexpressions can be treated together with calculated parameters, as we are not interested in individual calculated parameters but only in their overall impact.

3.2 Neural FMUs

In (Thummerer, Mikelsons, and Kircher 2021), neural FMUs where introduced as the combination of FMUs, ANNs and a numerical ODE solver. In the easiest case, only a single FMU and a single ANN are connected. Even if a common scenario is the modification of the FMU state derivative by the ANN, a variety of different combination schemes are possible. Whereas neural FMUs are in general not restricted to ME-FMUs, we only focus on ME within this work, because we see this as more relevant for the topic of proper SA, and it is definitely more relevant within the topic of neural FMUs. However, we give a short outlook on CS in the conclusion section.

As introduced, within neural FMUs, no restrictions are made with respect to the connections between the FMU and ANN(s). To make this more tangible, in Fig. 2 all possible positions to extend a ME-FMU by ANNs are given. In addition to the theoretical possibility of doing this, there is of course also the question of practical relevance, which we shall examine below by a brief investigation of fields of applications:

- One of the first applications of hybrid modeling (in terms of combining physical and machine learning models) was the **correction of parameters** by an ANN (Psichogios and Ungar 1992). This is still a meaningful measure within similar applications.
- The most valuable point of intervention is the **correction of state derivatives**. Note that this allows, for example, to incorporate additional forces (like friction) to a system because forces act on the state derivative (acceleration). Manipulation of the state derivative allows us to change the entire behavior of the system because it allows us to overwrite or extend the actual right-hand side of the ME-FMU.
- Time can be manipulated to induce, e.g., **time shifts** to compensate for unsynchronized measurements. In the parallel contribution (Thummerer, Jarmolowitz, et al. 2025), this is investigated.
- By changing the **system input**, corrections can be made to faulty signals. This further opens up the topic of learning an input trajectory to achieve *optimal control*.
- If only the **system outputs** (or local variables) are changed, the correction is learned on top of the simulation performed, without actually influencing the ODE solution, leading in general to less generalization within the ANN. However, this can still be useful in applications where generalizability is less important.
- Manipulation of event indicators allows, for example, for **augmentation of the event indicators** to learn for not modeled event conditions and therefore new discontinuities.
- **Correcting the system state** before passing it to the FMU allows for correcting modeling errors like general shifts in state-space (Thummerer, Mikelsons, and Kircher 2021) or equilibrium position for fluid simulations such as in (Thummerer, Tintenherr, and Mikelsons 2021).

Based on the finding that such architectures have practical relevance, the question arises as to which sensitivities are required for optimization including such architectures.

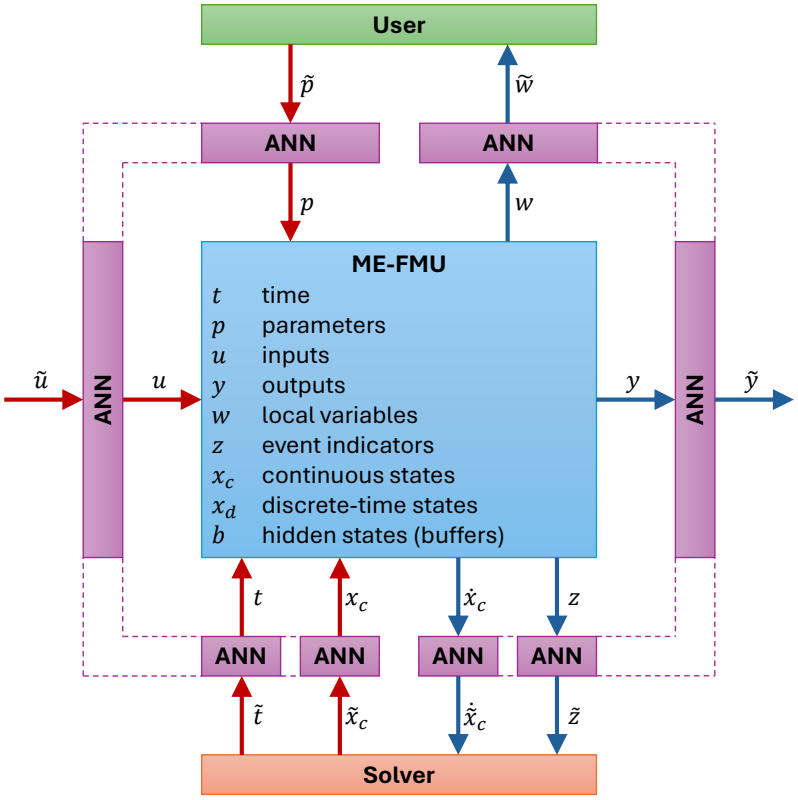


Figure 2. The neural FMU (ME), showing possible intervention points where ANNs can be incorporated to extend the ME-FMU. Individual ANNs could be used for each signal, or all causal input ANNs (p, u, t and x_c) could be merged to obtain a single input ANN (purple, dashed silhouettes). The same applies for the causal output ANNs (w, y, $\dot{x}_c$ and z). The discrete-time state x_d and the buffer b are not accessible from outside the FMU within the standard definition, granted access to x_d is part of the LS-SA, however.

Since all possible permutations of existing ANNs are to be investigated, it is expedient to assume the most complex case: All ANNs are incorporated.

3.3 Required Sensitivities

Based on the two fields of applications introduced and the corresponding observations, all necessary sensitivities are collected below. It is required to allow for JVP and VJP operations involving the Jacobians[3] $\partial\gamma/\partial\upsilon$, where $\upsilon \in \{t, x_c, u, p\}$ and $\gamma \in \{\dot{x}_c, z, y, w\}$. Furthermore, the actual event affect function that computes the new state of the system after event x_c^+ based on the system state before the event x_c^- is required but neglected in the picture to be compatible with the illustration in the standard specification. Furthermore, sensitivities are required not only for the continuous state, but also for the entire state of the system, which additionally requires the discrete state x_d. In summary, the required sensitivities are $\upsilon \in \{t, x_c^-, x_d^-, u, p\}$ and $\gamma \in \{\dot{x}_c, x_c^+, x_d^+, z, y, w\}$.

3.4 Available methods within FMI

In FMI, there are two commands available that allow to access partial derivatives over an FMU. First, `fmi3GetDirectionalDerivative` allows for JVP-multiplication with a custom seed vector, while `fmi3GetAdjointDerivative` provides functionality for VJP-multiplication. Although this does not allow one to directly access the Jacobians, it allows a straightforward integration within AD frameworks. Forward-mode AD involves consecutive evaluations of JVPs, which allows for a natural propagation of the current sensitivity seed through the FMU with `fmi3GetDirectionalDerivative`. The other way round, for reverse-mode AD, consecutive VJPs are computed and `fmi3GetAdjointDerivative` provides an interface for backpropagation of the adjoint sensitivity (or co-tangent) through the FMU. However, there are two issues with these commands: First, the implementation is optional, and, therefore, only a fraction of tools support these commands. Second, and even more important, not all required sensitivities (value references) are accessible by these commands, event if they are implemented. This is investigated in detail, along with solution proposals, in the next section.

[3] For the case of $\gamma = t$ (scalar time) it is more common to refer to a *gradient* than *Jacobian*.

4 Layered Standard

In the following subsections, we investigate all open issues derived from the introduced applications. For each issue, we will highlight the problem and available workarounds.

DOI
10.3384/ecp218681

Workarounds already implemented within *FMISensitivity.jl* and *FMIFlux.jl* are briefly described below. Finally, possible solutions and their technical implementation in the form of a LS are investigated. We explicitly try to reuse the existing command interface as much as possible, as we want to minimize the implementation effort in the tools and enable a wide adaptation of the LS-SA.

4.1 Event condition

The FMI standard allows for the calculation of derivatives of state derivatives $\dot{x}$ and outputs y with respect to states x and inputs u. If the system is subject to events, a complete SA requires the sensitivity of the point in time where an event occurs. As indicated in subsection 3.1, this requires derivatives of the event condition, or as it is named in FMI, the event indicator. Since FMI3, event indicators are part of the model variables (so they are exposed), which was not the case for FMI2. Within the FMI specification, there is no restriction that prohibits providing sensitivities for event indicators via the directional or adjoint derivative commands, however it is also not enforced.

Workaround The only generally applicable workaround to obtain partial derivatives w.r.t. event indicators is to sample them via finite differences. This way it is implemented in *FMISensitivity.jl*. The required number of samples scales linearly with the Jacobian size to multiply with, that is, linearly with the number of rows for forward AD or the number of columns for reverse AD.

Proposed Solution (LS-SA) The proposed solution for the LS-SA is straight-forward: Because the model variables of the event indicators are already known in FMI3, it is only necessary to provide sensitivities for the event indicators via `fmi3GetDirectionalDerivative` and `fmi3GetAdjointDerivative`. For FMI2, it is additionally required to expose the event indicators in the same way as they are exposed in FMI3, by adding the corresponding entries to the section `<ModelStructure>` in the model description.

4.2 Discontinuous State Change

After the event is handled, the numerical integration is restarted with a new state x^+. However, also this new state may depend on the state before the event x^-, time, inputs before the event and/or parameters. These sensitivities are not present within FMI — in general, it is not possible to access sensitivities w.r.t. to quantities *before* an event after event handling was performed.

Workaround One could sample the required sensitivity by restarting the simulation and disturbing the investigated variable right before the event (finite differences). This is computationally infeasible, because this requires a new simulation for every required sensitive state / input / parameter.

In *FMIFlux.jl*, a much more performant approach is implemented, however, this requires the optional FMI features `fmi3Get/SetFMUState`. If these commands are available, a memory snapshot of the FMU can be made right before the event. In this way, sampling becomes much more efficient because the snapshot can be used to reinitialize right before the event to disturb the signals for sampling, instead of starting a new simulation. In this way, the complexity reduces to only performing one evaluation of the system of equations rather than performing an entire simulation.

Proposed Solution (LS-SA) The challenge is that the x^+ and x^-, which are required for the Jacobian $\partial x^+ / \partial x^-$, represent the same state (model variable) at different locations in (super-dense) time. In FMI, there is no mechanism to access partial derivatives w.r.t to the same variable at different points in super-dense time, because the state before and after the event have the same value reference.

However, in the Modelica modeling language, previous values before event handling can be accessed using the function `pre`. We therefore propose to introduce new model variables that can be used to access previous values of states. This captures the continuous, as well as the discrete-time state. Because derivatives of continuous states are introduced with the `derivative` keyword within the definition of the model variable, we propose a new keyword `previous` for previous states before the event. A code snippet for this is given in Listing 1.

Listing 1. Example for a state definition within the model description, including the new *previous* attribute.

```
<Float64
    name="mass.s"
    valueReference="33554432"
    description="absolute position of mass"
    unit="m"
</Float64>
<Float64
    name="der(mass.s)"
    valueReference="587202560"
    description="derivative of mass.s"
    unit="m/s"
    derivative="33554432">
</Float64>
<Float64
    name="pre(mass.s)"
    valueReference="587202561"
    description="previous value for mass.s"
    unit="m"
    previous="33554432">
</Float64>
```

In case of multiple events occurring at the same point in time, the `previous` model variable actually refers to the *previous* value before the last event handled, and not the value before the first event at the time instant. Finally, note that this measure is not necessary for other relevant inputs like u and p, because they do not change in super-dense time, for example $\partial x^+ / \partial p^- = \partial x^+ / \partial p^+$.

4.3 Time Gradients

As investigated, the gradient with respect to time is needed. Since FMI3, time (or the independent variable

in general) is registered as special model variable with `causality="independent"`. However, there is no enforced possibility within FMI to request partial derivatives w.r.t. time.

Workaround As for the event condition, the influence of time, e.g. on the state derivative or outputs, can be sampled straightforwardly during continuous time by disturbing only the time and re-evaluation of the FMU.

Proposed Solution (LS-SA) We would like to enforce that directional and adjoint derivatives are allowed to be requested w.r.t. the independent variable during continuous time.

4.4 Event Time

In addition to event indicators that depend on state and time, events can be registered solely depending on time. The time of these events is known exactly, but only stepwise, the next event time is determined during initialization (the first time event) or during event handling (for the upcoming time events). Such events are not determined by zero crossings of event indicators, hence their influence on the FMU's outputs has to be determined separately.

Workaround To calculate the influence on the current state or parameter on the next event time, a sampling-based approach is used as for the state change in the previous Sec. 4.2: Before actually handling the current time event, a memory snapshot (`fmi3GetFMUState`) is performed, and the influence of the state and parameters can be successively sampled by disturbing one input to SA, handling the time event and quantifying the influence on the determined event time. This requires at least one sample (one event handling procedure) for every state and parameter in the system and is therefore expensive for large simulations.

Proposed Solution (LS-SA) As outlined, the next event time may also depend on the current time. Therefore, it is not sufficient to require the time to be defined as model variable (cf. Sec. 4.3), but necessary to require an additional one for the *next* event time. This enables to calculate the sensitivity of the next event time w.r.t. state and parameters as well as the influence of the current event time on the next event time. Although these sensitivities are required for SA at the next event, they already have to be calculated at the current event and need to be cached.

4.5 Parameter Sensitivities

Even if FMI supports functions to obtain directional derivatives, it is important to state that parameter sensitivities are not available during the actual simulation. Even tunable parameters are only available in event mode (Modelica Association 2024b), which is not useful during continuous-time simulation. This is of course a huge problem and actively prevents parameter SA for FMUs. There are also good reasons to not treat these parameters as tunable, as we are often (e.g., in case of parameter SA) considering the impact for the entire simulation. Furthermore, tunable parameters can complicate the storage of the memory state, and SA requires storing a large number of memory states.

Workaround To our knowledge, there is no obvious workaround to this. However, we observed that some FMU implementations do not implement this limitation and provide correct parameter sensitivities even during continuous-time mode.

Proposed Solution (LS-SA) The proposed solution is straightforward: Within the LS, we enforce that parameter sensitivities (at least for tunable parameters) must be accessible via the already existing interface for directional and adjoint derivatives.

Beyond the listed issues, we would also like to add requirements for the application of neural FMUs in the field. In addition to the mathematical considerations for a comprehensive SA, the optimization or training process can be made more efficient and robust with the following enhancements.

4.6 Get/Set Discrete State

A common technique in the training of neural FMUs, and in machine learning in general, is mini-batching. Training data is divided into chunks called *mini-batches*, and parameter gradients are determined for each chunk rather than the entire dataset. A new batch is selected for each training step, either algorithmically or randomly.

If mini-batching is paired with neural FMUs, or mixed continuous-discrete systems in general, a new challenge pops up: The system needs to be initialized for the initial state of every mini-batch. This covers the continuous state, but also the discrete-time state of the FMU. Note that the discrete state can influence the system dynamics, in the extreme case, the discrete state can be used to switch between completely different sets of equations for the continuous right-hand side. In this case, the discrete state is sometimes referred to as *mode* of the system.

By default, FMI completely hides the discrete-time state of the FMU. It is neither readable nor writable from the outside. However, this is required to initialize the system based on data. However, it is important to state that for proper initialization the continuous and discrete states must *match*. Depending on the system model, not every combination of continuous and discrete states is meaningful and leads to a solvable system.

Workaround For now, FMI does not provide an interface to directly access the discrete state, but an interface to catch it *indirectly*. The corresponding commands are `fmi3Get/SetFMUState`, which create (or activate) a memory copy of the entire FMU memory state, including the continuous and discrete state, but also additional variables, such as starting values for iteration of algebraic loops. It is also important that commands for the memory

DOI
10.3384/ecp218681

states are declared *optional* in the FMI specification and only a fraction of tools implemented these.

Furthermore, while these commands can be used to *revisit* a discrete state after capturing it once, they do not allow one to actually modify the discrete state. For example, if the system needs to be initialized in a very specific discrete state, one needs to *guide* the system to this specific state, e.g. by controlling inputs, in order to capture the intended discrete state. This is an immense effort and is generally not feasible in the case of a complex simulation model. This workaround should therefore only be regarded as a makeshift solution.

Proposed Solution (LS-SA) Even if this step is associated with technical challenges, we see no alternative to allowing reading / writing of the discrete state of the system, as is allowed for the continuous part of the state within FMI. We propose the following additions to FMI:

1. Because the data type of the discrete state may vary, compared to the continuous state which is always `Float64` / `Float32`, we do not suggest introducing new commands such as `fmi3Get/SetDiscreteState`. We propose to reuse the existing commands for reading/writing variables of the FMU, like e.g. `fmi3Get/SetInt32`. This way, no new commands are introduced, which we see as the main obstacle when implementing a LS, and there is no problem with multiple data types used for the discrete state, as it can be set/get by multiple consecutive calls (e.g. mixing boolean and integer values).

2. The value references of the discrete states must be made available from outside the FMU, in order to allow for setting/getting the discrete state. We propose to add this information to the model description, as part of the section `<ModelStructure>`. There, we like to add a new section `<DiscreteState>`, where value references of discrete states are listed in the same way as, e.g. for outputs. Because FMI does not allow us to add new fields to the standardized model description, the addition of fields does not happen within the existing model description. Technically speaking, a separate file is created consisting only of the new fields and a minimum representation of the model description XML tree around.

3. Discrete states are only required to be settable in event mode (where the discrete state changes during regular simulation), and we propose to require triggering event handling consecutively to ensure that the given continuous states match the discrete states.

4.7 Comply with Assertions

The use of assertions when modeling large systems is a meaningful measure in developing correct and easily maintainable simulation models. Typically, assertions are formulated in binary form, i.e. they remain unnoticed until a condition is violated. The existence of an assertion and the monitored condition cannot be seen outside the FMU. This has drastic effects on the application of hybrid modeling with FMUs (e.g. neural FMUs). If, for example, the system dynamics is changed by an ANN, the violation of a condition within an assertion can only be detected when it is already too late and an exception is thrown, which often leads to the termination of the simulation and thus the SA.

To our knowledge, there is **no workaround** to this.

Proposed Solution (LS-SA) We propose to introduce a concept related to the event indicator interface, which we call straightforward *error indicators*. Error indicators provide a distance measure for the violation of modeling errors. Unlike event indicators, the sign is important. Negative error indicators identify the non-existence of a modeling error, whereas positive error indicators show that a modeling error occurred. For the case of at least one positive error indicator, we do not expect the FMU to provide meaningful values, e.g. returning `fmi3StatusError` or `fmi3StatusDiscard` for calls to `fmi3GetX`. To implement error indicators, we propose the following modifications:

1. Error indicators are introduced as model variables in the same way as event indicators, and are made public in the section `<ModelStructure>` in the model description, where we add a new subsection `<ErrorIndicator>`, again in the same way as for `<EventIndicator>`. As for the discrete state in 4.6, this measure is implemented in a parallel model description file and not by modification of the standardized FMI model description.

2. To make the values of error indicators accessible, two obvious strategies are available. First, the error indicators could be accessed through `fmi3GetFloat64` as any other (`Float64`) model variable. For now, we see no reason to not enforce all error indicators to be of type `Float64` (for 64-bit FMUs). Second, because all event indicators share the same data type, a new command could be introduced in analogy to `fmi3GetEventIndicators`, see code 2, that allows capturing all error indicators at once.

3. As proposed for event indicators, we also like to keep error indicators differentiable. Although this is not required for proper SA, this opens up an interesting and valuable new use case: Explicit regularization. If not only the distance to an error is known, but also in which direction parameter changes influence the error (*gradient*), error indicators can be used to actively prevent errors during training, for example by adding the (differentiable) error indicators to the training objective as additional summands (referred to as *regu-*

larization terms). Sensitivities could be accessed in analogy to the event indicators.

Listing 2. The new error indicator function.

```
typedef fmi3Status fmi3GetErrorIndicators(
    fmi3Instance instance,
    fmi3Float64 errorIndicators[],
    size_t nErrorIndicators);
```

Although we consider all the proposed measures within this section important, we have sorted them in descending priority order (with the aim of carrying out a correct SA) in order to support a targeted implementation.

5 Conclusion

In summary, we have investigated the opportunities that already exist for SA in FMI. By investigating two example applications (parameter SA and neural FMUs), we were able to identify deficits, particularly in the analysis of discontinuous systems. We went on to consider how we could close these gaps (mathematically and technically) and made a concrete proposal for the LS-SA, which will be able to solve the issues investigated under reasonable computational performance.

Current & Future Work Even if the LS is not yet fully implemented, a dedicated working group within the research project *OpenSCALING*, including tool vendors, users, and academia, continues to develop and implement the LS to be able to publish it during the project period. As soon as more prototypes of tool implementations are available, we want to examine and quantify the practical computational benefits of the LS besides the theoretical considerations.

One major part of future work is a more detailed investigation and specification of the proposed measures, e.g. it is necessary to specify allowed (or forbidden) attributes for the newly introduced model variables.

Of course, the development of LS-SA is also strongly driven by *specific* applications for which we are aiming, which will benefit greatly from the realization of the standard extension. For example, training of neural FMUs, the combination of ANNs and physics-based simulation models captured in an FMU, will benefit from a significantly faster training process (multiple factors, depending on the number of states and events).

In theory, the proposed LS-SA can be extended for CS-FMUs. However, this introduces additional challenges because parts of SA, which are now part of the simulator, need to be incorporated to the FMU. Further, a clever interface needs to be discussed that allows for efficient determination of sensitivities, for example by defining required sensitivities before starting the actual simulation (e.g. during the setup of experiment).

This publication lays the methodological foundation for further discussions on a large and importing topic within the standard, as well as prototype implementations within importing and exporting tools. It is important for us to point out that the LS is in active development, but not officially released. We would like to draw the community's attention to the development, as well as the intended measures, and we hope to achieve a broad discussion of this topic in the community so that a strong LS with broad adoption can emerge from this. We welcome any kind of participation.

Funding

This research was partially funded by ITEA4-Project OpenSCALING (Open standards for SCALable virtual engineerING and operation) No. 22013. For more information, see: `https://openscaling.org/` (accessed on 30 April 2025).

Author Contributions

Conceptualization, T.T., O.H., S.C., G.J., B.T., M.L.; methodology, T.T., O.H., G.J., B.T., M.L., S.C.; writing—original draft preparation, T.T., O.H., S.C., G.J., B.T., M.L.; writing—review and editing, T.T., O.H., S.C., G.J., M.L.; visualization, T.T.; All authors have read and agreed to the published version of the manuscript.

Abbreviations

AD Automatic Differentiation

ANN Arifical Neural Network

CS Co-Simulation

FMI Functional Mock-up Interface

LS Layered Standard

FESD Finite Elements with Switch Detection

FMU Functional Mock-up Unit

JVP Jacobian-Vector Product

ME Model-Exchange

MILP mixed-integer linear programming

ODE Ordinary Differential Equation

SA Sensitivity Analysis

VJP Vector-Jacobian Product

References

Andersson, Joel and James Goppert (2024). "Event support for simulation and sensitivity analysis in CasADi for use with Modelica and FMI". In: *Modelica Conferences*, pp. 99–108.

Andersson, Joel AE et al. (2019). "CasADi: a software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation* 11, pp. 1–36.

Ascher, Uri M. and Linda R. Petzold (1998). *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Philadelphia, PA: Society for Industrial and Applied Mathematics. DOI: 10.1137/1.9781611971392.

Baker, Nathan et al. (2019). *Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence*. Tech. rep. USDOE Office of Science (SC), Washington, DC (United States).

Bertsch, Christian et al. (2023). "Beyond FMI-Towards New Applications with Layered Standards". In: *Modelica Conferences*, pp. 381–388.

Céa, Jean (1986). "Conception optimale ou identification de formes, calcul rapide de la dérivée directionnelle de la fonction coût". In: *Revue Française d'Automatique, Informatique, et Recherche Opérationnelle (RAIRO)* 20.4, pp. 371–402.

Chen, Ricky T. Q., Brandon Amos, and Maximilian Nickel (2020). "Learning Neural Event Functions for Ordinary Differential Equations". In: *CoRR* abs/2011.03902. arXiv: 2011.03902. URL: https://arxiv.org/abs/2011.03902.

Franke, Rüdiger et al. (2008). "Model-based online applications in the ABB Dynamic Optimization framework". In: *Proc. of the 6th Int. Modelica Conference, Bielefeld. www. modelica. org/events/-modelica2008/Proceedings/sessions/session3b1. pdf*.

Gutermuth, Georg, Bernhard Primas, and Jan Bitta (2023). "Sizing matters". In: *ABB Review*. URL: https://new.abb.com/news/detail/103158/sizing-matters.

Hairer, E., S.P. Nørsett, and G. Wanner (1992). *Solving Ordinary Differential Equations I Nonstiff problems*. Second. Berlin: Springer.

Modelica Association (2024a-11). *Functional Mock-up Interface for Model Exchange and Co-Simulation. Document version: 2.0.5*. Tech. rep. Linköping: Modelica Association. URL: https://github.com/modelica/fmi-standard/releases/download/v2.0.5/FMI-Specification-2.0.5.pdf.

Modelica Association (2024b-11). *Functional Mock-up Interface Specification. Version 3.0.2*. Tech. rep. Modelica Association. URL: https://fmi-standard.org/docs/3.0.2/.

Nurkanović, Armin and Moritz Diehl (2022). "NOSNOC: A Software Package for Numerical Optimal Control of Nonsmooth Systems". In: *IEEE Control Systems Letters* 6, pp. 3110–3115. DOI: 10.1109/LCSYS.2022.3181800.

Nurkanović, Armin, Tommaso Sartor, et al. (2021). "A Time-Freezing Approach for Numerical Optimal Control of Nonsmooth Differential Equations With State Jumps". In: *IEEE Control Systems Letters* 5.2, pp. 439–444. DOI: 10.1109/LCSYS.2020.3003419.

Nurkanović, Armin, Mario Sperl, et al. (2024). "Finite elements with switch detection for direct optimal control of nonsmooth systems". In: *Numerische Mathematik* 156.3, pp. 1115–1162.

Petr, Philipp, Wilhelm Tegethoff, and Jürgen Köhler (2017). "Method for designing waste heat recovery systems (WHRS) in vehicles considering optimal control". In: *Energy Procedia* 129, pp. 232–239.

Pfeiffer, Andreas (2008). "Numerische Sensitivitätsanalyse unstetiger multidisziplinärer Modelle mit Anwendungen in der gradientenbasierten Optimierung". PhD thesis. Martin-Luther Universität Halle-Wittenberg.

Psichogios, Dimitris C. and Lyle H. Ungar (1992). "A hybrid neural network-first principles approach to process modeling". In: *AIChE Journal* 38.10, pp. 1499–1511. DOI: https://doi.org/10.1002/aic.690381003.

Rackauckas, Christopher, Yingbo Ma, et al. (2021). *Universal Differential Equations for Scientific Machine Learning*. arXiv: 2001.04385 [cs.LG].

Rackauckas, Christopher and Qing Nie (2017). "DifferentialEquations.jl – A Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia". In: *The Journal of Open Research Software* 5.1. DOI: 10.5334/jors.151.

Sapienza, Facundo et al. (2024). *Differentiable Programming for Differential Equations: A Review*. arXiv: 2406.09699 [math.NA]. URL: https://arxiv.org/abs/2406.09699.

Serban, Radu and Alan C Hindmarsh (2003). *CVODES: An ODE solver with sensitivity analysis capabilities*. Tech. rep. Citeseer.

Thummerer, Tobias, Fabian Jarmolowitz, et al. (2025). "Br(e)aking the boundaries of physical simulation models: Neural Functional Mock-up Units for Modeling the Automotive Braking System". In: *16th International Modelica & FMI Conference, Lucerne, Switzerland, Sep 8-10, 2025*. (accepted).

Thummerer, Tobias, Lars Mikelsons, and Josef Kircher (2021). "NeuralFMU: towards structural integration of FMUs into neural networks". In: *Proceedings of 14th Modelica Conference 2021, Linköping, Sweden, September 20-24, 2021*. Ed. by Martin Sjölund et al. ISBN: 978-91-7929-027-6. DOI: 10.3384/ecp21181297.

Thummerer, Tobias, Johannes Stoljar, and Lars Mikelsons (2022). "NeuralFMU: Presenting a Workflow for Integrating Hybrid NeuralODEs into Real-World Applications". In: *Electronics* 11.19. ISSN: 2079-9292. DOI: 10.3390/electronics11193202. URL: https://www.mdpi.com/2079-9292/11/19/3202.

Thummerer, Tobias, Johannes Tintenherr, and Lars Mikelsons (2021-11). "Hybrid modeling of the human cardiovascular system using NeuralFMUs". In: *Journal of Physics: Conference Series* 2090.1, p. 012155. DOI: 10.1088/1742-6596/2090/1/012155.

DOI
10.3384/ecp218681

FMI Layered Standard for Network Communication: Applications in Networked ECU Development

Christian Bertsch[1] Kahramon Jumayev[2] Andreas Junghanns[3] Pierre R. Mai[4] Benedikt Menne[5]
Masoud Najafi[6] Tim Pfitzer[1] Jan Ribbe[3] Klaus Schuch[7] Markus Süvern[5] Patrick Täuber[5]

[1]Robert Bosch GmbH, Germany, {Christian.Bertsch, Tim.Pfitzer}@de.bosch.com
[2]Akkodis Germany Consulting GmbH, Germany, Kahramon.Jumayev@akkodis.com
[3]Synopsys, Germany, {Andreas.Junghanns, Jan.Ribbe}@synopsys.com
[4]PMSF IT Consulting, Germany, pmai@pmsf.eu
[5]dSPACE GmbH, Germany, {BMenne, MSuevern, PTaeuber}@dspace.de
[6]Altair, France, Masoud@altair.com
[7]AVL List GmbH, Austria, Klaus.Schuch@avl.com

Abstract

This paper introduces the FMI 3.0 Layered Standard for Network Communication (FMI-LS-BUS), an extension of the Functional Mock-up Interface 3.0 (FMI 3.0) standard designed to address interoperability challenges in simulating distributed, networked systems, particularly in automotive applications. By leveraging FMI 3.0 features such as clocks, clocked variables, and hierarchical terminals, the standard defines two complementary abstraction layers:

Physical Signal Abstraction (High-Cut): Representing physical signal values as clocked variables.

Network Abstraction (Low-Cut): Emulates hardware-level bus protocols (e.g., CAN, Ethernet) using FMI 3.0's clocked binary variables.

Aligning with the V-model development process, we demonstrate how these layers address distinct challenges in different design phases: High-Cut supports requirements engineering and functional testing by simplifying signal exchange during Virtual Electronic Control Unit (vECU) integration. Low-Cut enables later phases of the design validation by replicating network timing and protocol specific properties, such as error handling.

The standard's applicability currently focuses on automotive use cases (e.g., CAN, CAN FD, CAN XL, Ethernet, FlexRay, LIN) but can be extended to industrial automation and IoT, facilitated by its domain-agnostic structure.

Keywords: FMI, layered standard, network communication, bus, V-model, High-Cut, Low-Cut, Physical Signal Abstraction, Network Abstraction, FMU, vECU

1 Motivation

The development of complex cyber-physical systems in the modern automotive industry pushes the boundaries of engineering complexity, demanding new standardized methods, virtual validation, and simulation-based testing practices (Rausch 2022). As vehicles become increasingly software-driven, the need for virtual validation is particularly evident since advanced vehicle functions like Advanced Driver Assistance Systems (ADAS) are extensive and resource-intensive (dSPACE 2025). Real-world testing and hardware-in-the-loop (HiL) systems alone cannot adequately cover all test scenarios. Existing development methodologies, such as the Vehicle Development V-model (Müller-Ettrich 1998), provide a well-established framework for system design and verification. However, these methods must be complemented by adapting processes, methods and tools to the new requirements. Therefore, new approaches are being developed to validate vehicle functions, ECU software components and network communication. Strengthening support for communication protocols and system interactions is essential to ensure performance, reliability, and functional safety in the development of modern vehicles.

Furthermore, the development of these complex, cyber-physical systems and their components is highly distributed, with development of critical functionalities spread across multiple suppliers and engineering teams, introducing significant challenges in requirements specification, system integration and validation (Lawrenz 2013). A single vehicle's electronic architecture may involve numerous Original Equipment Manufacturers (OEMs) and a diverse network of tier suppliers, such as Tier 1 and Tier 2, each with different roles in the development process of ECUs. These ECUs rely on various automotive bus protocols, such as CAN, LIN, FlexRay, and Automotive Ethernet (Rausch 2022), to exchange data and coordinate system behavior. Each supplier may employ different simulation environments, modeling methodologies, and protocol implementations, making it difficult to achieve a unified, end-to-end system simulation. Ensuring seamless interoperability between simulation tools and across different teams and companies has become essential. Engineers encounter substantial challenges in:

- Verifying end-to-end system behavior across diverse

ECUs and networks

- Ensuring consistent communication between heterogeneous components and network technologies
- Validating system performance at different stages of development
- Reducing time-to-market while managing escalating development costs
- Maintaining comprehensive system reliability, safety, and compliance with standards such as ISO 26262 (ISO 2011)

A key enabler for addressing these interoperability challenges is the Functional Mock-up Interface (FMI) (Blochwitz 2011) (Blochwitz 2012) (Junghanns 2021), which is a non-proprietary, free standard for model exchange and co-simulation, allowing simulation artifacts from different sources and different simulation tools to interact seamlessly instead of existing proprietary and incompatible solutions from different software manufacturers. By integrating bus simulations into an FMI-based workflow, engineers can achieve a more modular, reusable, and interoperable simulation environment. Since the FMI 3.0 standard does not cover bus or network communication, the Layered Standard (Bertsch 2023) FMI-LS-BUS extends the capabilities of FMI 3.0 by standardizing network simulations using only features of FMI 3.0. This enables the validation of communication protocols at both the frame level (Low-Cut simulation) and the signal level (High-Cut simulation) while ensuring consistency across different tool chains (dSPACE 2024).

2 Concepts

This section delves into the general concept of the FMI-LS-BUS. First, the existing system compositions are presented. Second, the two specified abstraction layers are explained and differentiated.

The FMI 3.0 standard itself allows interoperability between the tool chains of different software manufacturers and reflects a fundamental need and the necessity of users who have to develop and secure software in a distributed manner. For example, across company boundaries in an OEM and supplier relationship or across departmental boundaries in large companies. Based on this stable foundation, the FMI-LS-BUS (MAP FMI 2025a) specifies an extension of the FMI 3.0 standard that enables the simulation of network communication. This extension supports various bus systems such as CAN, CAN FD, CAN XL, FlexRay, Ethernet and LIN and uses native FMI 3.0 features such as binary variables, clocks and terminals. With two different abstraction layers, the High-Cut and the Low-Cut, it also offers users the necessary flexibility and variety to simulate a network of controllers at their current phase of development.

- Physical Signal Abstraction ("High-Cut"): Use individual, clocked signal variables to transport logical, unit-based values between vECUs, ignoring transport layer-specific properties. The layered standard for this abstraction basically defines how bus signals have to be described in the model description file.
- Network Abstraction (Low-Cut): This abstraction allows for the implementation of virtual bus drivers within FMUs on the level of a hardware abstraction layer. It uses clocked binary variables to exchange bus operations between FMUs based on a lightweight protocol defined by the layered standard. Bus operations are used to transmit bus messages as well as bus events like acknowledge or error events. This enables both ideal and more realistic bus simulations depending on the capabilities of both sides of the API: The FMU and the importer. These capabilities can include timing, arbitration, error handling, status monitoring and other effects.

Since the FMI-LS-BUS is constrained to the description of the FMU interface, it remains neutral and agnostic to the actual bus communication process. For supported bus technologies, convenient templates are available to keep the integration effort low.

2.1 System Compositions

The FMI-LS-BUS supports three different ways of system compositions for integration into a specific simulation platform. Each individual system composition offers different advantages that are relevant depending on the user's concrete needs. The integrated FMUs across the different system compositions are the same, so that the same FMU can be integrated in all three communication architectures.

2.1.1 Direct Connection

The first option is to use a common FMU importer. Within this configuration, the FMU importer does not require any special features for simulating buses, apart from supporting FMI variables, clocks and terminals. Figure 1 illustrates the direct communication of two FMUs.

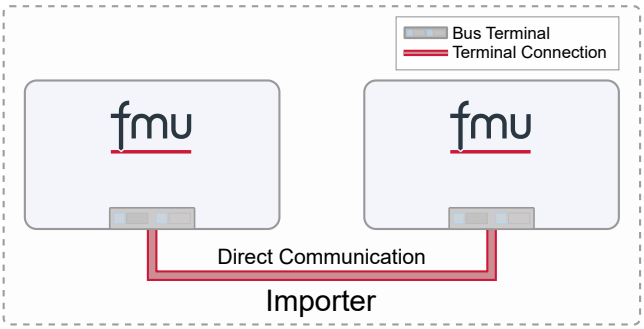


Figure 1. The first system composition allows the connection of exactly two FMUs using a common FMU importer to simulate bus systems.

DOI 10.3384/ecp218693

Direct bus communication is limited to exactly two FMUs. The idealized bus simulation does not consider bus-specific properties like the simulation of transmission timing or arbitration and, therefore, differs from physical networks. The main points resulting from this are as follows:

- Network congestion/bandwidth: The respective bus transmits the specific messages at an infinite speed. Baud rate considerations are not taken into account.
- Arbitration of network frames: All frames are transmitted at the same time without delay, which means that network-specific priority rules are not considered.
- Bus-specific protocol functions: All frames are transmitted at the same time without any delay. Protocol-specific functionalities, such as the acknowledgment of frames in CAN or any other network protocol, are ignored.
- Incoming buffer overflow: The Network FMU will receive all frames, regardless of buffer size or other hardware limitations and would need to handle those internally.

2.1.2 Composition with a dedicated Bus Simulation FMU

To simulate more realistic network characteristics, a bus simulation must usually be integrated, which simulates bus-specific properties like arbitration or the actual transmission time. One option is to implement this behavior in a separate FMU, the so-called Bus Simulation FMU. This FMU is responsible for simulating the respective bus behavior depending on the specified bus type (e.g., CAN, Ethernet, or any other bus) and simulates transmission times and failures of bus messages. Also it realizes the connections between the different FMUs that want to exchange network data (Network FMUs). The primary idea behind this concept is: All Network FMUs send their bus messages to the Bus Simulation FMU, which can confirm, delay or reject, and forward them. This approach allows for detailed bus simulations using common FMU importers. Figure 2 shows two Network FMUs connected to a Bus Simulation FMU; a total of three FMUs are running on an FMU importer.

This type of communication allows for the simulation of all required bus features:

- Transmission timing: An essential and one of the important features of bus-specific simulations is the consideration of transmission times for specific transmissions.
- n:m Bus Communication: The Bus Simulation FMU can provide the appropriate number of inputs and outputs to connect all Network FMUs within the complete bus network.

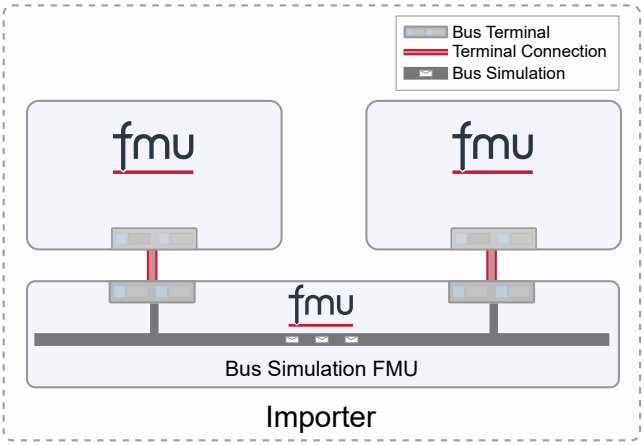


Figure 2. Bus simulation using a dedicated Bus Simulation FMU allows the simulation of n:m bus systems and very detailed simulations based on common FMU importers.

- Failure injection: Both the FMI-LS-BUS and this system composition in which all network data is routed via central Bus Simulation FMU(s) allow the possible targeted injection of transmission errors for simplified testability.
- Arbitration of network frames: Depending on the bus type and the specific arbitration requirements, appropriate implementations can be carried out via the central Bus Simulation FMU.

2.1.3 Importer with integrated Bus Simulation

Within this system composition, the functionality of the Bus Simulation FMU is integrated into the importer or simulator, so it allows the integration of the FMI-LS-BUS into an already existing manufacturer-specific simulator. These simulators may implement network communication with proprietary interfaces.

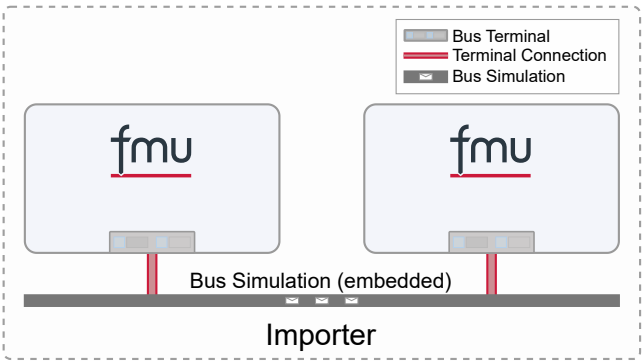


Figure 3. Bus simulation by using an importer with integrated Bus Simulation allows proprietary simulators to integrate FMI-LS-BUS specific FMUs.

This kind of system composition supports the same bus features as the setup with a dedicated Bus Simulation FMU, but with importer-specific benefits or constraints.

Figure 3 shows two Network FMUs integrated by an importer that supports the FMI-LS-BUS standard without the need for an additional Bus Simulation FMU.

2.2 Physical Signal Abstraction ("High-Cut")

Physical Signal Abstraction, or shortly called High-Cut, allows the exchange of unit-based physical signals between FMUs. Variables representing these physical signals are clocked to reflect the bus timing aspects of the information flow. The activation of these clocks always indicate that new network data is available at a given point of time.

As shown in figure 4, the core idea of the High-Cut consists of the bus signals, which are modeled as FMI 3.0 variables. These FMI variables are grouped into Protocol Data Unit (PDU) terminals, which are then grouped into frame terminals to declare their structure according to the network description.

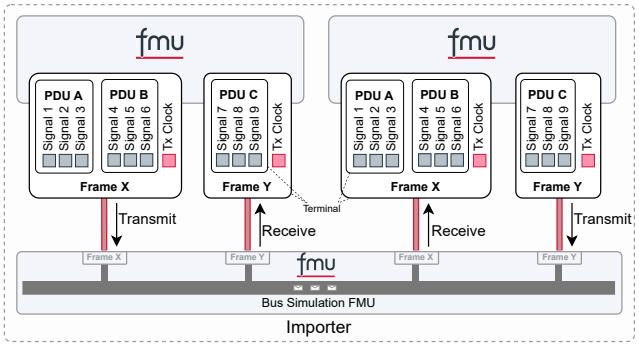


Figure 4. The figure shows an example architecture of a Physical Signal Abstraction. The signals (Signal 1…9) are modeled as clocked FMU variables of a specific type. The corresponding signals are structurally combined using PDUs, which in turn are assigned to frames. The PDU and frame structuring is done via terminals (PDU A, B, C and Frame X, Y).[1]

This approach of loose coupling between the physical signals and the lightweight PDU/frame grouping communication structure allows for easy creation of such High-Cut-based Network FMUs without the need for complex communication stacks. These configurations make them perfect for the rapid creation or prototyping in an agile, iterative development of the final communication structure. The High-Cut is therefore primarily suitable for early development phases, where its data-based simplicity offers efficiency advantages, rather than being disrupted by the fact that it lacks complete accuracy in simulating specific bus systems.

2.3 Network Abstraction ("Low-Cut")

The Network Abstraction, or shortly called "Low-Cut", allows the exchange of bus operations between FMUs over binary clocked variables. Bus operations are defined for different bus types and allow the emulation of the respective bus system above an electrical level. The focus here is on the level of abstraction for virtual scenarios and execution performance. Each operation consists of an OP-code and optional arguments. The primary operation is *Transmit*, which indicates that a Network FMU, connected to the bus, intends to send a payload. The payload content mirrors that of a real bus payload, with additional arguments to provide necessary communication frame data, as shown in figure 5 for a *CAN Transmit* bus operation corresponding to a CAN Standard Data Frame. A corresponding Bus Simulation typically responds with a *Confirm* operation to acknowledge receipt of the frame, but it may also reply with a *Bus Error* or other bus system specific operations. The set of provided bus operations ensures that a particular bus system can be simulated to the required level of detail.

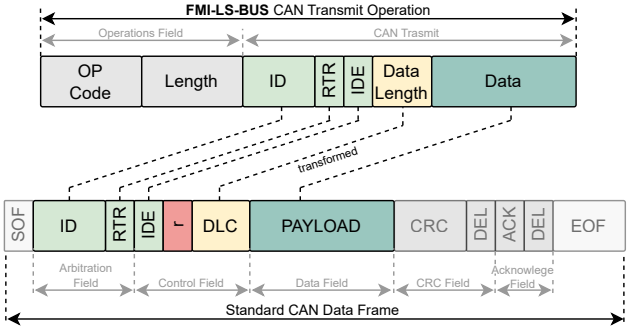


Figure 5. Shows the mapping of the FMI-LS-BUS *CAN Transmit* bus operation to a CAN Standard Data Frame. The information that can be mapped directly is shown in green. The yellow representation describes transformed values and the information shown in red is not used.

In contrast to the High-Cut approach, only one terminal per bus is needed. It consists of a binary variable and a clock variable for each direction of communication (transmission and reception). With the aid of bus specific operations, definition any message on the particular bus can be transported.

Thus, the sum of the instruments of the bus operations, as well as the bus terminals specified for the interconnection of Network FMUs, provides the possibility to use the Network Abstraction within the presented system compositions (see section 2.1). Figure 6 illustrates two Network FMUs integrated via an importer with an integrated Bus Simulation.

Network Abstraction allows a detailed and precise simulation of bus systems. This allows extensive tests using a suitable simulator in later development stages, when it comes to ensuring the functionality of a communication stack and software based on it. Currently, the bus systems planned for Network Abstraction are CAN, CAN FD, CAN XL, FlexRay Ethernet and LIN. However, this list might be extended to include other domains in the future.

[1] The term *Tx Clock* was chosen as an abbreviation for *Transmission-Clock* within the figure.

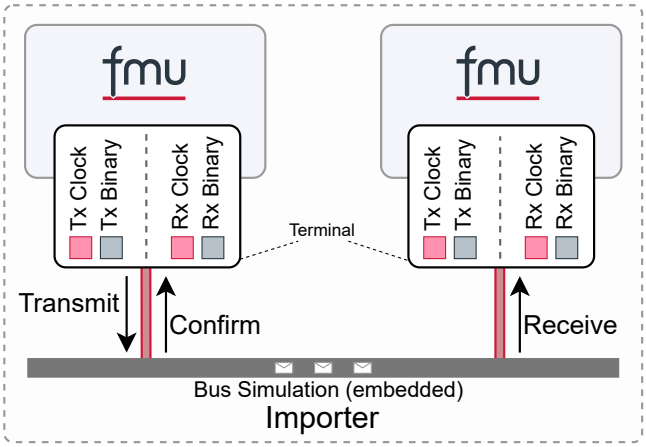


Figure 6. Two Network FMUs integrated via an importer which is coming with integrated Bus Simulation support, based on the Network Abstraction.

2.4 Principle of FMI-LS-BUS

Based on FMI 3.0, FMI-LS-BUS defines the API between FMU and importer and its semantics: What a variable and its values mean, how and when to exchange values and how to interpret those values.

Neither FMI 3.0, nor FMI-LS-BUS, specify what happens inside FMUs and importers. Both standards are agnostic of the FMU generation process (e.g. hand written, generated, wrapper or tool connector) or its content (e.g. AUTOSAR (AUTOSAR R24-11) vECU, file access for replay, aerospace line-replaceable unit (LRU)).

By not defining the internals of either side of the FMI-LS-BUS API's participants, FMI-LS-BUS can be used for many different use cases, as described above: Rough or detailed functional behavior, function emulations or production behavior wrapped in a thin FMI-LS-BUS wrapper.

3 Benefits of Integrating FMUs with FMI-LS-BUS within the Development Process

This section proposes an enhanced V-model, tailored to address the complexities of modern automotive development (see figure 7). This approach uses FMUs as central, standardized exchange artifacts across the entire development lifecycle. Being inherently tool-agnostic, FMI proves particularly valuable in distributed development by facilitating seamless integration at the interfaces between diverse domains (e.g., controls, sensor fusion, vision), distinct development stages (design, implementation, testing), and across heterogeneous tool chains.

During the initial design phases on the left side of the V-model, requirements decomposition occurs, leading to system and architectural decisions, including function distribution. This involves the definition of software components (SWCs) and their specific network interactions. These designs, including functional behavior, parameters, interface definitions, and network information, are then abstracted and encapsulated into executable FMUs, potentially leveraging FMI 3.0 features and the FMI-LS-BUS.

The availability of executable FMUs enables early and continuous virtual integration testing (e.g., MIL/SIL) on the verification branch of the V-model. For example, FMUs representing distinct ECUs, potentially created from multiple suppliers, can be simulated together using FMI-LS-BUS. By enhancing conventional methods (which typically rely on static documentation and interface specifications) by integrating executable specifications, we can substantially reduce ambiguity and error potential. A key benefit of this integration is the ability to formulate and apply test cases early on, enabling the early identification and resolution of inconsistencies that might otherwise arise from interpretation mismatches or format translations between different development stages and tools. This includes managing inconsistent signal definitions across different tool domains, simulating realistic network requirements, and validating end-to-end timing, interactions, and protocol adherence.

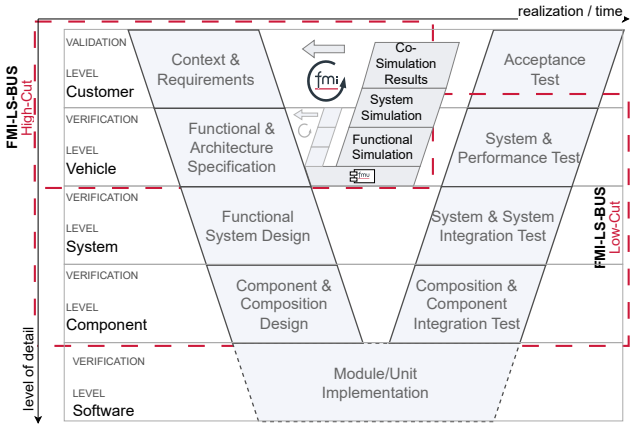


Figure 7. Enhanced V-model incorporating FMI-LS-BUS. This model illustrates the application of High-Cut and Low-Cut FMUs throughout distinct phases of the automotive development and verification life cycle.

This tight coupling between design and validation aspects of the V-model, facilitated by executable FMUs, enables an agile, iterative methodology, thereby supporting Continuous Integration/Continuous Testing (CI/CT) practices. Smaller, faster development cycles for components or subsystems provide rapid feedback. This significantly reduces integration risks later in the process and enhances overall efficiency, traceability, and the capacity to manage the complexity inherent in modern vehicle compositions. Furthermore, the standardized nature of FMUs promotes model reuse across different projects, simulation environments, and even between collaborating organizations (e.g., OEM and Tiers), maximizing the return on modeling effort.

Figure 7 illustrates this enhanced V-model framework that integrates FMI-LS-BUS to support both High-Cut and

Low-Cut FMUs, targeting different phases in the development process. Typically, High-Cut FMUs encapsulate application logic or functional behavior (often derived early in the design), while Low-Cut FMUs focus on detailed network interface behavior and network stack implementations (like AUTOSAR Classic), enabling verification at different integration levels as development progresses.

3.1 System Requirements and Architecture Definition

The High-Cut variant addresses the "top-left" of the development process, including requirements decomposition, system architecture definition, and the allocation of functions to logical or preliminary architectural components (see figure 8).

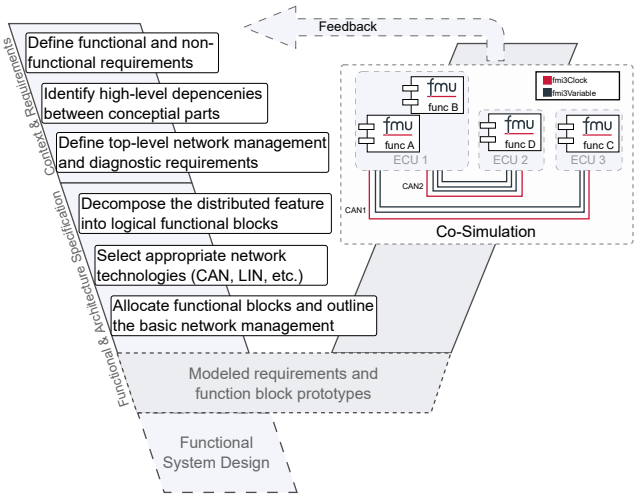


Figure 8. High-Cut use cases in the V-model. A proposed workflow for a quick development testing feedback loop using FMI-LS-BUS High-Cut features.

In this context, FMUs become executable design objects that embody requirements and components, offering a precise and verified definition early in the development lifecycle. The relationship between requirements and executable models improves traceability, verification, and impact analysis. The interoperability of the FMI standard can be a significant improvement in addition to traditional document-centric methods, allowing experts to integrate models from diverse tools like MBSE platforms and domain-specific simulators. These models serve as preliminary executable definitions that yield insights more rapidly and accelerate the specification process.

One example usecase is the allocation of distributed functions within a vehicle network, where a function is spread across multiple components. Simulation can be used to assess the impact of moving a function from one control unit to another, revealing how the change affects system behavior and communication. FMI-LS-BUS describes the interaction between these functional FMUs as part of automotive networks at a High-Cut abstraction level by modeling the interaction between these FMUs at the "signal level. This abstraction emphasizes the validation of functional data exchange between components while omitting the characteristics of network protocols.

The primary validation objective in this design phase is to confirm the architecture's functional accuracy, validate data flows, and detect conceptual errors in distributed logic before they are implemented. By enabling Verification and Validation activities much earlier than conventional, late-stage integration methods, High-Cut simulations with FMI-LS-BUS allow teams to identify and correct architectural flaws before they propagate into more costly development stages. Crucially, even with signal-level abstraction, this approach facilitates the early evaluation of closed-loop behavior, capturing essential timing factors and discretization effects that might otherwise be overlooked. Moreover, these FMUs serve as a shared, executable specification that improves collaboration between system architects and implementation teams. This single source of truth provides a clear, unambiguous, and validated artifact that can be shared with suppliers, which is especially beneficial when developing functions across organizational boundaries.

3.2 ECU Software Development and Component Testing

Building on the validated architecture, this iteration concentrates on the detailed development of Software Components (SWCs) and their integration into virtual ECUs (vECUs). Function developers and software engineers refine the FMUs with the executable specification or create new, more detailed FMUs representing concrete SWCs or vECUs, potentially incorporating auto-generated or hand-written code. The adoption of FMUs supporting FMI-LS-BUS substantially enhances the workflow, enabling early virtual integration and testing of software long before hardware is available. For instance, an OEM can easily integrate and test vECUs from multiple suppliers in a common simulation environment. This early, hardware-independent validation results in robust vECUs that serve as reliable deliverables for the next stage, significantly accelerating the overall development cycle.

In advance, network communication can be verified at a more detailed "Low-Cut" abstraction level using FMI-LS-BUS. This involves simulating the AUTOSAR communication stack, including the packing and unpacking of signals into network messages (e.g., CAN frames) based on standard description formats like DBC or ARXML. Simulating these vECUs together allows for thorough testing of the communication software, ensuring the implementation meets network interface requirements. This helps developers identify critical errors, such as incorrect signal packing or protocol deviations, preventing them from becoming costly problems during later hardware integration phases. The simulation can also be extended to model key characteristics of the bus itself, enabling the analysis of network-level phenomena such as timing effects and the correctness of error-handling mechanisms like mes-

DOI
10.3384/ecp218693

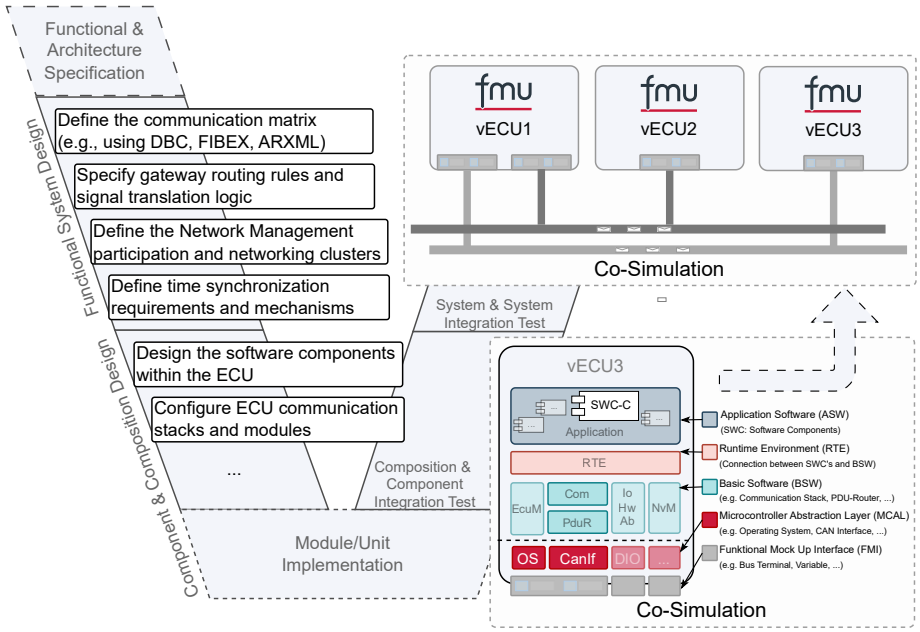


Figure 9. Low-Cut use cases in the V-model. This figure illustrates a proposed workflow for a quick development testing feedback loop using FMI-LS-BUS Low-Cut features.

sage retransmission.

3.3 System Testing

Proceeding from the integrated vECU developed and validated in section 3.2, this final virtual iteration assembles the individual vECUs into a complete system simulation. The primary objective is comprehensive testing in a realistic environment to create a high-confidence virtual release candidate. System integration and test teams utilize the collection of vECUs alongside potentially more sophisticated Bus Simulation FMUs (see section 2.1.2) or use an integrated Bus Simulation (see section 2.1.3) representing the network bus itself. These Bus Simulations, also leveraging FMI-LS-BUS (Low-Cut), incorporate enhanced capabilities beyond simple message transport, such as detailed timing models and error injection mechanisms, providing a realistic virtual testbed. The standardized FMU interface within this iteration affords significant flexibility, allowing test teams to readily combine vECUs from various sources (different internal teams or external partners) with advanced bus models and test harnesses, potentially originating from different tool vendors. This enhances test flexibility, coverage, and the ability to simulate realistic system conditions, ensuring functional requirements and communication specifications are met at the system level.

The validation at this system level is thorough, intended for ensuring end-to-end correctness and identifying integration problems that isolated component tests may not detect. The procedure include validating functional performance throughout the complete virtual system and evaluating detailed network attributes such as latencies, scheduling conformity, and bus load. By systematically injecting network errors—such as message loss, data corruption, or node failures—through the advanced bus simulation, the system's reaction can be properly assessed against its requirements. Successful completion of this virtual testing phase provides high confidence in the system's correctness, performance, and robustness. The validated set of integrated FMUs represents a comprehensive virtual prototype, ready to be shared with other stakeholders (e.g., for calibration or HiL testing) as part of the distributed development and release process, ensuring consistency prior or in parallel to hardware deployment.

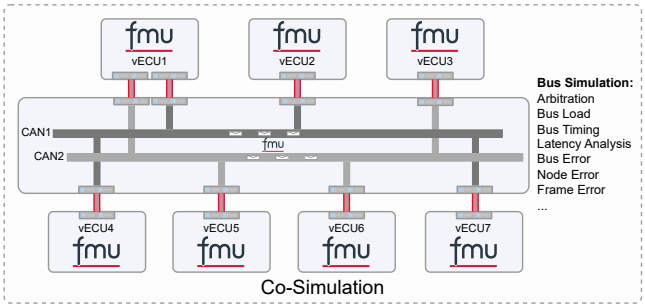


Figure 10. This figure illustrates a total of seven VECU FMUs connected via two virtual CAN bus lines (CAN1 and CAN2). A portion of the VECU FMUs is connected via both bus lines and thus communicates in both CAN networks. Using the Low-Cut approach, very detailed and global tests on a system level are possible to ensure the functionality of the various vECU FMUs in various scenarios.

3.4 Operation and Maintenance

The integration of FMUs connected via FMI-LS-BUS extends its benefits beyond initial development and testing into the vehicle's operational phase and subsequent main-

tenance activities. Virtual models and test environments created during the development life cycle, encompassing a wide range of potential vehicle configurations, become valuable assets for post-release tasks.

Managing Variant Complexity with Virtual Fleets: A virtual fleet enhances troubleshooting capabilities. When addressing field-reported issues potentially specific to certain variants, the corresponding virtual configuration can be quickly instantiated to reproduce the fault condition. Validated system representations, containerized as FMUs, are used within these simulations. Field recordings are utilized to stimulate the simulation. Therefore, the Low-Cut abstraction is especially useful since the captured traces can be easily transformed to the FMI-LS-BUS datagram. Furthermore, operational data collected from vehicles in the field can then be added into the simulation database to improve and correlate the original FMU models, gradually increasing accuracy over time.

Quality Assurance of Updates Across Variants: A virtual system environment, representing the necessary variants from the virtual fleet, is crucial for validating software upgrades before deployment, especially for Over-the-Air (OTA) updates. Prior to releasing updates, their compatibility and performance can be rigorously tested across all relevant deployed configurations within the FMI-LS-BUS simulation system. This pre-deployment validation against the virtual fleet acts as essential quality assurance, significantly reducing the risk of introducing variant-specific defects or regressions into operational vehicles, thereby safeguarding the safety and reliability of OTA updates across the diverse product range.

Digital Twin for Long-Term Maintenance: The containerized nature of FMUs simplifies archiving specific system configurations and component baselines for each variant (Coïc 2021). This supports consistent, traceable long-term maintenance across the entire vehicle portfolio. A specific configuration from the virtual fleet, potentially updated with operational data for a particular physical vehicle, can function as a key component of that vehicle's Digital Twin, enabling variant-specific analysis and prediction.

Service Training: The library of FMU models, covering various vehicle configurations, serves as an effective tool for the supplementary training of service technicians. They can learn about variant-specific systems and diagnostics in a virtual environment, enhancing competence without the constraints and costs associated with accessing diverse physical hardware.

3.5 Conclusion

By establishing FMUs as standardized, executable exchange artifacts throughout the proposed "enhanced" V-model development process, FMUs close critical gaps between development domains, stages, tools, teams and companies. The iterative application of this methodology, leveraging both High-Cut (signal-level) and Low-Cut (message-level) FMI-LS-BUS abstractions, enables the following significant benefits:

This methodology improves efficiency, reduces integration risks and costs, enhances quality and reliability, fosters collaboration, and manages the inherent complexity in developing modern, distributed automotive electronic systems through standardization, virtualization, and early, continuous validation.

4 Validation of FMI-LS-BUS

Interoperability is an essential aspect of modern model-based systems engineering, enabled by standards like the FMI. FMI encourages exchange and simulation of dynamic models created with various authoring tools. The FMI 3.0 Layered Standard for Network Communication (FMI-LS-BUS) has been developed to extend these capabilities to distributed systems and network-based simulation architectures. The demonstrated interoperability across a wide range of technologies and organizations is an important measure of such standards' success and practical adoption.

At the ASAM International Conference 2024 (Bertsch 2024), the FMI-Design Group demonstrated the seamless interoperability between different tools. This demonstration showcased the High-Cut and Low-Cut capabilities based on automotive simulation use cases.

Well-known SiL software tool vendors such as Akkodis, Altair, AVL, Bosch, dSPACE, PMSF IT Consulting and Synopsys participated in this demonstration with their tools (Bertsch 2024).

5 Other usage domains

The current focus of standard network protocols predominantly targets technologies utilized within the automotive sector. However, other industries, such as shipping, aerospace, off-road machinery, and agriculture, employ distinct network technologies tailored to their specific requirements. Examples include CANaerospace, ARINC 667, AFDX, MIL-STD-1553, and ISOBUS.

With the successful integration of automotive network technologies into the FMI-LS-BUS, there is potential for incorporating additional domain-specific technologies. This process can be facilitated by interested parties from both within and outside the FMI community. Such integrations may occur with or without formal endorsement from the Modelica Association Project FMI.

6 Summary and Outlook

The increasing complexity of products and development processes necessitates not only the standardization of signal exchange but also the harmonization of network message exchange for simulation artifacts. The FMI-LS-BUS addresses this need by providing a standardized framework for network simulation, building upon the widely adopted FMI 3.0 standard. By introducing two abstraction layers, the standard enables flexible utilization of network

DOI 10.3384/ecp218693

communication throughout all phases of product development. Early development stages can leverage high-level abstractions (High-Cut), while later stages can adopt low-level details (Low-Cut) as more specific network behaviors are defined.

FMI is already a standard for many OEMs and suppliers when developing and exchanging vECUs. The FMI-LS-BUS, as a key technology for cross-manufacturer description of bus simulations in FMUs, will strengthen the core FMI standard itself and allow users to use it in new and unprecedented areas of application. To facilitate implementation, a range of resources is available, including an FMI-LS-BUS Implementers' Guide (MAP FMI 2025b) and reference FMUs, which are accessible via the FMI website *www.fmi-standard.org*. These resources aim to streamline integration efforts for tool providers and their developers.

Regarding a possible description format for the system interconnection of buses described in FMUs for the FMI-LS-BUS, the SSP standard (MAP SSP 2024) is a suitable option. Current analyses regarding the possible specification are part of ongoing work. This could offer further simplifications for users.

Additionally, stakeholders are encouraged to participate in the FMI community or develop their own layered standards on top of FMI 3.0. Such initiatives can occur independently or in collaboration with the FMI project, fostering innovation and extending the applicability of FMI standards across diverse domains.

Acknowledgments

We would like to thank all contributors to the FMI Standard and the Layered Standards built on top of it. Numerous ideas and feedback from FMI users and the Advisory Board supported us throughout the design process and provided valuable insights. We thank the companies, especially those of the authors, for their support, which enable us to continue the standardization.

References

AUTOSAR (R24-11). *AUTOSAR Layered Software Architecture*. URL: https://www.autosar.org/.

Bertsch, Christian et al. (2023). "Beyond FMI - Towards New Applications with Layered Standards". In: *15th International Modelica Conference*. URL: https://doi.org/10.3384/ecp204381.

Bertsch, Christian et al. (2024). *FMI, Layered Standards and ASAM Standards - Enabling Seamless SiL Simulation of Virtual ECU, Presentation at International ASAM Conference*. URL: https://fmi-standard.org/assets/literature/FMI-LS-ASAM_2024.pdf.

Blochwitz, Torsten et al. (2011). "The Functional Mockup Interface for Tool independent Exchange of Simulation". In: *8th International Modelica Conference*. URL: http://www.ep.liu.se/ecp/063/013/ecp11063013.pdf.

Blochwitz, Torsten et al. (2012). "Functional Mockup Interface 2.0: The Standard for Tool Independent Exchange of Simulation Models". In: *8th International Modelica Conference*. URL: http://dx.doi.org/10.3384/ecp12076173.

Coïc, Clément et al. (2021). "Linking Design Requirements to FMUs to create LOTAR compliant MBSE models". In: *14th International Modelica Conference*. URL: https://doi.org/10.3384/ecp2118149.

dSPACE (2024). *(R)evolution in the Exchange of V-ECUs*. URL: https://www.dspace.com/en/pub/home/news/engineers-insights/fmi-v-ecu-standard.cfm.

dSPACE (2025). *Virtual Validation Redefined*. URL: https://www.dspace.com/en/pub/home/news/engineers-insights/fmi-v-ecu-standard.cfm.

ISO (2011). *ISO 26262: Road vehicles – Functional safety*. Norm. URL: https://www.iso.org/obp/ui/#iso:std:iso:26262:-1:ed-2:v1:en.

Junghanns, Andreas et al. (2021). "The Functional Mock-up Interface 3.0 - New Features Enabling New Applications". In: *14th International Modelica Conference*. URL: https://doi.org/10.3384/ecp2118117.

Lawrenz, Wolfhard (2013-12). *CAN System Engineering*. Springer Science and Business Media.

MAP FMI (2025a). *FMI Layered Standard for network communication v1.0.0rc.1*. URL: https://github.com/modelica/fmi-ls-bus.

MAP FMI (2025b). *FMI LS BUS Implementers' Guide*. URL: https://modelica.github.io/fmi-guides/main/ls-bus-guide/.

MAP SSP (2024-03). *System Structure and Parameterization (SSP) Standard 2.0*. URL: https://ssp-standard.org/docs/2.0/.

Müller-Ettrich, Gunter (1998). "System Development with V-Model and UML". In: *The Unified Modeling Language*. Ed. by Martin Schader and Axel Korthaus. Heidelberg: Physica-Verlag HD, pp. 238–249.

Rausch, Mathias (2022). *Kommunikationssysteme im Automobil - LIN, CAN, CAN FD, CAN XL, FlexRay, Automotive Ethernet*. M: Carl Hanser Verlag GmbH Co KG. ISBN: 978-3-446-47457-4.

DOI
10.3384/ecp218693

An innovative heterogeneous modeling approach to build a cooling system for battery thermal management with common fluid properties involving FMI terminals

Karim Besbes

Siemens Digital Industry Software, France
karim.besbes@siemens.com

Abstract

In this paper, a new modeling approach combining native Simcenter Amesim submodels and Modelica submodels is presented. FMI terminals are used to enable the physical connection of fluids between Simcenter Amesim and Modelica. The innovation lies in the fluid properties that are computed in the causal and acausal worlds using the same technology. The architecture of a new Modelica AmeTpfMedia library developed to ensure continuity of fluid properties is presented, along with its accuracy and performance. Using this new approach, a demonstrator of a closed-loop heterogeneous cooling system for battery thermal management is built, opening the door to a new way of thinking complex multi-physics systems.
Keywords: Simcenter Amesim, Heterogeneous Modeling, FMI Terminals, Fluid Properties, Battery Thermal Management

1 Introduction

In multi-physics modeling, the choice between causal and acausal modeling paradigms, aligned with the needs of the modeling project, requires careful consideration.

The causal approach, commonly used in the Bond-Graph method (Hermien and Wiyatini 2021), consists of fixing the input and output status of variables of a model and each of its submodels, creating a causal relationship between inputs and outputs. This traditional approach makes it possible to build robust models. It allows precise control of system behavior, making it easier to interpret and understand system dynamics.

By contrast, acausal modeling, for which Modelica is particularly well suited, is based on equations (Fritzson 2014). In such a modeling paradigm, the only requirement is a well-balanced mathematical system defined by all declared equations, variables and data sets. Then, causality is naturally fixed by solving the mathematical system. This approach, more flexible, offers a convincing alternative when dealing with highly interconnected systems.

To meet the ever-increasing modeling challenges of tomorrow, with increasingly complex and interconnected systems, modeling techniques need to become ever more innovative and sophisticated. Combining causal and acausal paradigms, to build an heterogeneous system, can be one such innovative solutions.

Thanks to the Functional Mock-up Interface (FMI) standard and especially the 3.0 (Modelica Association 2022), such combined modeling approach is simplified, since it allows to regroup different signals in one connector. It then becomes conceivable to build an heterogeneous model involving fluids, as long as the mechanism used to calculate the thermodynamic properties in each submodel is the same.

This article aims to present an innovative heterogeneous modeling approach in Simcenter Amesim to build a cooling system for the battery of electrical vehicles. Simcenter Amesim, well known in the realm of system simulation, is natively causal and Bond-Graph based. It provides also a full-featured Modelica Editor including physical interfaces to connect both worlds "physically", with Terminals. To meet the challenge of ensuring perfect continuity of thermodynamic properties between Simcenter Amesim and Modelica's fluid libraries, a new Modelica media library, named AmeTpfMedia, has been developed. AmeTpfMedia takes advantage of the mechanism used in Simcenter Amesim to calculate fluid properties.

In the following article, the Modelica support in Simcenter Amesim is described as well as the definition of the physical interfaces. Then we will go over the AmeTpfMedia library developed to be fully consistent with Simcenter Amesim's Two Phase Flow library. Finally, the cooling system built as a proof-of-concept is presented and the simulation results are given.

2 Modelica support in Simcenter Amesim

The IMAGINE company, now Siemens Digital Industry Software has released its System Solution tool Simcenter Amesim back in 2012. This simulation platform gives access to a set of libraries of mechanical, electrical, hydraulic, pneumatic, fluid, and hybrid components created with the help of domain experts from both the industrial and the academic worlds. It enables the development of any new mechatronic product without having to write equations and offers numerous facilities for 1D system simulation (Vasiliu et al. 2018).

Among these facilities, Simcenter Amesim offers a full-

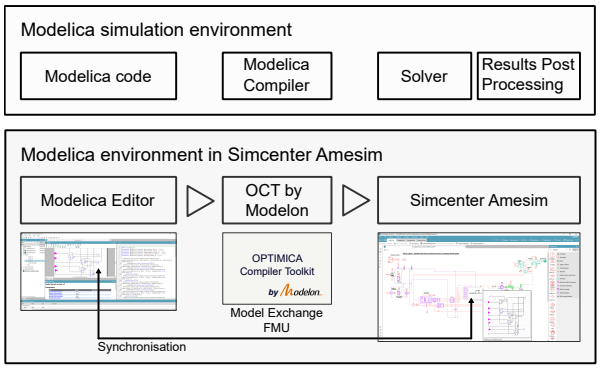


Figure 1. Modelica support and integration in Simcenter Amesim

featured editor dedicated to the Modelica language, as shown in the Figure 1. Since the version 17, the Modelica Editor embeds Modelon's OPTIMICA Compiler Toolkit (OCT), which is today compliant with the Modelica language specification 3.4 (Modelica Association 2017) and supporting both Modelica Standard Library (MSL) 3.2.3 as well as version 4.0.0, except Synchronous Language Elements and State Machines.

For the selected model in the Modelica Editor, the Modelica compiler generates a Model Exchange Functional Mock-up Units (FMU) version 2.0.4 extended with the FMI 3.0 terminals definition. It allows to define a group of variable to ease connecting compatible signals on the system level. The Model Exchange FMU is then treated as a native submodel by Simcenter Amesim, synchronized with the original Modelica code, removing pains that would be introduced due to cosimulation usage. This submodel can be connected to other native submodels and simulated with the internal solver. Thus, Simcenter Amesim can be used as a pure Modelica simulation environment made up by a Modelica Editor, a Modelica compiler, a solver and a bunch of post-processing features, or can be used to mix two modelling approaches to build a heterogeneous model.

3 Heterogeneous model with physical interfaces

In this article, we consider that a heterogeneous model is composed of at least two submodels built using different modeling tools or approaches.

In Simcenter Amesim, all the variables at the ports of the submodels have their causality fixed. Figure 2 illustrates four submodels taken from four different libraries. For example, the wall (c) has two ports where the heat flow rate is defined as an input and the temperature is defined as an output on both ports. Most of the time, several versions of the same submodel are available to modify causality. Such native submodels can only be connected to a Modelica submodel if three conditions are met. The first condition concerns the port variables at the interface, which must be defined as a group of variables. The second condition concerns the causality consistency and last one concerns unit consistency. The following sections describe how to deal with these conditions.

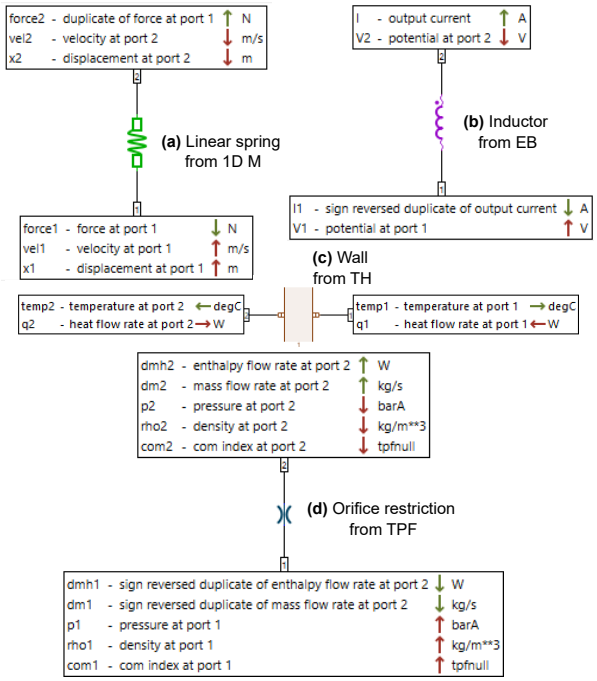


Figure 2. Example of Simcenter Amesim ports with fixed causality. The submodel (a) comes from the 1D Mechanical library, (b) from the Electrical Basics library, (c) from the Thermal library and (d) from the Two Phase Flow library

3.1 Terminals

In order to address new use cases, FMI 3.0 standard introduces the concept of layered standards adding optional features to the FMU, (Bertsch et al. 2023). It includes also an optional mechanism in the `icons/terminalsAndIcons.xml` to define the terminals and their graphical representation, as described in the section 2.4.9 of the specification. In Simcenter Amesim, the terminals definition has been backported to FMI 2.0.4 using the `/extra` directory. It is used to convert a Modelica connector into a physical Simcenter Amesim port.

Assuming that a Modelica connector is coded with terminals requirements, the compilation generates automatically an XML file where the root element is `fmiTerminalAndIcons` with *fmiversion="3.0"* attribute. The children elements are `GraphicalRepresentation` which is ignored in the current implementation and `Terminals` which is a sequence of `Terminal`. Each `Terminal` is described by a name, a mandatory matching rule (plug, bus or sequence) and an optional terminal kind used for the nominal typing. A `Terminal` is a sequence of terminal member variables, identified by their unique name among the list of scalar variables.

To illustrate a typical terminal definition, the Figure 3 shows an example of a XML tree from a compiled Mod-

DOI 10.3384/ecp218703

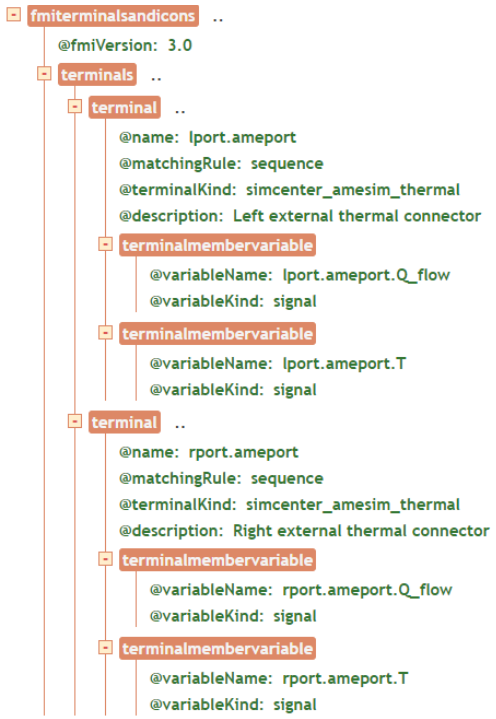


Figure 3. Example of a XML tree with terminals definition

elica submodel packaged into a FMU. The two terminals have a kind defined as `simcenter_amesim_terminals` and a **sequence** matching rule attribute. In Simcenter Amesim, a thermal port is declared with a group of two variables, heat flow rate and temperature, as shown in Figure 2(c). Given that `lport.ameport.Q_flow` and `lport.ameport.T` define a group of scalar variable already existing in Simcenter Amesim with the same causality, the corresponding port is automatically created. Figure 4 shows the external view of the corresponding submodel imported in Simcenter Amesim. Note that the two signal ports `TempOutput1` and `TempOutput2` are also created. But they refer to two other scalar variables with output causality that are not part of a terminal.

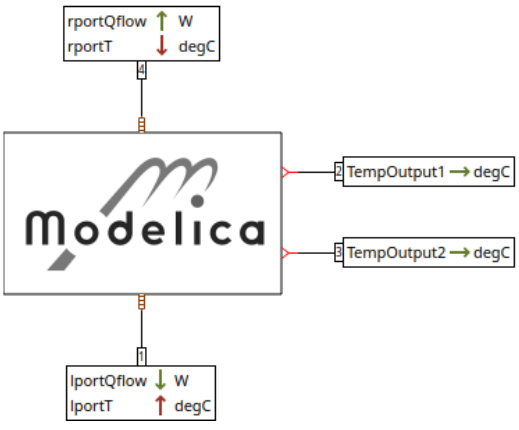


Figure 4. Example of a compiled and imported Modelica submodel using the terminals described in the XML tree

From Modelica point of view, in order to generate in the `modelDescription.xml` a terminal for a group of variables, Modelica compiler requires the following tool vendor annotation:

```
annotation(__simcenter(port(port_type="port
    type name")));
```

In Simcenter Amesim, there are 13 different port types classified per physical domain, listed in the Table 1.

Table 1. List of native port types in Simcenter Amesim

Domain	*Native Port Type*
Mechanical	lshaft, rshaft, rope, mecha 2D
Fluid	hflow, pflow, geocom
Thermal	thermal
Magnetic	magnet
Electrical	elect, elect3p
Chemical	electroch
Signal	signal

3.2 Connectors

The equivalent of a Modelica connector in Simcenter Amesim is a port. From the moment a port is targeted, the rule to build the associated Modelica connector is always the same. The order of the declared variables in the connector must be exactly the same as the order of variables defined in the corresponding port. The same applies to variable causalities and types. And the port type name listed in the Table 1 must be specified in the tool vendor annotation. The following code uses a Modelica connector to generate the thermal port, shown in Figure 4:

```
connector PortTinQout
  output Q_flow(unit="W") "Output heat flow
      ";
  input Real T(unit="degC") "Input
      temperature";
annotation (__Modelon(isTopLevel=true),
    __simcenter(port(port_type="thermal")));
end PortTinQout;
```

As can be seen, order, causalities and units have been set to be consistent with the thermal port targeted as well as the port type name. Knowing these rules, a Modelica submodel can be coded to connect to any of Simcenter Amesim's native physical ports.

3.3 Interfaces

In this context, a Modelica interface is defined as a class where connectors from both worlds are already declared. It provides end-user capability to connect a Modelica library to a Simcenter Amesim's library for the same physical domain. In the Modelica Editor, in addition to the MSL, the library Simcenter Amesim Modelica Library (SAML) is provided. It is a standardized library based on the native submodels including several physical interfaces as illustrated in the Figure 5.

For most used physical domains and associated causalities there is a dedicated interface in the SAML. From

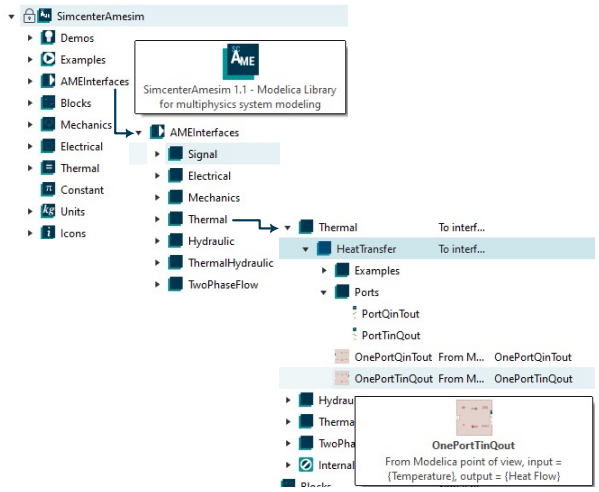


Figure 5. Simcenter Amesim Modelica Library 1.1 tree with the thermal connectors and interfaces

Modelica point of view, the MSL is the reference library of connector definition. For instance, the thermal interfaces in the SAML are built in order to connect any Modelica submodels containing a `HeatPort` connector from the MSL to any Simcenter Amesim submodels coming from the thermal library. In the MSL, the `HeatPort` connector contains the variables T and Q. From Simcenter Amesim point of view, thermal port contains also T and Q but with fixed opposite causalities. Then two causalities and two interfaces can be defined. Figure 6 shows both of them allowing to convert a convection submodel from the MSL dedicated to Simcenter Amesim. As illustrated, the interfaces contain two different connectors, one for Simcenter Amesim and the other to be connected to the convection submodel. Within the interfaces, declared connectors ensure causality consistencies for each variable, and equations ensure units consistency between the two environments.

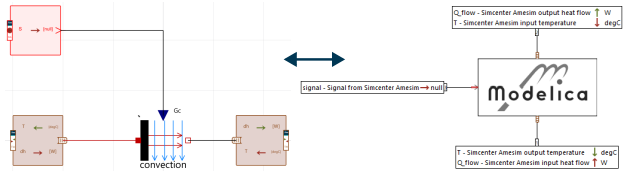


Figure 6. Usage of thermal interfaces to convert a Modelica convection submodel for Simcenter Amesim with the thermal ports

Note that thermal connectors in Figure 6 are not connected to anything, and the compilation is still proceeding correctly. This is due to a second tool vendor annotation telling that connector must be treated as a top-level connector, regardless of the layer in which it has been declared in the model. It allows to simplify the workflow for end-user. A simple drag and drop of the interface into the diagram and a connection with the Modelica submodel is sufficient. The following code is for the interface with heat flow rate as input and temperature as output:

```
model OnePortQinTout "From Modelica point
    of view, input = {Heat Flow}, output =
    {Temperature}"
  SimcenterAmesim.Thermal.Interfaces.
      HeatPort port;
  Ports.PortQinTout ameport;
equation
  port.T = ameport.T + SimcenterAmesim.
      Constant.celsius2kelvin;
  port.Q_flow = -ameport.Q_flow;
end OnePortQinTout;
```

And the thermal connector `ameport` including the two tool vendor annotations, one to treat the connector as a top-level connector and the second for the terminal definition, is coded as follows:

```
connector PortQinTout
  output Real T(unit="degC") "Simcenter
      Amesim output temperature";
  input Units.AMESIUnits.HeatFlowRate
      Q_flow "Simcenter Amesim input heat
      flow";
annotation (__Modelon(isTopLevel=true),
    __simcenter(port(port_type="thermal")))
    ;
end PortQinTout;
```

In this section, terminals, connectors and interfaces have been introduced, showing how to build a thermal interface as an example. The same strategy can be applied for other physical fields like mechanical or electrical and for other connectors than those from the MSL. However, as soon as a transport of a fluid is involved, the interface must also ensure mass and energy balance between both worlds. Such interface is described in the last section of this article where the demonstrator is presented. Finally, in this context, where Simcenter Amesim is the integration and simulation platform, the Modelica part must use the same mechanism than Simcenter Amesim to calculate the fluid properties to avoid physical and numerical discontinuities. This mechanism is introduced with AmeTpfMedia.

4 AmeTpfMedia library

The AmeTpfMedia library has been developed in Modelica for calculation of fluid properties. It uses the same utilities than the Two-Phase Flow (TPF) library and developed for dynamic simulation and design of thermodynamic applications, such as automotive air-conditioning, heat pumps, Rankine cycles... These utilities have been recompiled specifically for AmeTpfMedia and allow to calculate the thermodynamic state properties and their partial derivatives for a set of refrigerants. The description quality of fluid properties depending on the type of equation of state (EoS) used to do the calculation can have a significant impact on the results, performance and robustness of the simulation. With AmeTpfMedia, when a refrigerant is selected, it is possible to find the best compromise for simulation requirements, providing the choice between

DOI
10.3384/ecp218703

Helmholtz, MBWR or Peng-Robinson EoS. Most refrigerants have at least two EoS choices including Helmholtz, and a small proportion have only one. The list of fluids available is presented in the Table 2.

4.1 Architecture

The structure of the library is illustrated in Figure 7. It contains the package AmeTpfMedium extending from PartialTwoPhaseMedium defined in the MSL media package. This strong choice of architecture is motivated by the number of well known libraries in the Modelica community already using the MSL media structure to call the function of fluids, like Buildings (Wetter et al. 2014) or ThermoCycle (Dechesne et al. 2014).

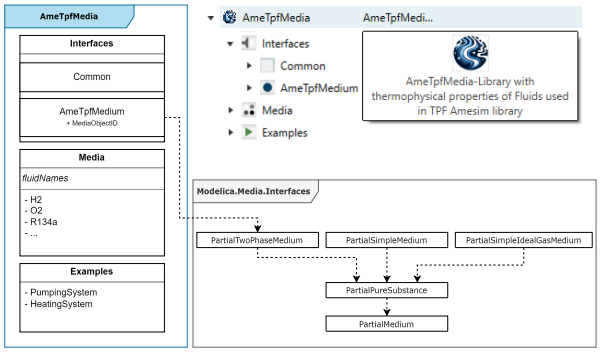


Figure 7. Structure of the AmeTpfMedia library

Inside the AmeTpfMedium package, the class MediaObjectID defines an ExternalObject instantiating the medium, set by an index of fluid named ID, as follows:

```
class MediaObjectID
  extends ExternalObject;
  function constructor
    input Integer ID;
    output MediaObjectID media;
external"C" media =
    create_modelica_media_as_predefined_fluid
    (ID)
    annotation (Include="#include <
        media_public.h>", Library={"
        AmeTpfMedia"});
  end constructor;
  function destructor
    input MediaObjectID media;
external"C" free_modelica_media(media)
    annotation (Include="#include <
    media_public.h>",
         Library={"AmeTpfMedia"});
  end destructor;
end MediaObjectID;
```

The ExternalObject is used as a pointer by all thermodynamic functions contained in the Internal package in order to call directly the compiled resources (headers and compiled objects). For instance, the function calculating the specific enthalpy according to the density and the temperature is coded as follows:

```
function h_dT
  input Density d;
  input Temperature T;
  input MediaObjectID media=mediaID;
  input Integer EoS=EoSInt;
  output SpecificEnthalpy h;
external"C" h =
    modelica_media_compute_h_from_rho_t(
    media,EoSInt,d,T)
    annotation (Include="#include <
        media_properties_from_rho_t_public.
        h>", Library{"AmeTpfMedia"});
end h_dT;
```

These internal functions allow to rebuild one by one the inherited functions from the MSL. The derivatives required for the inversions of temperature and density functions are implemented in Modelica as well as the time derivatives of the enthalpy, density, pressure and entropy to facilitate the numerical inversion.

4.2 Validation

The AmeTpfMedia library has been tested with a set of standard functions calculating the pressure and the specific enthalpy over a range of density and temperature. The utilities used by the AmeTpfMedia library have already been validated in the context of Simcenter Amesim. Nevertheless, to ensure the good integration in the Modelica world, a protocol of test has been defined for three very common fluids, which are Water, CO2, and R134a.

The protocol consists in comparing the results with reputed Modelica libraries of fluid properties: the MSL, the CoolProp library (Casella 2008) and the non-profit version of TilMedia from Clara library (XRG-Simulation GmbH TLK-Thermo GmbH 2023). Figure 8, 9 and 10 compare the results generated from AmeTpfMedia and those from CoolProp, TilMedia and/or MSL. These are results obtained in the form of $Y(T,\rho)$, where Y is the variable of interest, pressure p or specific enthalpy h, evaluated in the domain defined by framing the critical point of each fluid, with n the number of calculated points. To perceive the variations between the libraries, we define the average relative deviation *AvDev* between AmeTpfMedia and the other libraries as follows, where lib can be CoolProp, TilMedia or MSL:

$$AvDev_{lib}(Y) = \sum_{n}(Y_{AmeTpfMedia}/Y_{lib} - 1) * 100/n \quad (1)$$

Regarding the thermodynamic properties of the water calculated by AmeTpfMedia, Figure 8 shows a comparison with the three other libraries. For p and h, there is an average relative difference of 0.5% between AmeTpfMedia and the other libraries. This small difference can be explained by the fact that Simcenter Amesim uses a different standard for the thermodynamic formulation of water.

Regarding the CO2, Figure 9 shows an average relative deviation between AmeTpfMedia and CoolProp of 0.01% and between AmeTpfMedia and TilMedia of 1e-6%. Both are negligible. Note that the non-profit version

Table 2. List of fluids available in AmeTpfMedia

Family	*Fluid usage name*
Common fluids	pH2, H2, oH2, N2, O2, CO2, H2O, H2O2, NH3, Air
Refrigerants	R22, R32, R123, R125, R134a, R13b1, R13i1, R143a, R152a, R227ea R236ea, R236fa, R245fa, R1123, R1224yd(Z), R1234yf, R1234ze(E) R1243zf, R1336mzz(Z), Novec 649, Novec 7000, R404a, R407c R410a, R450a, R452b, R454c, R455a, R507a
Organic compounds	Ethylene, Methane, Ethane, Propane, Butane, Iso-Butane, Pentane Iso-Pentane, Cyclopentane, Heptane, Octanen, Iso-octane Dodecane, Methanol, Ethanol
Noble gas	He, Ne, Ar, Kr, Xe

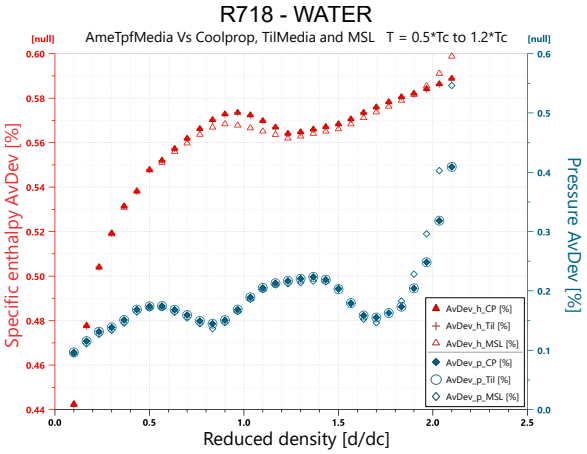


Figure 8. Comparison between AmeTpfMedia, Coolprop, TilMedia and the MSL for the Water

of TilMedia gives only an access to GERG CO2 built using the GERG EoS (Kunz and Wagner 2012) instead of Helmholtz, which is used by CoolProp and AmeTpfMedia. And yet, the biggest difference lies between AmeTpfMedia and Coolprop.

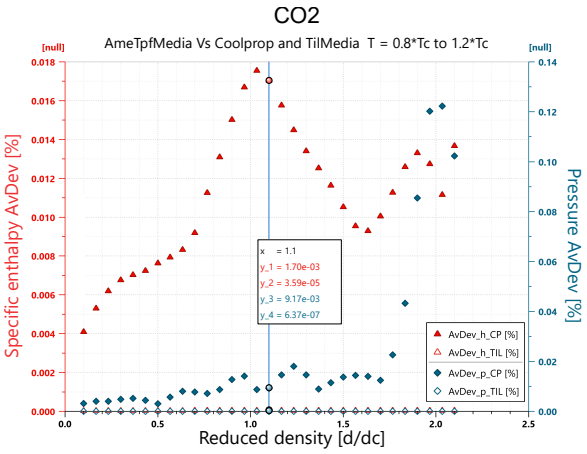


Figure 9. Comparison between AmeTpfMedia, Coolprop and TilMedia for the CO2

Finally, regarding the R134a, Figure 10 shows an average relative deviation of 5e-6% for h and of 1.5e-4% for p, between AmeTpfMedia and CoolProp or the MSL. As a result, the R134a is the fluid with the greatest similarity between the libraries.

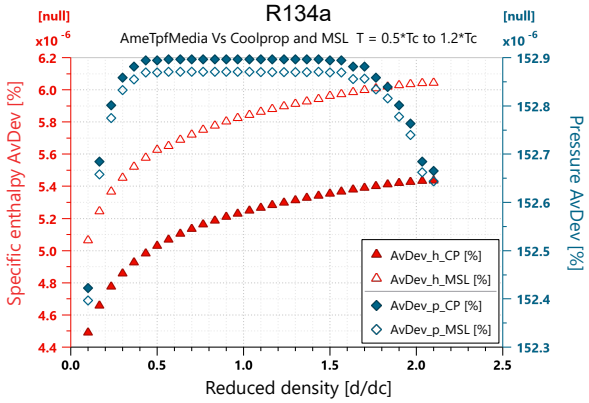


Figure 10. Comparison between AmeTpfMedia, Coolprop and the MSL for the R134a

In this section, thermodynamic properties calculated for three different fluids have been presented. Another article dealing specifically with TPF utilities in Simcenter Amesim would provide more results for more fluids, and address the slight differences observed. Nevertheless, it is considered that AmeTpfMedia reflects the expected behavior and the transition of TPF utilities to Modelica is validated.

4.3 Performance and Robustness

Performance and robustness are key when it comes to accurately calculating fluid properties for thermodynamic cycle simulation. This section introduces a benchmark consisting in simulating on the same environment and the Simcenter Amesim's solver an evaporator model taken from the ThermoCycle library. Water and carbon dioxide are chosen as refrigerants and CPU times of each library are compared. The aim is not to make a general statement on which library is the best performing. This would require a much larger number of tests with a wide selection of models and fluids. This is a qualitative reference to see

DOI 10.3384/ecp218703

if AmeTpfMedia performs comparably to other reputable libraries.

The model used for the test bench coming from ThermoCycle is named `Hx1DConst`. This is a discretized counter-current heat exchanger in which one of the two fluid has a constant heat capacity. The other fluid (Water or carbon dioxide) is described at each cell of the tube, where enthalpy, density, pressure and vapor quality is computed. The heat exchanger is discretized into 30 cells and the start values chosen are exactly the same for a selected fluid. The boundary conditions are defined in order to evaporate the selected fluid.

With regard to the water, as can be seen in Figure 11, CPU times of libraries using external objects for calculation are fairly similar, with AmeTpfMedia slightly ahead. As expected, the MSL gives the best CPU time. Indeed, as described in the MSL documentation, the water was explicitly designed to work well in dynamic simulations with a high focus on the performance. In the second graph, a comparison of the cooling power can be found. The results show a relative difference of 0.2% between AmeTpfMedia and the other libraries, which is consistent with the deviation observed in Figure 8.

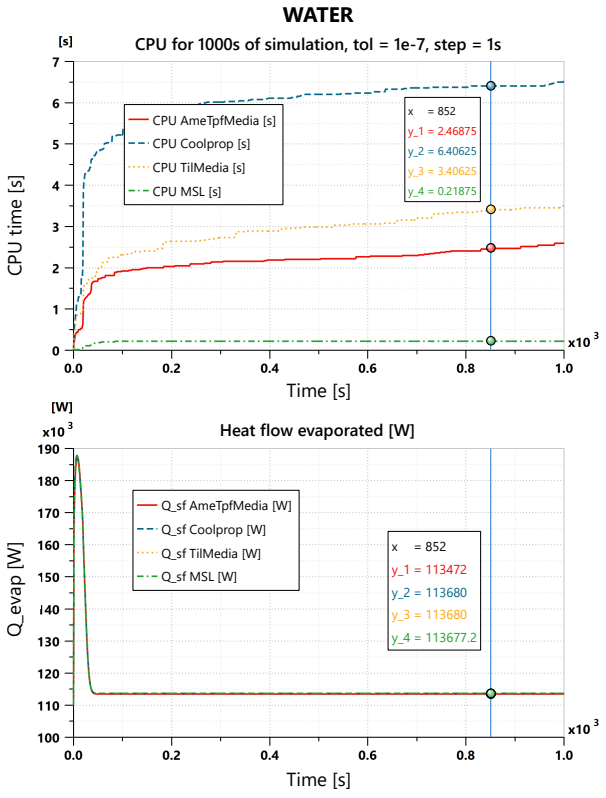


Figure 11. Comparison of CPU time and cooling power with water as working fluid

For carbon dioxide, as shown in Figure 11, AmeTpfMedia and TilMedia have similar CPU times, while CoolProp is 3 to 5 times slower. TilMedia is slightly ahead probably due to the GERG EoS used for the CO2. As for the cooling power, the result shows negligible differences.

During the benchmark, no significant difference was observed concerning the robustness. The libraries behaved in the same way at each simulation.

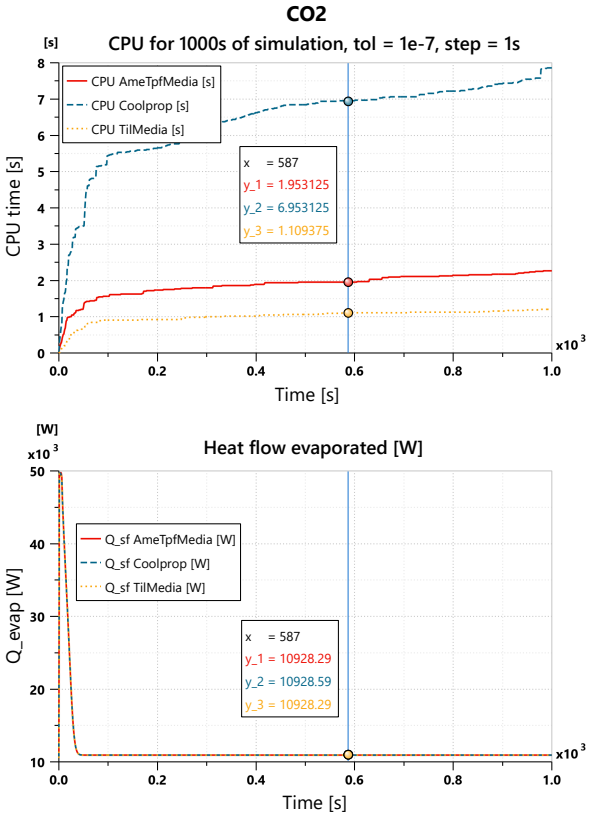


Figure 12. Comparison of CPU time and cooling power with carbon dioxide as working fluid

5 Use case: heterogeneous cooling system

Imagine two teams tasked with modeling a complex system like battery thermal management. One team employs Simcenter Amesim, while the other develops in-house submodels using the Modelica language. Nowadays, this scenario is becoming increasingly common, particularly among large industrial groups with multiple R&D departments. The following demonstrator aims to answer this need.

5.1 Description

The first team has developed the complete system in Simcenter Amesim, with the exception of the compressor, condenser and valve, which were developed by the Modelica team. Thanks to the Terminals and AmeTpfMedia, it becomes possible to integrate each development made by each team to build the full system presented in the Figure 13. As can be seen, the system consists of the charger, the battery in purple, the coolant system in orange, the refrigerant system in blue, part of which is made in Modelica, and the control in red.

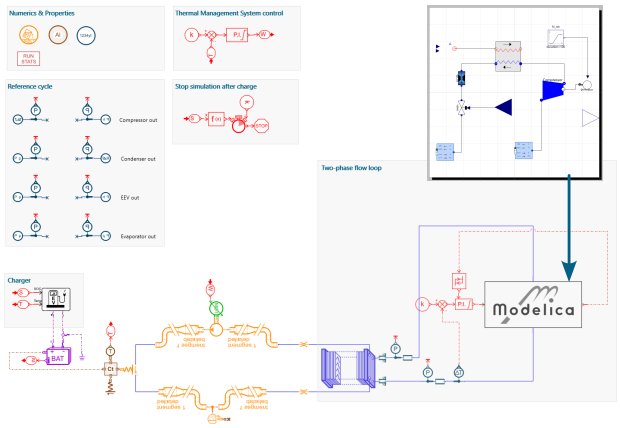


Figure 13. Heterogeneous chiller loop design for battery thermal management

For this use case, the focus is on the design of the cooling loop, composed of an evaporator from Simcenter Amesim and a compressor, condenser and expansion valve from Modelica's ThermoCycle library. The evaporator is a discretized brazed plate heat exchanger with 5 cells. The condenser from ThermoCycle is a discretized heat exchanger with 6 cells. The required cooling power is 7 kW. The evaporation temperature is set at 5°C with a superheat of 5°C. The condensation temperature is set at 55°C with a subcooling target of 7.5°C. Finally, the working fluid is R1234yf.

5.2 TPF physical interface

At evaporator boundaries, the causalities for both ports are defined as shown in the Figure 14(a). These boundaries impose the causality to the Modelica submodel. The Figure 14(b) is a representation of the Modelica TPF interface where pressure and density are given by Simcenter Amesim and specific enthalpy and mass flow are given by Modelica. The Modelica part built with ThermoCycle's submodels use the MSL connectors for the fluid. Thus, the TPF interface must be defined with a MSL fluid connector and a TPF connector.

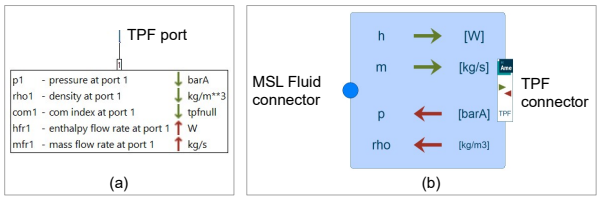


Figure 14. TPF port definition with a fixed causality. (a) Represents external TPF ports and (b) represents the corresponding TPF interface in the Modelica world

The TPF connector is coded as follows:

```
connector Portpinmout "p,rho in / m,Q
    out"
output SimcenterAmesim.Units.AMESIUnits.
    HeatFlowRate Q_flow
  "Enthalpy flow rate";
output SimcenterAmesim.Units.AMESIUnits.
    MassFlowRate m_flow "Mass flow rate";
input AmeTpfMedia.AMEUnits.
    AbsolutePressure_bar p "Pressure";
input SimcenterAmesim.Units.AMESIUnits.
    Density rho "Density";
input Real ComId(unit="tpfnull") "Com
    index";
annotation (
  __Modelon(isTopLevel=true),
  __simcenter(port(port_type="hflow")));
end Portpinmout;
```

The causalities are fixed in the good order, as well as the units and the annotations with the port type name related to the fluid domain. Note that the ComId is a communication variable specifically used in Simcenter Amesim context. This variable avoids redundant calculations and evaluates the total volume of a system by counting the number of capacitive (volume). In this demonstrator, the Modelica part receives this variable and therefore does not interfere with Simcenter Amesim.

The interface between the Modelica fluid connector and the TPF connector is coded as follows:

```
model OnePortpinmout "(R) TPF Interface"
  // Medium package
  replaceable package Medium = AmeTpfMedia.
      Media.partialMedium "Medium model";
    "Mass flow rate";
  // Connectors
  Modelica.Fluid.Interfaces.FluidPort_a
      fluidport(
    redeclare package Medium = Medium,
    C_outflow=fill(1/Medium.nC, Medium.nC),
    Xi_outflow=fill(1/Medium.nX, Medium.nX
        - 1));
  SimcenterAmesim.TPF.Interfaces.Ports.
      Portpinmout ameport(
    p(start=1),
    rho(start=500));
  // Medium state
  Medium.ThermodynamicState state=Medium.
      setState_pdX(ameport.p*1e5, ameport.
      rho);
equation
  fluidport.p = ameport.p*1e5;
  ameport.m_flow - fluidport.m_flow = 0;
  if ameport.m_flow >= 0 then
    ameport.Q_flow = ameport.m_flow*
        inStream(fluidport.h_outflow);
  else
    ameport.Q_flow = ameport.m_flow*state.h
        ;
  end if;
  state.h = fluidport.h_outflow;
  fluidport.Xi_outflow = inStream(fluidport
      .Xi_outflow);
  fluidport.C_outflow = inStream(fluidport.
      C_outflow);
end OnePortpinmout;
```

This TPF interface is made up of a replaceable package for defining the fluid, the required connectors, and a pressure- and density-dependent state functions (inputs from Simcenter Amesim). It contains also the equations

ensuring the unit consistency, the mass and energy balance and the sign consistency taking into account the Modelica `flow` and `stream` definition.

With such interface, the end-user only needs to drag and drop this class to the diagram, define the fluid and connect it to the the Modelica models as shown in the Figure 15.

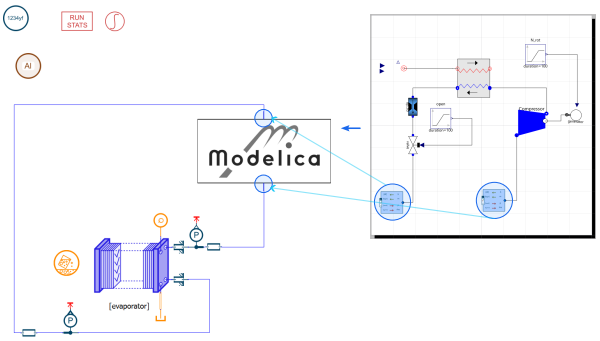


Figure 15. Heterogeneous cooling system with TPF evaporator, TPF Modelica interfaces and ThermoCycle submodels. The R1234yf is set in both worlds

5.3 Results

All submodels have been designed step by step to match the pre-design cycle calculated on the basis of the requirements listed in the descriptive section. The heterogeneous model was compared with the original homogeneous model built using only Simcenter Amesim's native submodels. Simulation time is 1000s with a tolerance of 1e-7 for both models. As shown in Figure 16, some differences in dynamics can be observed, as well as in the starting points. This is acceptable given that the compressor, the condenser, and the expansion valve found in the Modelica ThermoCycle library are not defined and built like the Simcenter Amesim's submodels replaced in this use case. Looking at the steady state, the cooling power of the evaporator sticks well to the requirement of 7 kW. The mass flow rate, pressures, and specific enthalpies are very similar in both models and are also aligned with the requirements. These observations validate the proof-of-concept that combines two modeling paradigms to build a closed-loop cooling system with its working fluid.

6 Discussion and Outlooks

In this use case, we intentionally selected the causality in order to obtain the ComId variable from Simcenter Amesim, rather than the other way around. This decision was made because our current OCT integration lacks internal communication to ensure a consistent value of the ComId from the Modelica's perspective. However, we can propose a workaround by introducing a small volume, such as a pipe, in the Simcenter Amesim component before the connection. This will reverse the causality and allow us to achieve the same conditions as demonstrated.

Furthermore, it's important to note that the connections between Modelica and Simcenter Amesim in our current

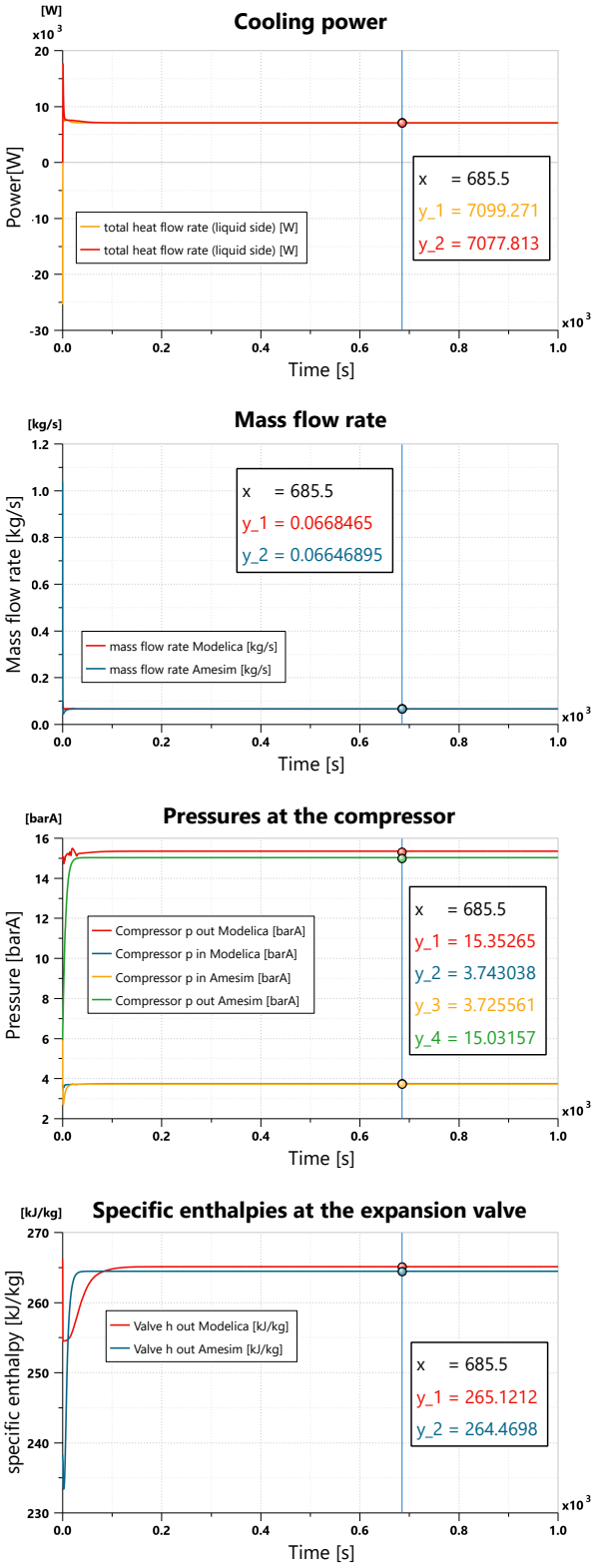


Figure 16. Comparison of simulation results between a homogeneous and a heterogeneous cooling system

demonstration are purely physical. Unfortunately, there is no capability to transmit information from Simcenter Amesim to an inner class in the Modelica world. There is currently no possibility to exchange the volume of the

system or the mass of the refrigerant between the two worlds, which could potentially impact the initialization process or certain internal variables, such as total pressure loss. To address this limitation, a workaround is to manually incorporate the correct mass, volume, or pressure loss values, calculated or fixed, in each world. However, this workaround assumes that the selected or developed Modelica library takes these factors into consideration. As a suggestion for future work, it would be beneficial to explore methods of communicating information outside the FMI standard to the compiler.

Another important milestone that has been identified pertains to AmeTpfMedia. Currently, the library only includes methods for calculating properties of a single fluid. However, TPF utilities already have the capability to calculate properties of fluid mixtures. Therefore, the next step would involve adding functionalities specifically dedicated to fluid mixtures, allowing for configuration of the number of fluids and their respective mass fractions.

7 Conclusions

This paper presented how the FMI terminals and the library AmeTpfMedia offer a new way to build heterogeneous closed loop fluid systems in Simcenter Amesim. This new solution meets needs coming from the industry where Simcenter Amesim and Modelica are most often used in separate departments. It can

A series of test cases demonstrated that AmeTpfMedia is well suited to the simulation of fluid systems, thanks to its accuracy and performance. Its architecture inherited from the MSL saves a lot of time when it comes to use open-source libraries also compatible with the MSL.

Finally, by following the steps described in the article, it becomes possible to recreate manually a Modelica physical interface to connect Simcenter Amesim to an in-house connector.

References

Bertsch, Christian et al. (2023-12). "Beyond FMI - Towards New Applications with Layered Standards". In: *Modelica Conferences* 204, pp. 381–388. ISSN: 1650-3740. DOI: 10.3384/ECP204381. URL: https://ecp.ep.liu.se/index.php/modelica/article/view/947.

Casella, Francesco (2008-01). *ExternalMedia: a Library for Easy Re-Use of External Fluid Property Code in Modelica*. URL: http://coolprop.org/.

Dechesne, Bertrand et al. (2014). *ThermoCycle 2.3 Library*. University of Liege Thermodynamic laboratory - Belgium. URL: https://thermocycle-docs.readthedocs.io/en/latest/index.html.

Fritzson, Peter (2014-11). *Principles of Object Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach*. Vol. 9781118859124. Wiley-IEEE Press, pp. 1–1223. ISBN: 9781118989166. DOI: 10.1002/9781118989166. URL: https://onlinelibrary.wiley.com/doi/book/10.1002/9781118989166.

Hermien, Nugraheni and Tri Wiyatini (2021-10). *Engineering Systems Dynamics, Modelling, Simulation, and Design*. British Columbia Institute of Technology, p. 252. URL: https://open.umn.edu/opentextbooks/textbooks/1603.

Kunz, O. and W. Wagner (2012-11). "The GERG-2008 wide-range equation of state for natural gases and other mixtures: An expansion of GERG-2004". In: *Journal of Chemical and Engineering Data* 57 (11), pp. 3032–3091. ISSN: 00219568. DOI: 10.1021/JE300655B/ASSET/IMAGES/MEDIUM/JE-2012-00655B_0029.GIF. URL: https://pubs.acs.org/doi/abs/10.1021/je300655b.

Modelica Association (2017-04). *Modelica – A Unified Object-Oriented Language for Systems Modeling. Language Specification Version 3.4*. Tech. rep. Linköping: Modelica Association. URL: https://www.modelica.org/documents/ModelicaSpec34.pdf.

Modelica Association (2022-04). *Functional Mock-up Interface for Model Exchange and Co-Simulation Version 3.0.0*. Tech. rep. Linköping: Modelica Association. URL: https://fmi-standard.org.

Vasiliu, Nicolae et al. (2018-04). *Simulation of Fluid Power Systems with Simcenter Amesim*. CRC Press. ISBN: 9781315118888. DOI: 10.1201/9781315118888. URL: https://www.taylorfrancis.com/books/mono/10.1201/9781315118888/simulation-fluid-power-systems-simcenter-amesim-nicolae-vasiliu-daniela-vasiliu-constantin-c%C4%83linoiu-radu-puhalschi.

Wetter, Michael et al. (2014-07). "Modelica Buildings library". In: *Journal of Building Performance Simulation* 7 (4), pp. 253–270. ISSN: 19401493. DOI: 10.1080/19401493.2013.765506.

XRG-Simulation GmbH TLK-Thermo GmbH, and (2023). *ClaRa 1.8.1*. URL: https://www.claralib.com/.

Enhancing Large-Scale Power Systems Simulations through Functional Mock-up Unit-based Grid-Forming Inverter Models

Sagnik Basumallik[1] Luigi Vanfretti[2] Mohammad Ali Dashtaki[3] Ziang Zhang[3] Reza Pourramezan[1] Hossein Hooshyar[1]

[1]New York Power Authority, {sagnik.basumallik, reza.pourramezan, hossein.hooshyar}@nypa.gov
[2]Rensselaer Polytechnic Institute, vanfrl@rpi.edu
[3]Binghamton University, {mdashta1,zhangzia}@binghamton.edu

Abstract

New York State (NYS) faces significant challenges in meeting the Climate Act's bold goals of 70% renewable energy generation by 2030 and total decarbonization of the electric grid by 2040. Extensive simulations are required to assess the impact of numerous inverter-based resources (IBRs) deployed to the large-scale NYS power grid, aiming to evaluate their dynamic behavior and mitigate any negative interactions with their control schemes. However, the modeling efforts required are huge and the computational burden of large-scale simulations is extensive, and often limited by the capabilities of domain-specific tools. This work addresses these limitations by developing a Functional Mock-up Unit (FMU) of Grid-Forming (GFM) Inverters for IBR control and integrating them with an electromechanical phasor-domain power system solver. The proposed FMU facilitates the simulation and parametric studies needed to analyze large-scale IBR usage with significantly improved manual modeling and computational efforts. The paper details the process of developing and FMU model for GFM IBRs, including all relevant control loops implemented in the Modelica language and FMU integrated in OPAL-RT's ePHASORSIM^{TM} software. Our FMU models are used to successfully deploy and study the impacts of up to $6,200+$ MVA from IBRs on the 5000-bus NYS transmission system.

Keywords: Power Grid, Power Systems, Droop-based Control, Grid Forming Inverter, Inverter-based Resource, Modelica, Functional Mock-up Unit

1 Introduction

In accordance with the NYS Climate Leadership and Community Protection Act, 70% of New York State's electricity must be derived from renewable energy systems (RES) by 2030 (Senate 2023). Grid-forming (GFM) inverter-based technology stands out as a promising solution to integrate RES that are interfaced with the NYS grid via power electronics-based inverters, including wind, PV solar, and fuel cells. This is due to the ability of GFM control to independently provide voltage and frequency support to the grid with fault ride-through capabilities (Du et al. 2019). Comprehensive simulation studies are essential to evaluate the impacts of IBRs on the future NYS grid. However, modeling and simulating a large power system with GFM IBRs deployed at scale introduces substantial challenges including modeling efforts, interfacing resources, and computational requirements.

To address these challenges, this paper presents the development of a complete IBR model with a droop-based GFM control as a Functional Mock-up Unit (FMU). An FMU is a prepackaged, standardized container of user-defined models that adheres to the Functional Mock-up Interface (FMI) standard. The FMI standard is a free tool-independent standard that defines *"a container and an interface to exchange dynamic simulation models using a combination of* `.xml` *files, binaries and C code" (Modelica Association 2025).*

The main advantage of developing IBR models in Modelica is that the models can be re-utilized in different simulation platforms by exporting them as an FMU. This provides end-users with full control over model parameters and IBR control modes, making their reuse in simulation highly scalable and efficient. In addition, FMUs enhances interoperability between different simulation tools (Luigi Vanfretti, Li, et al. 2013). Common industry tools such as PSS/E, PSCAD, EMTP and PowerFactory have different ways of defining models and data, making it difficult to share them between tools (Laera et al. 2022). In contrast, using FMUs offers a *"build once, deploy anywhere"* advantage, allowing models to be easily shared and reused across different simulation platforms without the need for extensive reconfiguration. This minimizes dependencies to the C compiler and the Modelica tool (Luigi Vanfretti, Laughman, and Chakrabarty 2024). Some of the power system simulation tools that currently support FMUs include ePHASORSIM, EMTP, Pandapower, and Powerfactory. Users can also create custom models and interface them as FMUs when existing native software libraries do not offer the required models. Flexibility is key when performing large-scale power system simulation and assessing the impact of integrating IBRs, as required in NYS grid studies, and FMUs offer such advantage.

The FMU developed for the GFM IBR used in this paper is implemented in the Modelica language. Modelica is

an open-access standard and a free, object-oriented mathematical modeling language that is generally used to describe large-scale physical systems (Fritzson and Engelson 1998) and build control applications that interact with their environment (Thiele et al. 2017; Hellerer, Bellmann, and Schlegel 2014). Modelica supports component-based modeling and allows one to mathematically describe a model using differential, algebraic, and discrete equations. In addition, the Modelica Standard Library provides a plethora of components that can be reused, for example, as done in this paper to build several control loops of the GFM IBR. Using object-oriented modeling constructs, such as inheritance (Fachini, Bhattacharjee, et al. 2023), the GFM inverter model is created hierarchically. Beginning with a simple voltage source, we progressively build a controllable voltage source that is the fundamental backbone of the GFM inverter model. The generated FMU is set up to be independently tested and verified inside a Modelica-compliant tool, Dymola (Brück et al. 2002), prior to external deployment. This is one of the main advantages of using the Modelica language and Dymola, which allows to test each of the individual component of the model in isolation, a feature currently not supported by power system simulation software (Laera et al. 2022).

1.1 Literature Review

The Modelica language has been widely used for various power system applications. One notable development is the Open Instance Power System Library (OpenIPSL) (L. Vanfretti et al. 2016; Baudette et al. 2018; de Castro et al. 2023) which has power system component models written in the Modelica language for power system dynamic studies, such as phasor time-domain simulations, while allowing one to extend the modeling scope of conventional power system simulators. The authors in (Luigi Vanfretti, Mukherjee, et al. 2019; Gomez et al. 2018) develop an automatic re-synchronization controller for islanded networks within a multi-domain gas turbine and power system model using Modelica. Specifically, they used Modelica libraries such as ThermoPower (Casella and Leva 2005) to model the thermomechanical dynamics of the gas turbine and OpenIPSL for the components of the power system. The authors in (Mukherjee and Luigi Vanfretti 2018) implemented a frequency controller in Modelica for island operation in power distribution networks. This work integrated multiple Modelica blocks that include a synchronous generator, a gas turbine model, and an excitation system, Modelica noise library for stochastic load modeling and zero-order hold, and `fixedDelay` blocks for PMU reporting rates and delays.

Modelica breaks through the limits of conventional power system simulators. It allows one to conduct specialized studies that usually need various tools or separate programs for different analyses. This is beneficial, for example, in studies related to the stability of island power systems, such as those presented in (Winkler 2018). Here, different models of an island power are connected through a single transmission line to the Icelandic power grid, and their stability is analyzed. The entire transmission and generator models were built in Modelica using the OpenIPSL library, and multiple scenarios are studied for cases where connection to the national grid is lost. Another example in (Segerstrom et al. 2023) develops a Modelica-based approach to analyze subsynchronous oscillations for a lumped mass torsional shaft model with 25 masses. The Modelica implementation was shown to have two unique advantages. In addition to including shaft torsional dynamics, turbine, boiler, and governor dynamics were taken into account using components from OpenIPSL. In addition, the authors emphasize that the `Modelica_LinearSystems2` library (Baur, Otter, and Thiele 2009) allows the analysis of modal-based eigenvalues to be performed directly, eliminating the need to develop a separate model for linear analysis or the use of additional tools for this purpose (Nikolaev et al. 2020).

Microgrid modeling and simulation often exclude phasor simulations, yet they offer benefits by being computationally efficient, as they simplify the switching behavior of power electronics components. For example, the authors of (Fachini, Bhattacharjee, et al. 2023; Fachini, Pigott, et al. 2023; Fachini, Bogodorova, et al. 2024) have implemented a microgrid model using Modelica standard libraries and the OpenIPSL library (Fachini, Bhattacharjee, et al. 2023; Fachini, Pigott, et al. 2023). In (Fachini, Bhattacharjee, et al. 2023), the microgrid was developed with two combustion turbo generators and four steam turbo generators using OpenIPSL machine, prime mover, and control system models, and analyses were carried out to study contingency and modal analysis. This study was extended in (Fachini, Pigott, et al. 2023), where distributed energy resources such as PV were integrated, and fault studies were carried out. These microgrid models were also extended to perform Model Predictive Control (MPC) in (Fachini, Bogodorova, et al. 2024), where the linearized model was used to obtain the MPC solution, and time-domain simulations were performed to apply the MPC actions for safe island operation and re-synchronization.

Developing power system models with Modelica provides additional benefits brought through the FMI ecosystem of tools for new and emerging needs. An example is the `ModelicaGridData` tool which was developed and interfaced with OpenModelica and/or Dymola through their respective Python APIs to generate large data sets of different operating conditions and disturbances, which were then used to train machine learning-based stability assessment approaches (Dorado-Rojas et al. 2023). Another example in (Castro et al. 2022) illustrates how power system models built using the OpenIPSL library can be exported as FMUs to be tested on real-time simulators such as dSPACE. Other power system applications using Modelica include control coordination for HVAC/HVDC power systems (Bakhos et al. 2017; Babaeifar, Barsali, and Ceraolo 2023), implementing and validating variable-speed drive-based induction mo-

DOI 10.3384/ecp218713

tor models (Fachini, Castro, et al. 2024), building energy simulation (Luigi Vanfretti, Laughman, and Chakrabarty 2024), and special protection schemes (Jakobsen et al. 2022).

Beyond power systems, Modelica has proven effective and extensively used in other domains such as thermofluid systems (Steinmann, Herold, and Schirmer 2024), aerospace (Reiner 2022), robotics and mechatronics (Reiser and Reiner 2023), and other industrial applications (Weber, Cartignij, and Zimmer 2023).

1.2 Contributions and Paper Organization

The three specific requirements of this study include: (a) the development of GFM inverter models suitable for large-scale power system simulation, (b) parametrization of GFM inverters, and (c) minimization of both manual and computational effort required for deployment and parameterization. This paper makes the following novel contributions to address the requirements:

1. Provides a guideline for modeling droop-controlled GFM IBRs using Modelica.

2. Develops a systematic approach to validate various subsystems of the GFM IBR model inside Modelica prior to full-scale deployment.

3. Develop interfaces to generate FMUs for the GFM inverter for integration with a specific power system simulator.

4. Discusses the benefits of using FMUs for large-scale power system simulation, such as 'easy-to-build' and 'easy-to-manage' advantages that involve minimal manual intervention and improved computational performance.

5. Examining scaling up of GFM IBR deployment via FMU across the New York State 5000-bus power grid.

The remainder of this paper is organized as follows: Section 2 presents the overview of the Modelica GFM library, Section 3 discusses the development of the GFM inverter model in Dymola, followed by FMU generation in Section 4 and interfacing with OPAL-RT's ePHASORSIM in Section 5. Results from simulation studies are discussed in Section 6 followed by the conclusions from this work in Section 7.

2 Modelica GFM Library Overview

This section presents the GFM IBR library in detail, including various dependencies, packages, and subpackages. In addition, it describes how the object-oriented approach is utilized to construct the GFM IBR model.

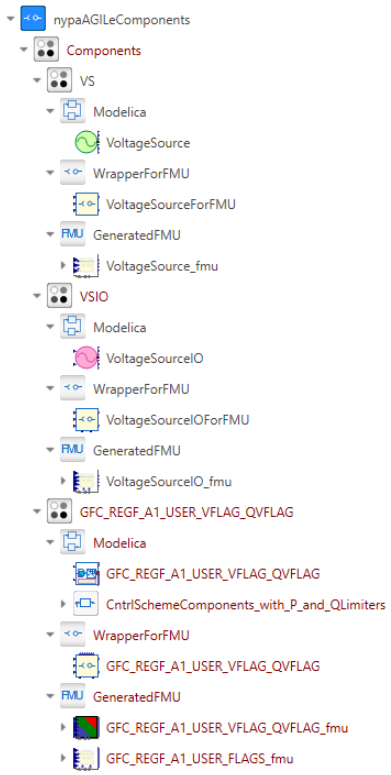


Figure 1. GFM library organization in Dymola

2.1 Organization of GFM Inverter Library

The GFM inverter components are built in the widely popular and proprietary Modelica-compliant language tool called Dymola (Brück et al. 2002). Dymola offers an intuitive graphical user interface with features that allow users to easily navigate built-in libraries and organize their own user-defined models. The GFM IBR library, called `nypaAGILeComponents`, needs to be combined with other Modelica libraries. These include two of Opal-RT's Modelica libraries, the `Opal_RT` [1], a proprietary library that uses the `Opal_RT` library to build power plant sub-systems called `GenUnit`, the Modelica Standard Library, and OpenIPSL. A snapshot of the `nypaAGILeComponents` library where the GFM inverter is built is shown in Figure 1.

There are three main components: (a) voltage source (VS), (b) controllable voltage source with input/output (VSIO), and (c) GFM control model REGFM_A1 (Du 2023). The library is organized reflecting the incremental manner in which components were developed and added. Each component has (a) the Modelica model, (b) an FMU wrapper, and (c) the generated FMU. The wrapper is a Modelica block that encapsulates the Modelica model and causalizes it when the FMU is generated. It contains the mandatory connectors, interfaces and parameters required by ePHASORSIM (described later in Section 4). In addition, each specific wrapper includes an additional set

[1] Available online: `https://github.com/Opal-RT-Technologies/modelica-ephasor-components`

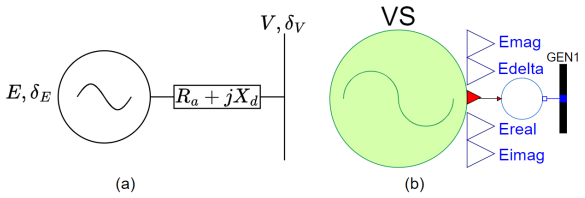


Figure 2. (a) Simple voltage source and (b) Modelica implementation

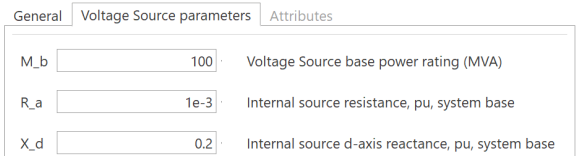


Figure 3. Simple voltage source parameters in Dymola

of parameters that the user can modify using the external tools, e.g., to select the control mode of the GFM inverter and parameterize it. The 'generated FMU' is a Modelica model that contains within the generated FMU the appropriate interfaces for use within the Modelica tool.

2.2 Model Building through Inheritance

Using inheritance, hierarchical and reusable component models were created for GFM modeling in Dymola. Each new component (such as the IBR model) inherits properties and behaviors from existing ones (such as the voltage source). This approach simplifies model development and maintenance by allowing modifications at higher levels of the hierarchy to propagate automatically to derived models. For example, the voltage source (VS) inside the GFM IBR extends from a "VS Droop model" that provides the required "measurement" outputs for the GFM controls. This component itself extends a simple VS whose voltage can be specified by external inputs, are applied to an internal impedance and couples with an acasual connector. An example of such inheritance is given through the statement: `extends VSIO.Modelica.VoltageSourceIO VS();`.

3 Droop-based GFM Components

The GFM inverter operates as a controllable VS behind a coupling reactance (Du et al. 2019). To understand the GFM inverter modeling in Dymola, we start with the design and validation of a simple VS, which forms the basis for the controllable VS that is needed for the GFM model. The controllable VS is followed by two droop-based control architectures (P-f and Q-V), overload limiters and a fault current limiter, as described below.

3.1 Simple Voltage Source

Consider a simple VS with an internal impedance of $R_a + jX_d$ connected to a bus shown in Figure 2. The objective is to find the internal voltage $E\angle\delta_E$ of the source from the active and reactive power (P and Q) measurements at the Gen1 terminal bus. The terminal voltage is denoted as $V\angle\delta_V = V_r + jV_i$. The current equation is given as:

$$I^* = \frac{P+jQ}{V_r+jV_i} = \frac{PV_r - jPVi + jQV_r + QV_i}{V_r^2+V_i^2} \tag{1}$$

The real part of the current is:

$$\Re\{I^*\} = I_r = \frac{PV_r + QV_i}{V_r^2+V_i^2} \tag{2}$$

The imaginary part of the current is:

$$\Im\{I^*\} = I_i = \frac{PVi - jQV}{V_r^2+V_i^2} \tag{3}$$

With internal source impedance of $R_a + jX_d$, the internal voltage ($E = E_{mag}\angle E_\delta$) is given as:

$$E = (I_r + jI_i)(R_a + jX_d) + (V_r + jV_i) \tag{4}$$

The real part of the internal voltage is:

$$\Re\{E\} = E_{real} = V_r - I_iX_d + I_rR_a \tag{5}$$

The imaginary part of the internal voltage is:

$$\Im\{E\} = E_{imag} = V_i + I_iR_a + I_rX_d \tag{6}$$

An excerpt of the VS model with parameters such as base and internal impedance is shown in Figure 3. The VS is implemented in Modelica, as shown in Listing 1. The terminal voltage values are interfaced with the mandatory (required by ePHASORSIM) connector of the `Opal_RT` library, `Powerpin (PwPin)`, through `p.vr` and `p.vi`. Details on PwPin and interfacing can be found later in Section 4.4.

3.1.1 VS Parameter Initialization

Using the VS parameters shown in Figure 4, the component is initialized as shown in the Listing 2. Note that we explicitly define variables with `start` values for initialization not only to provide a good starting guess value for the numerical solvers when simulating from different starting steady-state operating conditions, but also because these values need to be modified when the model is exported as an FMU to be used in a power system simulator.

Listing 1. Snippet of the Internal Voltage Source Equations

```
// Change from system (SB) to machine base (M_b)
parameter Real CoB=M_b/SB;
// Internal voltage source equations
Er = p.vr + CoB*R_a*p.ir - CoB*X_d*p.ii;
Ei = p.vi + CoB*R_a*p.ii + CoB*p.ir*X_d;
// Assing variables to outputs
Emag = E;
Edelta = delta;
Ereal = Er;
Eimag = Ei;
```

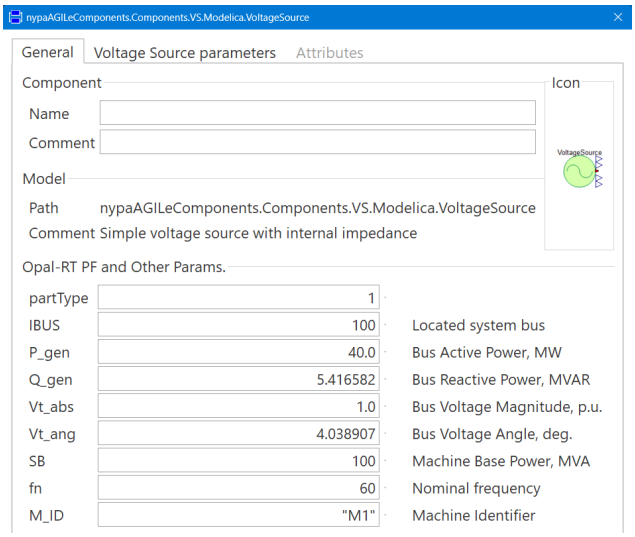


Figure 4. Parameters obtained from power flow used for voltage source initialization

Listing 2. Initialization of internal VS from power flow

```
protected
// Auxiliary parameters for initialization
v_0 = Vt_abs "Intial terminal Vmag from pf";
angle_0 =  Modelica.Units.Conversions.from_deg(
    Vt_ang) "Initial terminal V_angle from pf";
P_0 = P_gen "o/p power from pf";
Q_0 = Q_gen "o/p power from pf";

// Terminal Voltage variables
Real V(start=v_0) "Bus voltage magnitude";
Real anglev(start=angle_0) "Bus voltage angle";

// Public Auxiliary variables
Real P(start=P_gen/SB) "Active power";
Real Q(start=Q_gen/SB) "Reactive power";

// Public Internal voltage source variables
Real delta(start=delta0) "Internal VS angle";
Real E(start=E0) "Internal VS magnitude";

// Change of base
CoB=M_b/SB;
p0=P_0/M_b "Initial MW (machine base)";
q0=Q_0/M_b "Initial MVAR (machine base)";

// Initialization values
vr0=v_0*cos(angle_0);
vi0=v_0*sin(angle_0);
ir0=CoB*(p0*vr0 + q0*vi0)/(vr0^2 + vi0^2);
ii0=CoB*(p0*vi0 - q0*vr0)/(vr0^2 + vi0^2);

// Initialization of internal VS parameters
Er0 = vr0 + CoB*R_a*ir0 - CoB*X_d*ii0 "Initial
    value of E_real";
Ei0 = vi0 + CoB*R_a*ii0 + CoB*X_d*ir0 "Initial
    value of E_imag";
E0  = sqrt(Er0^2+Ei0^2);
delta0 = atan2(Ei0, Er0);

// Internal voltage source variables
Real Er(start=Er0);
Real Ei(start=Ei0);

equation
// Internal voltage source equations
delta = delta0 "assume Constant voltage angle";
E = E0 "assume Constant voltage magnitude";
Er = Er0;
Ei = Ei0
```

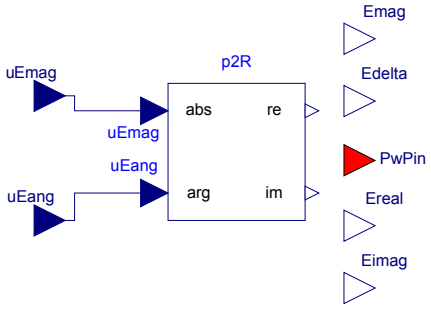


Figure 5. Controllable voltage source with two inputs

3.2 Controllable Voltage Source

The controllable VS is developed in Dymola based on the equations of the simple VS given in Equations (1) - (6). The controllable VS has two inputs `uEmag` and `uEang`, as shown in Figure 5, to vary the voltage magnitude and angle of the VS, respectively. To increase flexibility and facilitate testing, the input to the controllable voltage source can be provided from two sources: (a) Case 1: inject a voltage phasor to define the internal voltage of the source from an external tool (e.g., a power system simulator like ePHASOSRSIM), and (b) Case 2: inject a signal that is a deviation from the initial value of E0 and delta0 for internal testing. These two cases are shown in Listing 3.

Listing 3. Internal equations for controllable VS

```
if Case 1 then "input from external tool"
    E = uEmag;
    delta = uEang;
    Er = p2R.y_re "Real part of E";
    Ei = p2R.y_im "Imaginary part of E";
else "deviation from initial value"
    E = E0 + uEmag;
    delta = delta0;
    Er = Er0 + p2R.y_re;
    Ei = Ei0 + p2R.y_im;
```

3.3 Droop-based GFM Inverter Model

The controllable VS is used as a basic building block to build the GFM IBR model. Similarly to the controllable VS in Figure 5, the GFM inverter model has two inputs: `uEmag` and `uEang`. The input `uEang` that controls the voltage angle is determined by the P-f control loop, and `uEmag` that controls the voltage magnitude is determined by the Q-V control loop. The complete GFM IBR model, along with its control scheme, is shown in Figure 6.

3.3.1 Active Power-Frequency Droop-based Control

The active power-frequency (or P-f droop) droop-based control adjusts the output frequency of the inverter in response to changes in active power output. With $E\angle\delta_E$ as

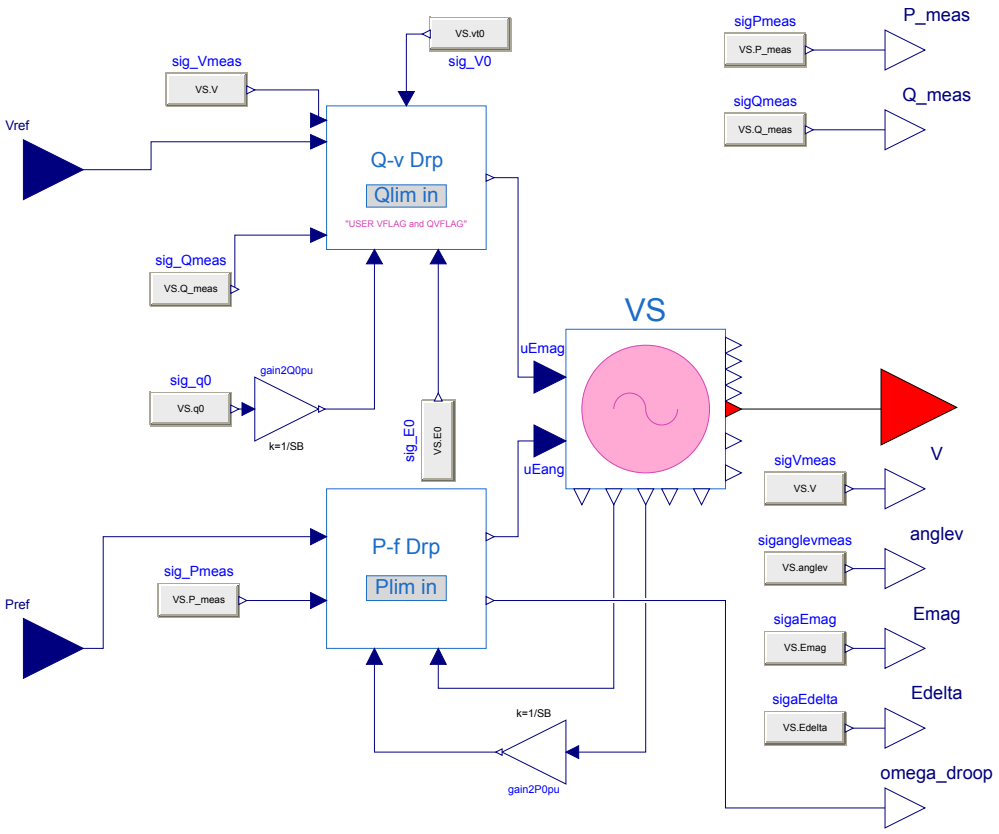


Figure 6. GFM Inverter Dymola Model Block Diagram

the internal voltage phasor and $V\angle\delta_v$ as the terminal voltage phasor, the active power output of the inverter P can be written as (Du 2023),

$$P \approx \frac{EV}{X_L} sin(\delta_E - \delta_v) \approx \frac{EV}{X_L}(\delta_E - \delta_v) \tag{7}$$

When a setpoint change or a disturbance causes the inverter active power to increase, the P-f droop-based control reduces the angular frequency ω of the internal voltage to reduce the inverter power output (Rocabert et al. 2012), which is modeled as a change in δ_E (Du 2023). The P-f droop-based control also ensures that multiple GFM inverters share power relative to their active power capacities under disturbances. The overall P-f droop-based control architecture is shown in 7, and the detailed P-f droop-based control loop is shown in Figure 8. The P-f droop loop takes the error between the reference active power `Pref` and the measured active power `Pfilt` (after passing through a low pass filter) as input, and provides the internal voltage phase angle and angular frequency of the VS as output. There is an auxillary input signal from the output power limiter block that prevents the active power output of the inverter from exceeding its limits. Note that the integrator in the control loop is initialized with the value `delta0` as discussed in Section 3.1.1 (see line 35 in Listing 2).

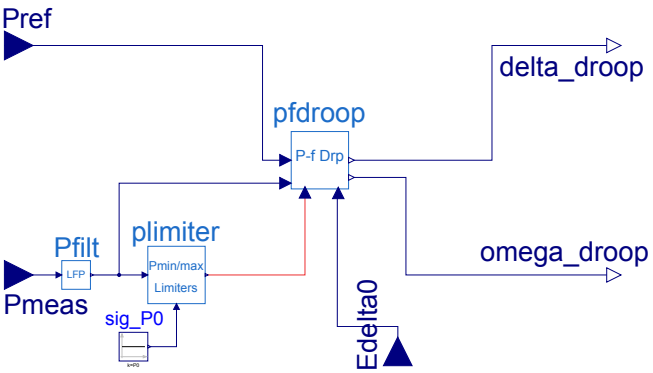


Figure 7. P-f Droop-based Control Architecture

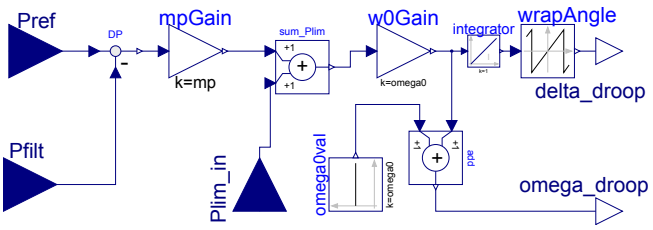


Figure 8. P-f Droop-based Control Function

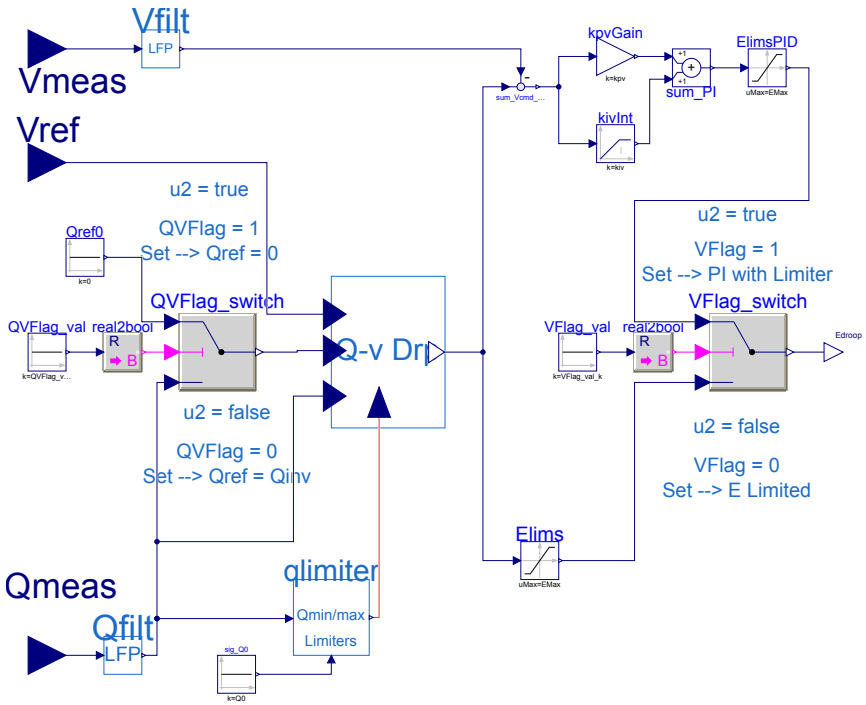


Figure 9. Q-V droop-based control architecture

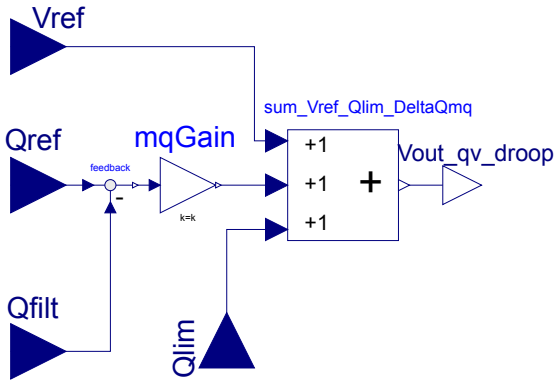


Figure 10. Q-V Droop-based Control Function

3.3.2 Reactive Power-Voltage Droop-based Control

The reactive power-voltage (or the Q-V) droop-based control adjusts the inverter reactive power output in response to changes in the voltage at its terminals. The reactive power output of the inverter can be written as (Du 2023),

$$Q \approx \frac{E^2 - EV\cos(\delta_E - \delta_v)}{X_L} \approx \frac{E(E-V)}{X_L} \tag{8}$$

When there is a step change in voltage setpoint or an event disturbs the terminal voltage, the Q-V droop-based control manipulates the magnitude of the internal voltage of the inverter to adjust the reactive power output accordingly. The Q-V droop has two voltage control modes; it can either regulate: (a) the internal voltage, $E\angle\delta_E$, or (b) terminal voltage, $V\angle\delta_V$, which is determined by the setting of `VFlag`. In addition, the Q-V droop has two reactive power control modes that are set by `QVFlag`, which can be set to represent the plant controller strategy of adjusting either the reactive power or voltage setpoint. When multiple GFM inverters are connected to the grid in parallel, the Q-V droop-based control prevents the circulation of the reactive power.

The input/output and control blocks of the Q-V droop scheme are shown in Figure 9. The Q-V droop model takes as input the reference voltage, the measured reactive power (after passing through the low-pass filter) and two flags, `VFlag` and `QVFlag`, and provides the internal voltage magnitude as output, as shown in Figure 10. In addition, the auxiliary signal from the output power limiter block prevents the reactive power output of the inverter from exceeding the converter's limits. Note that the integrator in the control loop is initialized with the value `v_0` as discussed in Section 3.1.1 (see line 9 in Listing 2).

3.4 Overload Limiters

For both control loops, there are corresponding maximum and minimum MW and MVAR limits to prevent the inverter from exceeding rated capacities. The block diagram for the limiters are shown in Figure 11 and Figure 12.

3.5 Fault Current Limiting Function

When there is a short-circuit and fault currents are very high (exceeding inverter rated current I_{maxF}), the GFM internal voltage $E\angle\delta_E$ will be calculated based on the inverter terminal voltage $V\angle\delta_v$, coupling reactance X_L and current phasor $I_{maxF}\angle\phi_{lim}$, which represents the limited current. The GFM internal voltage under fault conditions is given as:

$$E\angle\delta_E = V\angle\delta_v + jX_L I_{maxF}\angle\phi_{lim}. \tag{9}$$

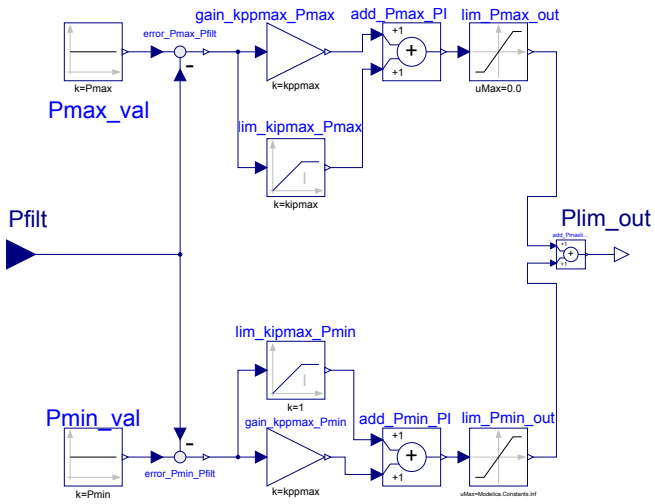


Figure 11. MW Limiter

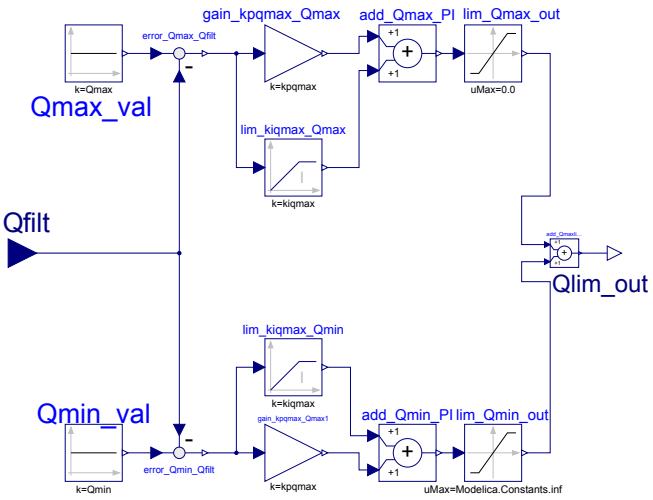


Figure 12. MVAR limiter

Due to space limitations, block diagrams for current limiters are not shown in this paper.

4 FMU Generation

In this section, we briefly discuss how Modelica model is setup to generate an FMU.

4.1 System Dependencies

To generate FMUs using Dymola, compatible with Windows and Linux (required by the ePHASORSIM simulator from Opal-RT), the following dependencies are necessary: Windows 10 or later versions, a suitable compiler for C++ (such as Visual Studio 2019 with the 'Desktop development with C++' workload enabled), and Windows Subsystem for Linux (WSL) 2 with Linux kernel update package.

4.2 Windows Subsystem for Linux (WSL)

The cross-compilation feature available in Dymola allows Dymola to compile models to be used on a Linux target. This is done through WSL, which enables a Linux system to run inside a Windows machine using WSL. FMUs generated with this feature can be executed by software tools native to Windows and Linux-based operating systems (OS). This is because the generated FMU will have specific target binaries for each of those OSs. ePHASORSIM crucially needs this feature since it relies on Windows for offline and Linux for real-time simulations.

4.3 FMU with and without Source Codes

FMUs can be generated with or without the C source code. With Dymola, this capability depends on whether or not the user has a license that enables code export. Regardless, both options are useful, since a user with a local "vanilla" Dymola license could regenerate the FMUs and execute off-line local simulations. Export of source code is only required when running the ePHASORSIM in real time. For FMUs without source code, the FMU package contains only the compiled binaries of the model, not the C source code used to build the binary (.dll or .so). With the source code export, users can recompile the FMU's code, offering greater flexibility.

ePHASORSIM has specific requirements for the FMUs it receives from Dymola, for example, it requires that internal variables of the FMU can be accessed, which is enabled in Dymola with the flag `Advanced.FMI.BlackBoxModelDescription := flase`. In this paper, we create the `FMUExportUtilities` feature that allows to create FMUs both with and without source code with the required flags for import in ePHASORSIM. Figure 13 shows the GUI of the function call used to generate the FMU with/without source code. Listing 4 shows the flags necessary to generate the FMU with source code from Dymola while taking into account the requirement of the availability of the source code generation license for code export. Note that OPAL-RT ePHASORSIM currently supports FMI version 1. While three types of interfaces are defined in FMI standard, we are interested in the FMI for Model Exchange (ME). ME implements only the model and not the required solution. It exposes the differential equations to an external solver that imports the model and performs numerical integration.

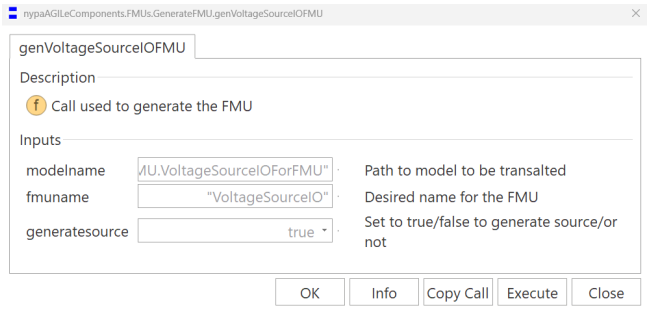


Figure 13. Custom function to generate FMUs with/without source code

Listing 4. Custom function to Generate FMU with Source Code

```
function generateFMUWithSourceCode
```

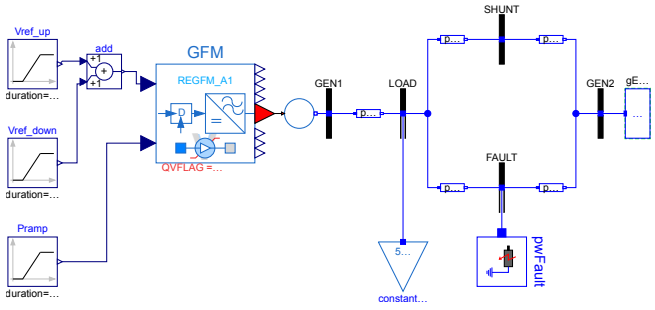


Figure 14. Testing GFM Model within a power system model built using OpenIPSL

```
  "Call used to generate the FMU with source
      code"
  extends Modelica.Icons.Function;
  input String modelname = "IBR_VS_Droop_FMU"
  input String fmuname = "GENSAL";
algorithm
  // Flags used in "..\FMUCreator\dymolaFlag.cfg
      "
  Advanced.CheckPackageRestriction :=false;
  Advanced.FMI.xmlIgnoreProtected :=false;
  Advanced.FMI.xmlIgnoreLocal :=false;
  Advanced.FMI.BlackBoxModelDescription :=false;
  Advanced.Define.AimForHighAccuracy :=false;
  Advanced.Define.NewJacobian :=false;
  Advanced.EnableCodeExport := true;
  Advanced.SourceCodeExportNormal :=true;
  Advanced.FMI.CrossExport :=true;
  Advanced.FMI.FMUIncludeSource :=true;
  Advanced.FMI.FMUSourceCodeUniqueNaming :=true;
  // Create FMU
  translateModelFMU(modelname, false, fmuname, "
      1", "all", true, 1);
end generateFMUWithSourceCode;
```

4.4 Testing of Models and FMUs in Dymola

The models and FMUs generated by Dymola are tested within the Dymola interface before interfacing with ePHASORSIM. In this section, we demonstrate both (a) testing the Modelica model within a power system model and (b) testing the FMU in Dymola within a power system model.

The GFM inverter model is tested within a power system modeled using OpenIPSL. A Single Machine Infinite Bus (SMIB) system with one load is extended from the OpenIPSL library using `extends OpenIPSL.Tests.BaseClasses.SMIB(SysData(fn=60), pwFault(R = 1e-6, X = 1e-3, t1 = 32.0, t2 = 32.15))` with a fault duration of 0.15 s. The setup is shown in Figure 14.

To connect the GFM model with the OpenIPSL SMIB within Dymola, the `Powerpin (PwPin)` connector is used by instantiating it from the `OPAL_RT` class `OpalRT.NonElectrical.Connector.PwPin`. The `PwPin` is required by ePHASORSIM, as when the FMUs are loaded, ePHASORSIM will determine if `PwPin` has been used or not. This is done by the software to force conformance to its internal numerical solver interface. The `PwPin` has two flow variables (`ir` and `ii`) and two potential variables (vr and vi). The implementation of `PwPin` is given in Listing 5.

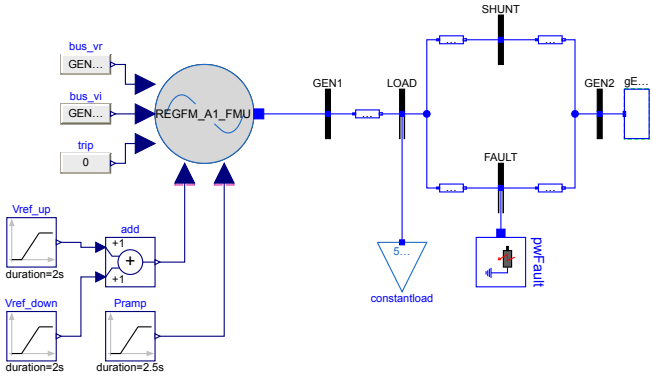


Figure 15. Testing GFM FMU in Dymola within a power system model built using OpenIPSL.

Listing 5. `PwPin` Connector from `Opal_RT`'s library

```
connector PwPin
    Real vr "real part of the voltage";
    Real vi "imaginary part of the voltage";
    flow Real ir "real part of the current";
    flow Real ii "imaginary part of the current"
        ;
end PwPin;
```

To interface the GFM model developed according to ePHASORSIM's requirements (with `PwPin`) with electrical components from the OpenIPSL library inside Dymola, an additional connector is required. In this connector, the variables of `PwPin` (instantiated as `pwPinA`) are assigned to the corresponding variables of the OpenIPSL electrical connector (instantiated as `pwPinB`). The interface between the power pins of both libraries is shown in Listing 6. This setup is used to test all models and submodels inside Dymola.

Listing 6. Code Excerpt of the Connector Interfacing the `Opal_RT` and OpenIPSL Libraries

```
model Opal2OpenIPSL "Interface between power
    pins of both libraries"
equation
  pwPinB.vr = pwPinA.vr;
  pwPinB.vi = pwPinA.vi;
  pwPinB.ir = pwPinA.ir;
  pwPinB.ii = pwPinA.ii;
end Opal2OpenIPSL;
```

The generated FMU model is also tested within a power system modeled in Dymola. A similar setup is shown in Figure 15.

Note that the `nypaAGILecomponents` library was developed and tested with Dymola 2024X and compatibility with other Modelica-compliant tools has not been verified. Meanwhile, the Opal-RT library is compatible with both `OpenModelica` and Dymola, while the `OpenIPSL` is compatible with Dymola, OpenModelica, Modelon Impact, and Wolfram SystemModeler.

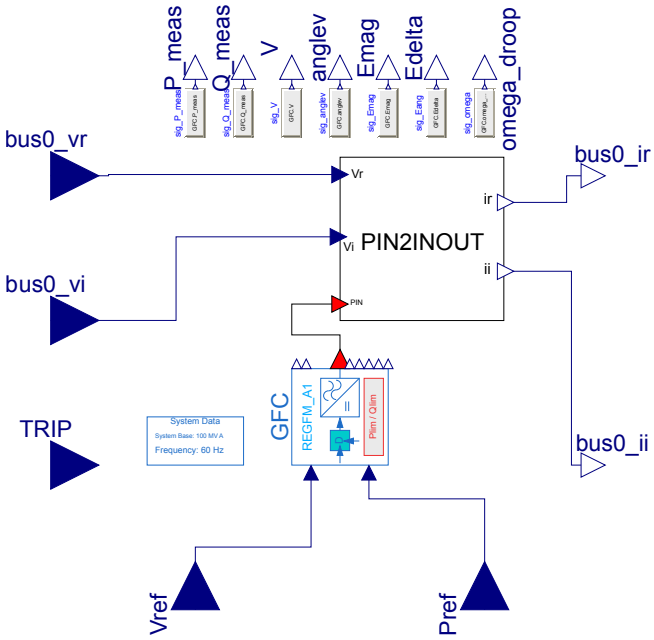


Figure 16. Wrapped GFM Model with Required Interfaces by ePHASORSIM for Export as FMU

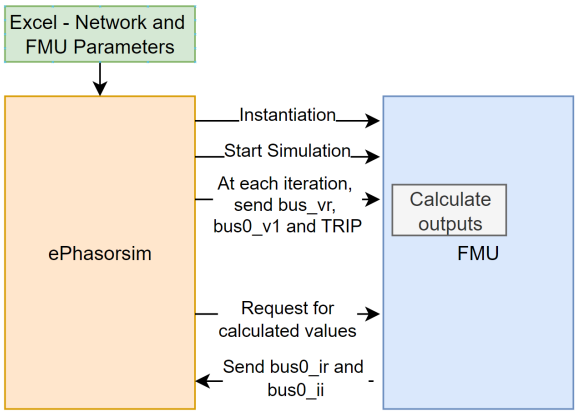


Figure 17. Flowchart of communication between OPAL-RT ePHASORSIM and FMU

5 Interfacing FMUs with ePHASORSIM

5.1 FMU - ePHASORSIM `PIN2INOUT` Interface

To interface the FMU with ePHASORSIM, an FMU wrapper is created. As discussed in Section 2.1, the wrapper includes the mandatory connectors, interfaces, and parameters required by ePHASORSIM and the set of parameters that the user can modify using external tools. The FMU wrapper is shown in Figure 16.

A `PIN2INOUT` block is required by ePHASORSIM to be used as an interface with FMU, which enforces the input (voltage) and generates (current) outputs as expected by ePHASORSIM. OPAL-RT's ePHASORSIM requires the mandatory `PwPin`. The block `PIN2INOUT` converts the acasual `PwPin` variables to casualized signals, defining the voltages as inputs and currents as output with the current flow direction required by ePHASORSIM. Listing 7 an excerpt of the `PIN2INOUT` block source code.

Listing 7. Excerpt of the `PIN2INOUT` block

```
block PIN2INOUT
equation
  p.ir = -ir;
  p.ii = -ii;
  p.vr = vr;
  p.vi = vi;
end PIN2INOUT;
```

5.2 Data Exchange between ePHASORSIM and FMU

Figure 17 shows the data exchange between OPAL-RT ePHASORSIM and the FMU generated from Dymola. In ePHASORSIM, all parameters for a power system model are defined in an Excel file. Once ePHASORSIM reads the network and FMU parameters from the Excel file, it instantiates the FMU and passes the parameter data to it. ePHASORSIM uses a fixed time-step solve, hence, at each simulation step, it sends the terminal bus voltage real and imaginary values, `bus0_vr` and `bus0_vi`, and any inputs defined in the wrapper to the FMU. The FMU equations are evaluated with the received values, and the results are sent back to ePHASORSIM in the form of real and imaginary current injection values `bus0_ir` and `bus0_ii`, and any of the output values included in the FMU wrapper. ePHASORSIM then solves for all variables within its internal models and assembles the overall network solution.

6 Simulation Results

6.1 Controllable Voltage Source

As discussed in Section 4.4, the models are interfaced to the OpenIPSL SMIB for testing within the Dymola environment. Testing within Dymola itself offers the advantage of efficient debugging, validation, and iterative development of models and FMUs without the need for external tools. The setup for testing the controllable VS is shown in Figure 18.

Figure 19 shows the responses of the internal magnitude and angle of the controllable voltage source to the ramp functions within Dymola. For `Emag`, the ramp was activated at 4 s., changing the value from 1.01439 p.u. to 1.03439 p.u. within 1 s. For `Eang`, the ramp was activated at 6s changing the value from 0.149386 rad to −0.243364 rad within 3 s. Meanwhile, Figure 20 shows the response of the active and reactive power, P and Q. It becomes apparent from the plot how a change in the internal voltage source δ_E has a direct impact on P according to Equation (7), while changes in E have a direct impact on Q as per Equation (8).

DOI 10.3384/ecp218713

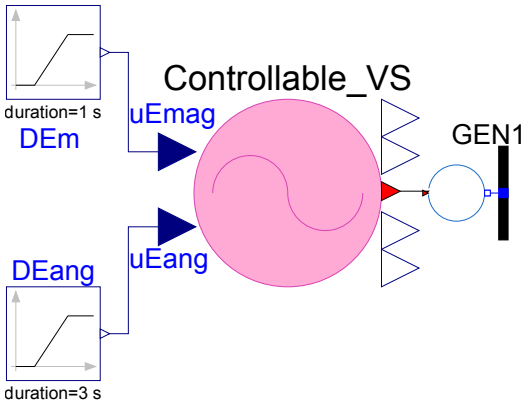


Figure 18. Controllable voltage source connected to a bus with inputs to change voltage magnitude and angle

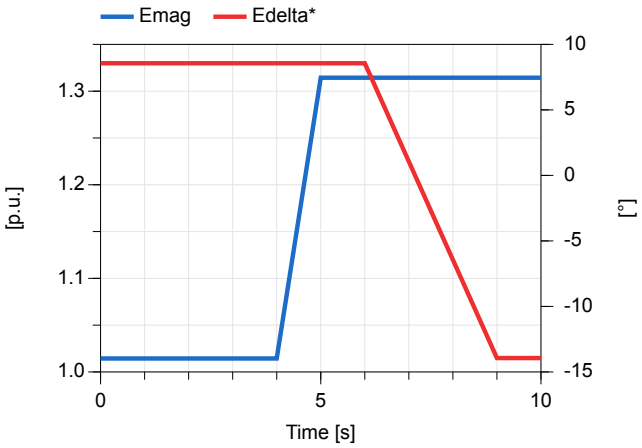


Figure 19. Controllable voltage source internal magnitude and angle ($E \angle \delta_E$) response to ramp function in Dymola

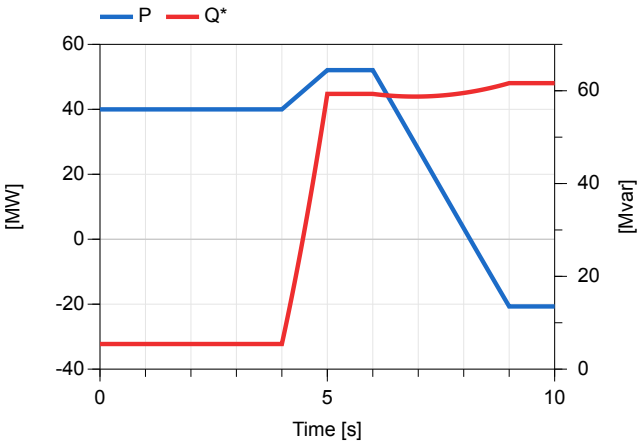


Figure 20. P and Q output of the controllable voltage source due to a ramp applied to its reference inputs in Dymola

6.2 Implementation of GFM inverter FMU on NYS Transmission

GFM IBRs, modeled as FMU, were integrated into the NYS 5000-bus, 800-machine transmission system in ePHASORSIM. In addition to the GFM components, models of synchronous generators (SG), exciters, turbines, wind generators, and power system stabilizers, developed using Modelica language and exported as FMU, were interfaced with ePHASORSIM. A total of 9 GFM IBR with 6,272.5 MVA capacity was configured to substitute traditional generators within the NYS transmission system. The impact on the transient stability was analyzed under different conditions, such as changes in GFM active power and voltage setpoints. For example, Figure 21 shows the response of when the MW set-point on one IBR increased from 187.46 MW to 300 MW. Figure 22 shows the response of the IBRs when the voltage setpoint on target bus was changed from 1.019 p.u. to 1.017 p.u. This demonstrates that the FMU generated from Dymola is successfully integrated with a large-scale power system, highlighting the approach that is easy to deploy, scalable, and computationally efficient.

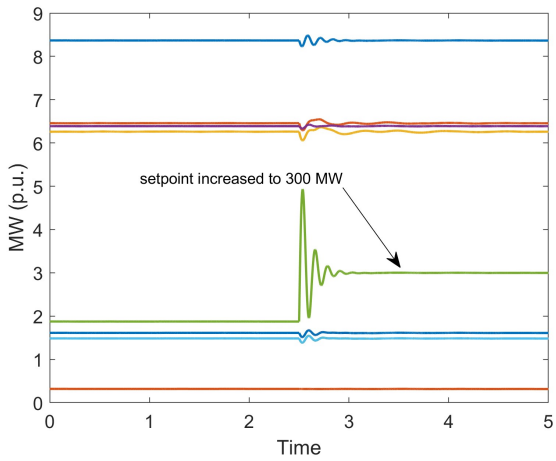


Figure 21. GFM response to changes in MW setpoint in OPAL-RT ePHASORSIM

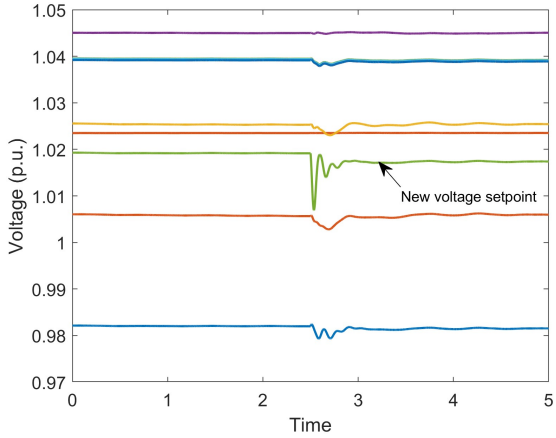


Figure 22. GFM response to changes in voltage setpoint in OPAL-RT ePHASORSIM

Table 1. Comparison between Simulink-Based and FMU-Based GFM IBR Deployment in ePHASORSIM

Aspect	**Simulink-Based Deployment**	**FMU-Based Deployment**
Controller Integration	Manual copy-paste of droop-based GFM controller block each time	Controller is pre-embedded in FMU; no need for repetition
Signal Wiring	Manual setup for each new GFM (all inputs/outputs to/from solver)	Minimal signal mapping (e.g., `Pset`, `Vset`) required
Model Scalability	Tedious and error-prone with more GFMs	Easily scalable to large numbers of GFMs (e.g., hundreds)
Transformer Setup	Details must be manually specified in Transformer sheet	Can be included directly in FMU or in the additional FMU-named sheet
Error Probability	High, especially with manual wiring and naming	Lower, due to fewer steps and more automation
Time Efficiency	Time-consuming due to repetitive steps and complex wiring	Very time-efficient and streamlined for high GFM penetration
Maintenance & Debugging	Difficult to trace and fix due to signal complexity	Easier maintenance

One significant advantage observed during the development process was the ease and efficiency of deploying FMUs with ePHASORSIM compared to using MATLAB/SIMULINK to build and deploy IBRs manually. A detailed comparison of both approaches is given in Table 1. These demonstrate that FMUs are the ideal candidates for efficient and scalable simulation.

7 Conclusion

This paper demonstrates the advantage of developing and deploying FMUs for large-scale power system simulations. A grid-forming inverter model was developed in Modelica with Dymola and integrated as an FMU with OPAL-RT's ePHASORSIM simulator to evaluate the impact of widespread renewable energy integration on the NYS transmission system. The development of the GFM FMU inverter was made easier through the object-oriented modeling features of Modelica. First, a generic voltage source model was implemented which was then extended to create a controllable voltage source and was further extended to develop the GFM-controlled inverter. With a 'build-once deploy-anywhere' approach, this paper shows that FMU offers significant time savings in modeling, debugging, and parametrization compared to other power system simulation approaches. The FMUs were tested inside Dymola and integrated with the NYS 5000−bus system in ePHASORSIM. In future, we will extend our work by (a) building FMU models for the virtual synchronous machine GFM model, (b) exploring FMU interoperability by deploying FMUs on different power system simulation tools and (c) investigating the impact of IBR parameter uncertainties on overall system stability.

References

Babaeifar, Mohammad, Stefano Barsali, and Massimo Ceraolo (2023). "Simulation of Sapei HVDC using modelica language". In: *2023 AEIT HVDC International Conference (AEIT HVDC)*. IEEE, pp. 1–6.

Bakhos, Gianni et al. (2017). "Hybrid AC/DC Power System Stability: An Attempt of Global Approach". In: *Available at SSRN 4704802*.

Baudette, Maxime et al. (2018). "OpenIPSL: Open-instance power system library—update 1.5 to "iTesla power systems library (iPSL): A modelica library for phasor time-domain simulations"". In: *SoftwareX* 7, pp. 34–36.

Baur, Marcus, Martin Otter, and Bernhard Thiele (2009). "Modelica libraries for linear control systems". In: *Proceedings 7th Modelica Conference*. DOI: 1, pp. 593–602.

Brück, Dag et al. (2002). "Dymola for multi-engineering modeling and simulation". In: *Proceedings of modelica*. Vol. 2002. Oberpfaffenhofen, pp. 1–6.

Casella, Francesco and Alberto Leva (2005). "Object-oriented modelling & simulation of power plants with modelica". In: *Proceedings of the 44th IEEE Conference on Decision and Control*. IEEE, pp. 7597–7602.

Castro, Marcelo de et al. (2022). "Power System Real-Time Simulation using Modelica and the FMI". In: *Modelica Conferences*, pp. 85–92.

de Castro, Marcelo et al. (2023-02). "Version [OpenIPSL 2.0.0] - [iTesla Power Systems Library (iPSL): A Modelica library for phasor time-domain simulations]". In: *SoftwareX* 21, p. 101277. ISSN: 2352-7110. DOI: 10.1016/j.softx.2022.101277. URL: https://www.sciencedirect.com/science/article/pii/S2352711022001959 (visited on 2023-10-17).

Dorado-Rojas, Sergio A et al. (2023). "ModelicaGridData: Massive power system simulation data generation and labeling tool using Modelica and Python". In: *SoftwareX* 21, p. 101258.

Du, Wei (2023). *Model specification of droop-controlled, grid-forming inverters (REGFM_A1)*. Tech. rep. Pacific Northwest National Laboratory (PNNL), Richland, WA (United States).

Du, Wei et al. (2019). "A comparative study of two widely used grid-forming droop controls on microgrid small-signal stability". In: *IEEE Journal of Emerging and Selected Topics in Power Electronics*. Vol. 8. 2. IEEE, pp. 963–975.

Fachini, Fernando, Srijita Bhattacharjee, et al. (2023). "Exploiting Modelica and the OpenIPSL for University Campus Microgrid Model Development". In: *Modelica Conferences*, pp. 285–292.

Fachini, Fernando, Tetiana Bogodorova, et al. (2024). "A microgrid control scheme for islanded operation and re-synchronization utilizing Model Predictive Control". In: *Sustainable Energy, Grids and Networks* 39, p. 101464. ISSN: 2352-4677. DOI: https://doi.org/10.1016/j.segan.2024.101464. URL: https://www.sciencedirect.com/science/article/pii/S2352467724001930.

Fachini, Fernando, Marcelo de Castro, et al. (2024). "Modeling of Induction Motors and Variable Speed Drives for Multi-Domain System Simulations Using Modelica and the OpenIPSL Library". In: *Electronics* 13.9, p. 1614.

Fachini, Fernando, Aisling Pigott, et al. (2023). "Developing a campus microgrid model utilizing modelica and the OpenIPSL library". In: *2023 11th Workshop on Modelling and Simulation of Cyber-Physical Energy Systems (MSCPES)*. IEEE, pp. 1–6.

Fritzson, Peter and Vadim Engelson (1998). "Modelica—A unified object-oriented language for system modeling and simulation". In: *ECOOP'98—object-oriented programming: 12th European conference Brussels, Belgium, July 20–24, 1998 proceedings 12*. Springer, pp. 67–90.

Gomez, Francisco J et al. (2018). "Multi-domain semantic information and physical behavior modeling of power systems and gas turbines expanding the common information model". In: *IEEE access* 6, pp. 72663–72674.

Hellerer, Matthias, Tobias Bellmann, and Florian Schlegel (2014). "The DLR Visualization Library-recent development and applications". In.

Jakobsen, Sigurd Hofsmo et al. (2022). "Modelica-based parallel computing framework for power system adaptive special protection schemes". In: *2022 Open Source Modelling and Simulation of Energy Systems (OSMSES)*. IEEE, pp. 1–6.

Laera, Giuseppe et al. (2022). "Guidelines and Use Cases for Power Systems Dynamic Modeling and Model Verification using Modelica and OpenIPSL". In: *Modelica Conferences*, pp. 146–157.

Modelica Association (2025). *FMI 3.0 Implementers' Guide*. URL: https://modelica.github.io/fmi-guides/main/fmi-guide/ (visited on 2025-04-03).

Mukherjee, Biswarup and Luigi Vanfretti (2018). "Modeling of PMU-based islanded operation controls for power distribution networks using Modelica and openIPSL". In: *Proceedings of The American Modelica Conference*, pp. 9–10.

Nikolaev, Nikolay et al. (2020). "PSS/E Based Power System Stabilizer Tuning Tool". In: *2020 21st International Symposium on Electrical Apparatus & Technologies (SIELA)*. IEEE, pp. 1–6.

Reiner, Matthias J (2022). "Simulation of the on-orbit construction of structural variable modular spacecraft by robots". In: *Modelica Conferences*, pp. 38–46.

Reiser, Robert and Matthias J Reiner (2023). "Modeling and simulation of dynamically constrained objects for limited structurally variable systems in Modelica". In: *Modelica Conferences*, pp. 151–158.

Rocabert, Joan et al. (2012). "Control of power converters in AC microgrids". In: *IEEE transactions on power electronics* 27.11, pp. 4734–4749.

Segerstrom, Eric et al. (2023). "Modeling Components of a Turbine-Generator System for Sub-Synchronous Oscillation Studies with Modelica". In: *Modelica Conferences*, pp. 541–550.

Senate, NY State (2023-05). *N.Y. Pub. Svc. Law § 66-P – Establishment of a renewable energy program*. en. Section: Public Service Law. URL: https://newyork.public.law/laws/n.y._public_service_law_section_66-p (visited on 2025-05-01).

Steinmann, Christoph, Johannes Herold, and Jens Schirmer (2024). "Model-Based Design and Characterization of an Actuator with Low-Boiling Liquid". In: *Modelica Conferences*, pp. 7–14.

Thiele, Bernhard et al. (2017). "Towards a standard-conform, platform-generic and feature-rich Modelica device drivers library". In: *Proceedings of the 12th International Modelica Conference*. Linköping University Electronic Press, 2017, pp. 713–723.

Vanfretti, L. et al. (2016). "iTesla Power Systems Library (iPSL): A Modelica library for phasor time-domain simulations". In: *SoftwareX* 5, pp. 84–88. ISSN: 2352-7110. DOI: https://doi.org/10.1016/j.softx.2016.05.001. URL: https://www.sciencedirect.com/science/article/pii/S2352711016300097.

Vanfretti, Luigi, Christopher R Laughman, and Ankush Chakrabarty (2024). "Integrating Generative Machine Learning Models and Physics-Based Models for Building Energy Simulation". In: *Modelica Conferences*, pp. 178–188.

Vanfretti, Luigi, Wei Li, et al. (2013). "Unambiguous power system dynamic modeling and simulation using modelica tools". In: *2013 IEEE Power & Energy Society General Meeting*. IEEE, pp. 1–5.

Vanfretti, Luigi, Biswarup Mukherjee, et al. (2019). "Automatic re-synchronization controller analysis within a multi-domain gas turbine and power system model". In: *2019 7th Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*. IEEE, pp. 1–5.

Weber, Niels, Camiel Cartignij, and Dirk Zimmer (2023). "Using the DLR Thermofluid Stream Library for Thermal Management of Fuel Cell Systems in Aviation". In: *Modelica Conferences*, pp. 415–422.

Winkler, Dietmar (2018). "Analysing the stability of an Islanded hydro-electric power system". In: *Modelica Conferences*.

DOI
10.3384/ecp218713

Design and Simulation Validation of Steam Power Systems Based on MBSE

Ruirui Zeng[1] Hui Gao[1] Jian Liu[2] Xingjian Han[2] Wei Liu[1]
Lei Huang[1] Qi Liu[1]

[1]Suzhou Tongyuan Software&Control Tech. Co. Ltd,
Suzhou,China,{zengrr, gaohui, liuw, huangl, liuq}@tongyuan.cc
[2]China Electronic Product Reliability and Environmental Testing Research Institute(The fifth electronic research institute of MIIT)
Guangzhou, China, {liujian,hanxingjian}@ceprei.com

Abstract

Steam power systems, as one of the critical power systems in industrial applications, require rigorous design and verification processes. Model-Based Systems Engineering (MBSE) provides a structured approach to decomposing system architecture from top to bottom and enabling multi-disciplinary collaborative design, ensuring precise requirement management and efficient design processes. To accelerate the iterative design and verification of steam power systems, this paper employs the SysML language to conduct requirement, functional, structural, and parametric analyses, thereby completing the system architecture design. The seamless transformation from system design architecture to system simulation architecture is achieved based on the SysML-to-Modelica tool. Additionally, Modelica simulation technology is utilized to construct simulation models and perform dynamic scenario analyses of the system. Innovatively, this paper proposes a closed-loop technical approach for steam power system design, simulation, and verification, which effectively optimizes system design and improves the efficiency of both design and verification processes.

Keywords: MBSE, Steam Power System, SysML, Modelica

1 Introduction

With the rapid advancement of industrial technology, the complexity of engineering systems has increased dramatically, and traditional systems engineering (TSE) is increasingly showing limitations in addressing the design and verification of complex systems. Meanwhile, the rapid development of information technology centered on modeling has provided new solutions for systems engineering. Driven by requirements and technology, the International Council on Systems Engineering (INCOSE) officially proposed Model-Based Systems Engineering (MBSE) in its Vision 2020 for Systems Engineering in 2007, clarifying the trend of future methodological and technological evolution in systems engineering toward MBSE. As the application of modeling methods in engineering activities, MBSE spans the entire process from top-level conceptual design to physical implementation, relying on interdisciplinary expertise to unify the expression of complex system development methods, thereby promoting interdisciplinary integration and the implementation of large-scale complex systems.

Currently, MBSE is widely applied in the design of complex large systems, particularly in the fields of aeronautics, astronautics, and shipbuilding, where significant achievements have been made. Overseas research institutions have deeply explored the foundational theories and modeling mechanisms of MBSE and conducted application studies. For example, in 2009, the U.S. National Aeronautics and Space Administration (NASA) applied MBSE to evaluate the practicality of new technologies, construct test and evaluation architecture configurations, and develop mission operation concepts[1]. The U.S. Navy has applied MBSE to the configuration management of hardware and software in submarine combat systems, achieving parallel management of internal and external interfaces in the Submarine Warfare Federated Tactical Systems (SWFTS), which includes over 20 projects, 35 subsystems, and 50,000 model elements[2]. In China, the China Shipbuilding Systems Engineering Research Institute, addressing the high complexity and strong integration of ship systems, adopted MBSE methods in the development of shipborne electronic countermeasure systems, successfully resolving design challenges associated with traditional systems engineering[3].

However, despite significant achievements in complex system design, MBSE still faces the following challenges in the specific domain of ship steam power systems: 1) The limitations of traditional systems engineering methods in complex system design, which struggle to effectively address the complexity of multi-disciplinary coupling and requirement management; 2) The potential of MBSE in ship steam power systems has not been fully realized, particularly in the systematic solutions for multi-disciplinary modeling and simulation verification; 3) Existing methods fail to comprehensively cover system performance and reliability assessments

during design verification and optimization, leading to long development cycles and high costs.

This paper focuses on ship steam power systems and employs MBSE methods based on SysML and Modelica languages. Using SysML, it establishes models for requirements, functions, architecture, and parameters, completing stakeholder analysis, requirement analysis, functional analysis, structural analysis, and parametric analysis. Modelica is used to build system simulation models to verify system requirement indicators and validate functional logic. This approach integrates forward design processes with bottom-up design and simulation, driving debugging simulation, co-simulation, system verification, and process validation. By exposing design issues early, this method reduces the number of product development iterations, lowers costs, shortens development cycles, and improves the efficiency of complex system design and development. This study provides a systematic solution for the efficient design and verification of ship steam power systems and promotes the application and development of MBSE in complex ship power systems.

2 System design method

2.1 Steam Power System Architecture Design

Steam power systems[4] use thermal energy as the driving force to convert water into high-temperature, high-pressure steam. This steam is then transformed into mechanical energy through turbines and other mechanical components, which drives generators to produce electricity. The system includes functions such as steam generation and transportation, steam condensation, exhaust steam transportation, equipment cooling, equipment lubrication, and steam sealing and extraction. These functions correspond to subsystems such as the steam system, condensate system, exhaust steam system, cooling water system, lubricating oil system, and steam sealing and extraction system. The subsystems interact through thermodynamic equipment. For example, turbines convert high-temperature, high-pressure steam from the steam system into mechanical energy and exhaust steam, which is transported by the exhaust steam system to the condenser. The lubricating oil cooler facilitates heat exchange between the cooling water system and the lubricating oil system, ensuring proper cooling of the lubricating oil. The logical interactions between subsystems in steam power systems are complex, involving a large number of components and design considerations spanning multiple disciplines, including thermodynamics, mechanical design, fluid mechanics, and control systems.

In recent years, traditional document-based methods[5] have shown the following limitations in complex system design: 1) Complex document management, making it difficult to track requirement changes and design iterations; 2) Lack of visualization and structured expression, making it challenging to clearly describe the logical relationships of complex systems; 3) Dispersed design processes, hindering multi-disciplinary collaborative design; 4) Low efficiency in verification and optimization, making it difficult to identify potential issues early, leading to long development cycles and high costs. Due to their lack of flexibility and reusability, traditional document-based design methods are gradually being replaced by model-based systems engineering (MBSE) approaches. Driven by advancements in computer and software technologies, MBSE has been widely applied in system modeling and design. This paper combines SysML language and Modelica simulation technology to achieve design and verification of steam power systems, optimizing system design management. The SysML models were constructed using SysBuilder software developed by Suzhou Tongyuan Soft Control Information Technology Co., Ltd. This tool serves as a system architecture design environment for complex engineering systems, adhering to the SysML specification. By taking user requirements as inputs, SysBuilder employs graphical, structured, and object-oriented methodologies to comprehensively support system development phases, including requirement modeling, functional analysis, architectural design, and verification/validation processes.

The three key elements of MBSE forward design are: modeling tools, modeling languages, and methodology. Among these, methodology, as the core of system architecture design, is the most critical. Based on the traditional MBSE framework (Requirements, Behavior, Structure, Parameters) and combined with the multi - level complex structural characteristics of the steam power system, this paper develops a matrix - based system architecture design method. (as shown in Figure 1). The entire design process is decomposed into three levels: system, subsystem, and equipment, with specific design content divided into four parts: requirements, behavior/function, structure, and parameters.

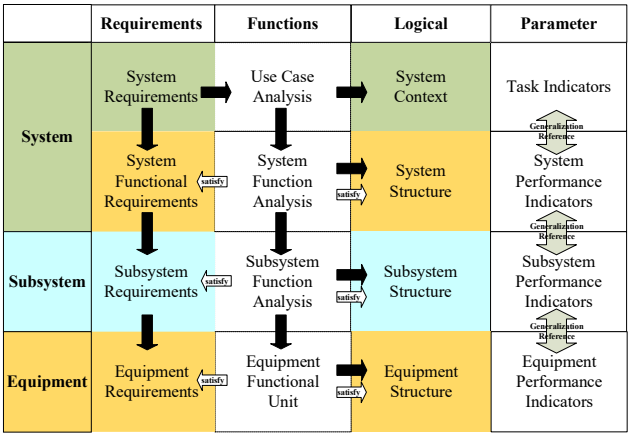


Figure 1. Matrix methodology for steam power system architecture design.

The architecture design of a steam power system starts from the overall requirements articulated by stakeholders

DOI 10.3384/ecp218727

and can be divided into two parts as inputs to guide the design: one part is functional requirements that describe the system's functionalities, while the other part is non-functional requirements that describe system performance constraints. Functional requirements are translated into the system's external capabilities, states or behaviors, while non-functional requirements are transformed into constrained indicators used for checking and validating system performance. Based on these principles, the functional activities and hierarchical structure of the system level (steam power system), subsystem level (steam system, condensate system, cooling water system, low-pressure steam system, sealing and extraction system, lubrication system) and equipment level (turbine, feedwater pump, condenser, vacuum pump, oil cooler, deaerator, etc.) are analyzed. Requirements are traced throughout the process to ensure that the overall requirements articulated by stakeholders are fully covered, and performance indicators are decomposed and verified through layers of indicators assignment and calculation as design constraints. As shown in Figure 1, black one-way arrows illustrate the process flow of system design, white single-direction arrows represent tracing and verification of requirements, and green two-way arrows represent decomposition and validation of performance indicators.

Based on the steam power system architecture design methodology described above and using the SysML modeling language, this paper models the steam power system using the MWORKS system architecture design software. As shown in Figure 2, this is the logical architecture of the steam power system, which reflects the main components of the steam power system and the logical interactions between them. The differently colored lines represent different object flows.

Further analysis of the system hierarchy architecture, each subsystem and equipment are designed until meeting the requirements of the model granularity. As shown in Figure 5, this is the subsystem architecture of the steam power system, including the cooling water system, lubrication system, low-pressure steam system, sealing and extraction system, steam system, and condensate system. Finally, each equipment and subsystem are simplified and assembled from bottom to top to obtain the overall architecture of the steam power system.

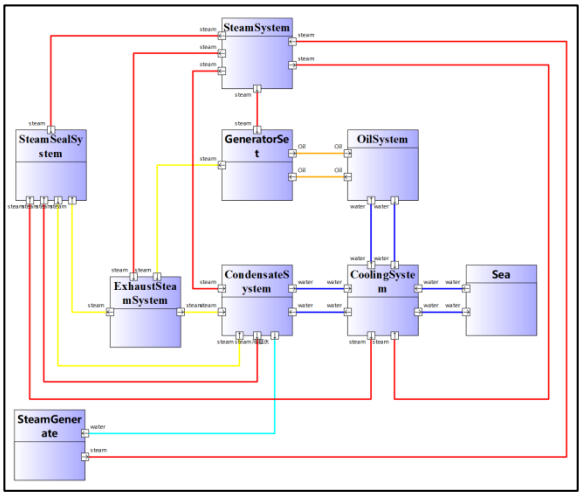


Figure 2. Steam power system architecture

Building upon the steam power system architecture, this study employs the automated SysML-to-Modelica conversion interface within the MWORKS platform to transform architectural models into simulation-ready models. This conversion mechanism operates on the principle of XMI (XML Metadata Interchange)-based model-to-library mapping. Specifically, it parses XMI files generated from SysML to extract critical metadata, including:

- Model organizational hierarchy (XMI, Model, Package)
- Component definitions (ownedAttribute)
- Interface connections (ownedConnector, End)
- Inter-object relational topology

The process extends SysPhs semantics to comprehensively convert parametric constraints, connection dependencies, and governing equations embedded in the design model. Through a GUI-driven workflow, engineers can:

- Select framework templates for physical model encapsulation
- Configure mapping rulesets (type matching, interface binding)
- Generate executable Modelica standalone models or reusable component libraries

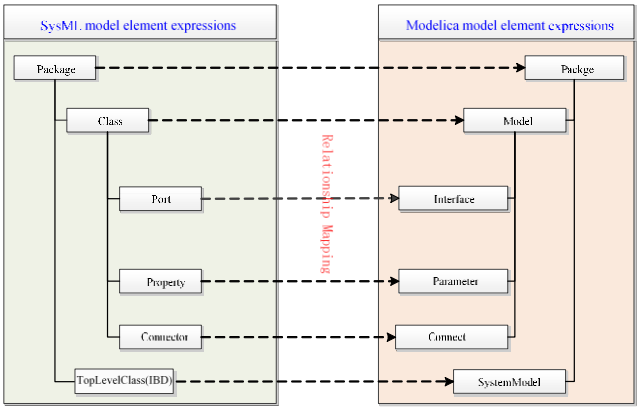


Figure 3 Mapping relationship between SysML model and Modelica model

The platform further provides co-simulation capabilities for iterative debugging and validation of converted models, ensuring behavioral consistency with original architectural specifications[6]. As shown in Figure 4, this is the architecture of the steam power system simulation model.

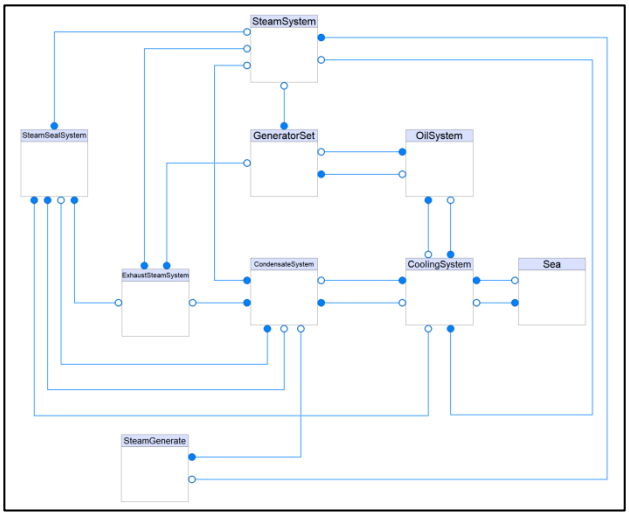
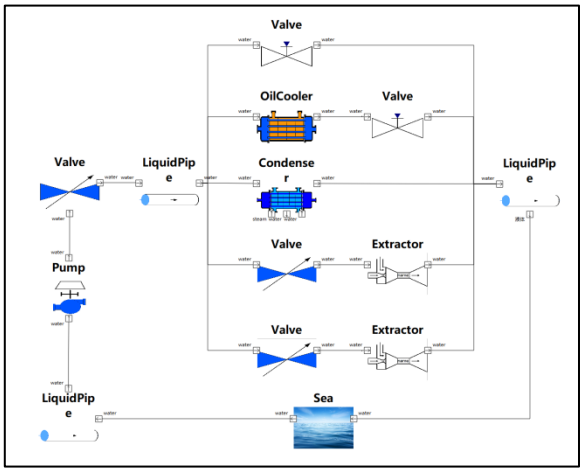
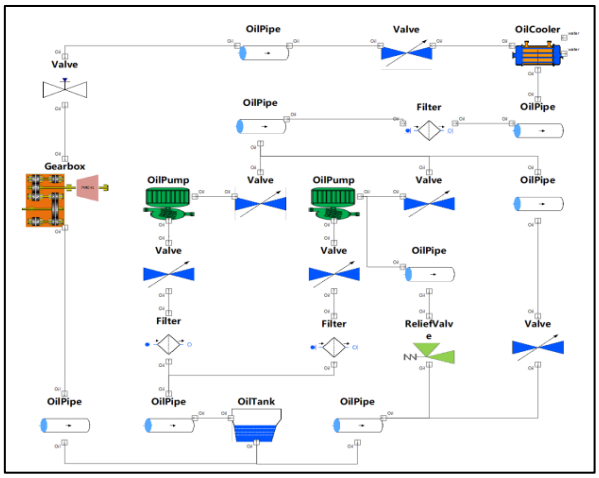


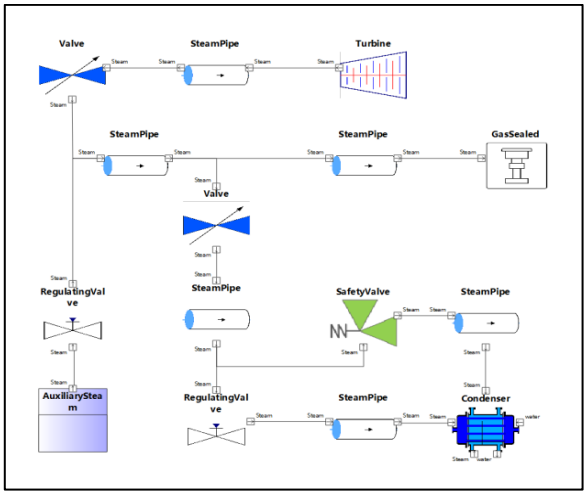
Figure 4 Simulation model architecture for steam power systems



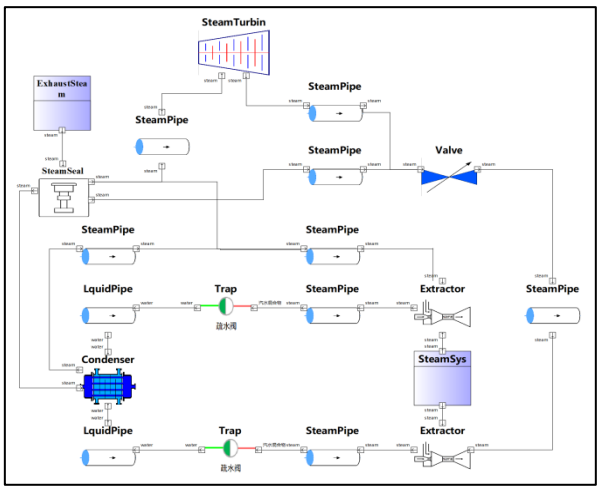
(a) Cooling System



(b) Oil System

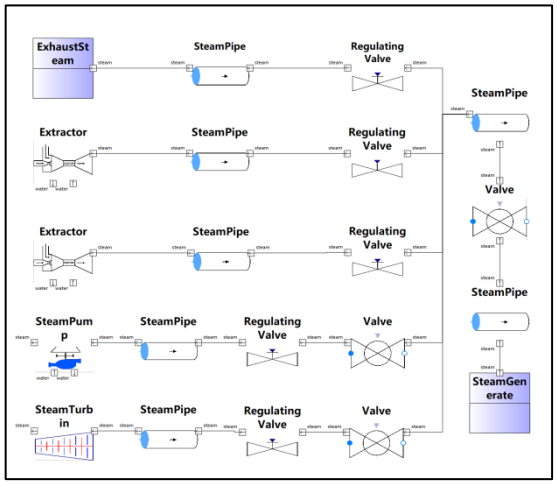


(c) Exhaust-Steam System

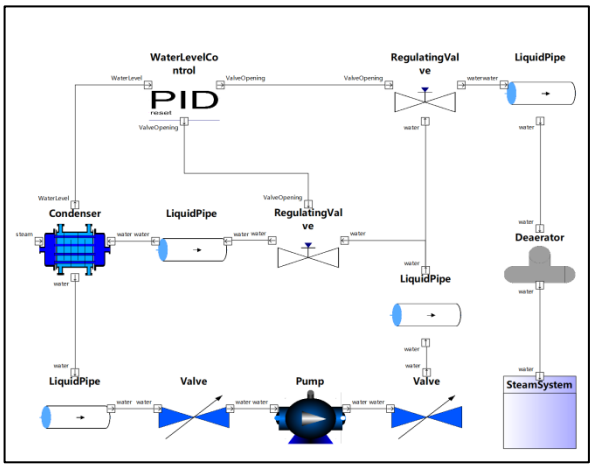


(d) Steam-Seal System

DOI 10.3384/ecp218727



(e) Steam System



(f) Condensate System

Figure 5 Architecture of steam power system subsystem

2.2 Steam Power System Simulation Model Implementation

Modelica is a multi-domain physical system modeling language that uses non-causal and declarative modeling techniques and supports almost all engineering disciplines. The language follows object-oriented programming principles and employs a hierarchical mechanism, component connection mechanism, and inheritance mechanism for model construction, thus simplifying the complexity of the model structure. Unlike Simulink modeling, where mathematical models need to be derived as definite causal relationships with fixed data flow directions, Modelica defines the behavior of variables through equations without artificial division of equation-solving directions. The solver solves unknown variables based on the given conditions. The language models the real physical topology of the system and the component view has a similar structure hierarchy and layout as the real system. Its built-in Modelica standard library includes basic component models in multiple domains, such as mechanical, electrical, hydraulic, thermal, and control, enabling comprehensive performance analysis and evaluation of modern complex engineering systems[7].

Currently, Suzhou Tongyuan Soft-Control Information Technology Co., Ltd. have jointly launched the MWORKS.Sysplorer platform based on the Modelica language for system modeling and simulation. The platform is a modeling and simulation environment for multi-domain industrial products. It fully supports the unified multi-domain modeling standard Modelica. It organizes models hierarchically based on the actual physical topology of products. It also supports various visual modeling methods, including physical, block diagram, and state machine modeling. Sysplorer offers embedded code generation, and supports the integration of design, simulation, and optimization. It is internationally recognized as advanced general-purpose software for system modeling and simulation. For this research, the MWORKS.Sysplorer platform based on the Modelica language is adopted for the construction of the steam system simulation model.

2.2.1 Component Model Construction

Based on the steam power system architecture, a steam power system simulation model library is constructed using the Modelica multi-domain unified modeling specification. The steam power system mainly includes equipment such as steam turbines, condenser, pumps, and valves. Below are the physical principles of some of the main components and equipment models.

a) Steam turbine

The main function of a steam turbine model is to calculate the power output generated by high-temperature and high-pressure steam through expansion. In this article, a mathematical model of the steam turbine is established based on the Rankine cycle formula[8]. The steam flow rate at the outlet of the steam turbine is given by:

$$\frac{G_E}{G_{E0}} = \sqrt{\frac{p_E^2 - p_L^2}{p_{E0}^2 - p_{L0}^2}} \sqrt{\frac{T_{E0}}{T_E}} \tag{1}$$

Steam outlet specific enthalpy:

$$h_L = h_E - \Delta h_i \eta_i \tag{2}$$

Turbine power:

$$P_t = G_E(h_E - h_L) \tag{3}$$

Among them, G_{E0} represents the rated steam flow rate, p_{E0} represents the rated steam inlet pressure, T_{E0} represents the rated steam inlet temperature, p_{L0} represents the rated steam outlet pressure, p_L represents the actual steam outlet pressure.

b) Condenser

The condenser is a common multi-inlet shell and tube heat exchanger in the thermal system. The cooling water flows through the pipe side by the circulating pump, and the steam passes over the outside of the tube in the shell side for condensation and heat release. The heat is transferred to the cooling water through the tube wall, and the lower temperature cooling water keeps the condenser high vacuum through continuous circulation.

Shell side steam mass conservation equation：

$$\frac{d\mathrm{m}_g}{dt} = \mathrm{q}_{m,in} + \sum q_{m,in}^{OT} + q_{m,HW} - q_{m,p} - q_{m,con} \quad (4)$$

In this equation, m_g represents the steam mass within the condenser, $q_{m,in}$ denotes the steam mass flow rate of exhaust steam from the turbine , $q_{m,HW}$ indicates the steam mass flow rate entering the steam space through the heat well , $q_{m,p}$ is the steam mass flow rate extracted by the extraction steam trap, $q_{m,con}$ refers to the condensate flow rate at the steam space outlet of the condenser.

Shell side steam energy conservation equation:

$$\frac{d(\mathrm{m}_g h_g)}{dt} = q_{m,in}h_{in} + \sum (q_{m,in}^{OT} h_{in}^{OT}) - q_{m,p}h_p - q_{m,con}h_{con} \quad (5)$$

Here, h_g denotes the enthalpy of steam in the condenser, h_{in} represents the enthalpy of exhaust steam from the turbine, h_p indicates the extracted steam enthalpy from the extraction steam trap, h_{con} is the enthalpy of condensate at the steam space outlet of the condenser.

Tube side thermal balance equation：

$$Q = KA\Delta T_m = q_{m,W}(h_{W,L} - h_{W,\mathrm{E}}) = q_{m,G}(h_G - h_C) \quad (6)$$

Where Q represents condensation heat transfer, K denotes the total heat transfer coefficient , A is the condensation heat transfer area, ΔT_m is the logarithmic mean temperature difference, q_m is the cooling water flow rate, $q_{m,G}$ is the exhaust steam flow rate, $h_{W,E}$ is the specific enthalpy of cooling water at inlet , $h_{W,L}$ is the outlet temperature of cooling water, h_G is the specific enthalpy of exhaust steam and h_C is the specific enthalpy of condensate.

The overall heat transfer coefficient K is calculated as：

$$K = 794 \times 4186/3600\beta\sqrt{v_W} \cdot \sqrt[4]{\frac{T_{W,L} + T_{W,E}}{2}} + 17.8 \quad (7)$$

Where β is the fouling coefficient, v_W is the cooling water flow velocity, $T_{W,L}$ is the inlet temperature of cooling water, and $T_{W,E}$ is the outlet temperature of cooling water .

c) Centrifugal pump

The pump model is used to convert mechanical energy into fluid kinetic and potential energy, thereby increasing fluid pressure. Pressure of centrifugal pump:

$$\frac{H}{H_0} = \left(\frac{n}{n_0}\right)^2 \quad (8)$$

Centrifugal pump mass flow rate:

$$\frac{q_{m,z}}{q_{m,0}} = \frac{q_m}{q_{m,0}} \cdot \frac{n_0}{n} \quad (9)$$

Centrifugal pump powe:

$$\frac{q_{m,z}}{q_{m,0}} = \frac{q_m}{q_{m,0}} \cdot \frac{n_0}{n} \quad (10)$$

Centrifugal pump powe:

$$P_p = \frac{\Delta p_p q_{mp}}{\rho \eta_p} \quad (11)$$

$$\Delta p_p = p_L - p_E = H \quad (12)$$

Outlet temperature of centrifugal pump:

$$T_L - T_E = \frac{(1-\eta_0)}{\eta_0} \cdot \frac{H_0}{\rho C_p} \quad (13)$$

Centrifugal pump torque:

$$\tau_p = \frac{30P_p}{\pi n} \quad (14)$$

Among them, H is the actual pressure head, H_0 is the design point pressure head, q_m is the actual flow rate, q_{m0} is the design point flow rate, q_{mz} is the flow rate converted from the actual flow rate to the design point speed, n is the actual speed, n_0 is the design point speed.

d) Steam regulating valve

Steam control valve is used to control the steam flow rate entering the steam turbine, thereby controlling the output power of the steam turbine[9]. Assuming that the steam flows adiabatically and isoentropically in the valve, only the flow rate produced by the pressure difference between the inlet and outlet is considered. The momentum equation is:

$$q_{m_E} = k_{m_x} k_v \left(1 - 0.367\frac{X}{X_{cr}}\right)\sqrt{\rho_E p_E X} \quad (15)$$

$$X_{cr} = 1 - \left(\frac{2}{K+1}\right)^{\frac{K}{K-1}} \quad (16)$$

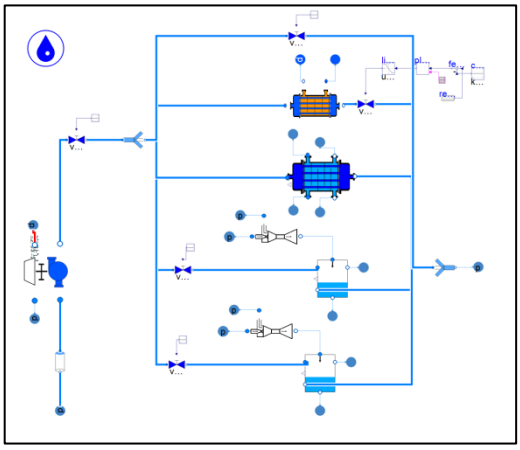
$$k_{mx} = \sqrt{\frac{2KA_{max}^2}{K+1}} \quad (17)$$

q_{m_E} is the steam mass flow, ρ_E is the inlet steam density, p_E is the inlet steam pressure,X is the valve pressure loss ratio, X_{cr} is the critical pressure loss ratio, K is the heat capacity ratio, k_{mx} is the valve characteristic correction coefficient, and A_{max} and is the maximum opening flow area of the valve.

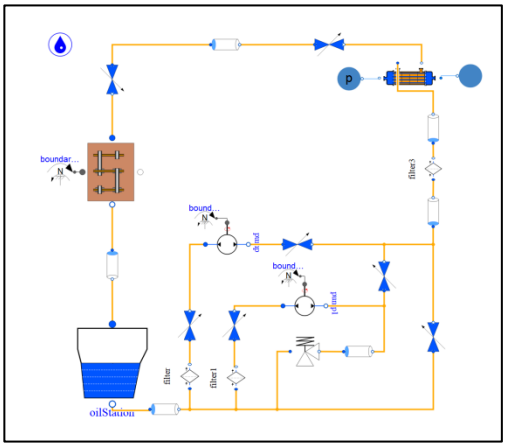
2.2.2 System Model Construction

Based on the general model library of steam power systems, equipment, subsystem, and system models are built from bottom to top according to the system design architecture. Figure 6 shows the sub-system simulation model of the steam power system. The cooling water system includes a lubricating oil temperature control module, and the condensate system includes a condenser water level control module.

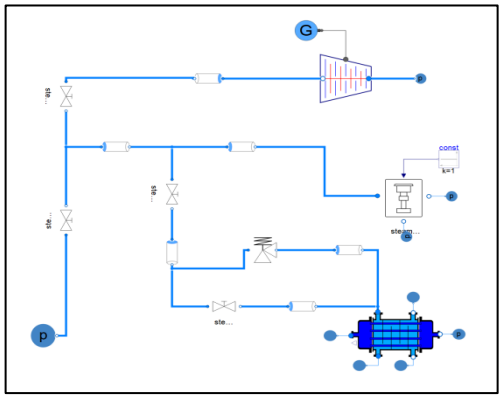
DOI 10.3384/ecp218727



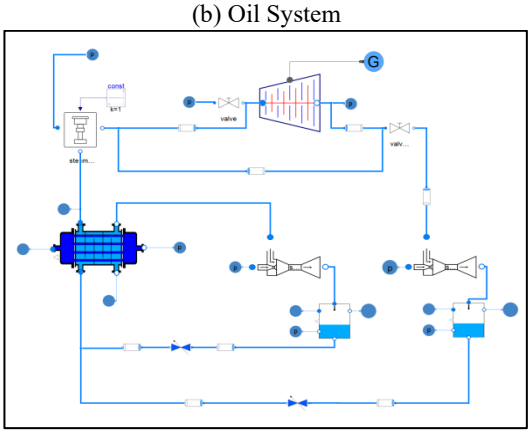
(a) Cooling System



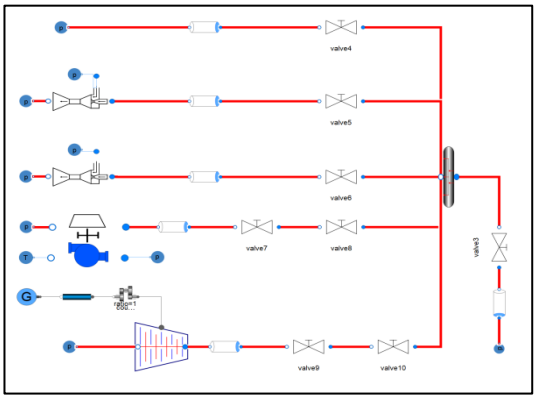
(b) Oil System



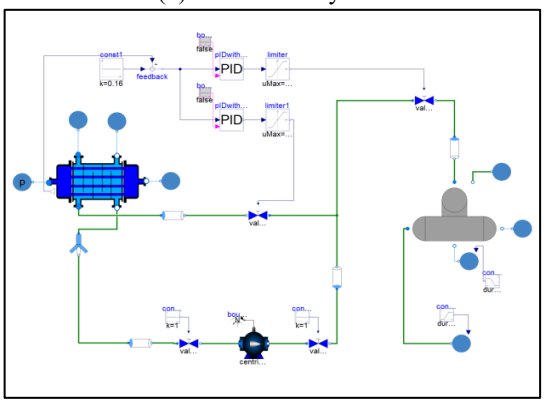
(c) Exhaust-Steam System



(d) Steam-Seal System



(e) Steam Ststem



(f) Condensate System

Figure 6 Simulation model of steam power system subsystem

After completing the construction of subsystem simulation models, they are connected according to the interface relationships between the subsystems of the steam power system based on its logical architecture. Finally, the steam power system simulation model is obtained (shown in Figure 7). The steam power system consists of a steam system, condensate system, low-pressure steam system, cooling water system, seal system and lubrication oil system. Its main equipment includes steam turbine, boiler feed pump, condenser, condensate pump, ejector, deaerator, lubricating oil cooler, lubricating oil pump, valves, and pipeline accessories. The control logic in the system includes condenser water level control and lubrication oil temperature control.

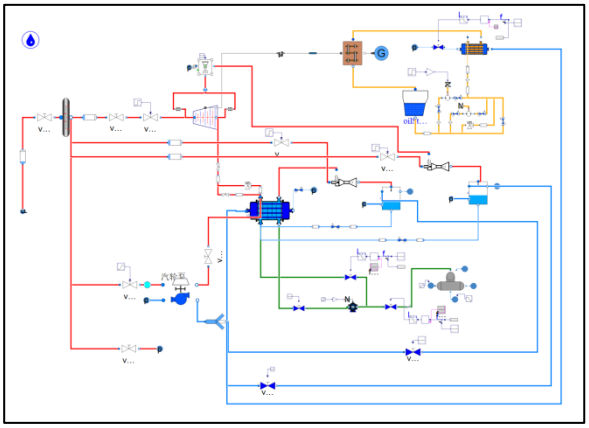


Figure 7 Simulation model of steam power system

3 System dynamic scene analysis and indicator validation

3.1 Dynamic scene analysis

In order to describe the dynamic scene of a steam power system, you can select the functional activity model in SysML language from the steam power system architecture design model library. By defining the dynamic operational scene of the steam power system, an activity diagram can be formed that can be used for simulation and verification. This achieves the integration of system architecture design and simulation verification.

In the activity diagram that describes the dynamic scene of the steam power system, functional activities are used to express the control behavior of the simulation model, such as the "start lubrication pump" functional activity. Based on the programming of the "start lubrication pump" behavior through architecture design software functionality, a design and simulation integrated script is defined, and a joint simulation interface for creating simulation models is established. Through UDP communication, the instruction signals for starting/stopping the "lubrication pump" device in the control simulation model are sent to the system simulation model, thus achieving dynamic verification of the steam power system operating scenario based on the simulation model. Figure 8 shows the activity diagram of the dynamic scene of the steam power system. In the activity diagram, functional activities for the steam power system are defined, including simulation preparation, system backup, start lubrication pump, start cooling water pump, start condensate pump, start air extractor, unit start-up test, lubrication temperature regulation, and test selection. The activities are connected by control flows according to the execution logic of the steam power system (represented by dashed lines with arrows in Figure 8), and the arrows indicate the sequence of activity execution.

Meanwhile, the steam power system simulation model needs to set corresponding control interfaces and

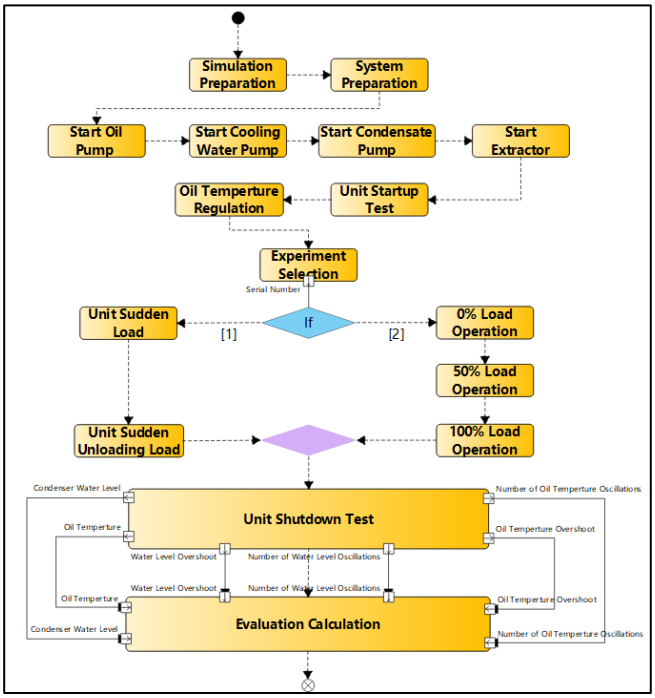
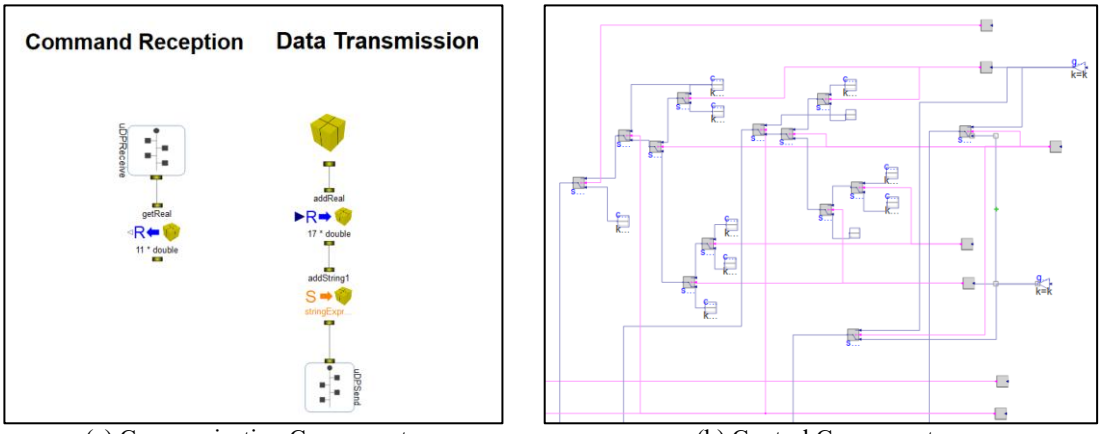


Figure 8 Steam Power System Dynamic Scene Activity Diagram

strategies to meet the evolving activity flow of the system's dynamic scene. Therefore, based on the previously constructed steam power system simulation model, a simulation model control module needs to be further established. The control instructions corresponding to different functional activities and the activity transition conditions are received and sent through the UDP communication component. When the activity diagram starts the simulation, the activity nodes execute according to the control flow of the activity diagram. Different activity nodes send different operation instructions to the simulation model. The simulation model feeds back the calculation results of the key parameters of the model to the activity diagram in real time. After the calculation parameters of the simulation model meet the transfer conditions of the corresponding activity, the activity diagram executes the next activity node. Figure 9 shows the communication component and control module of the system simulation model. The communication component is used to receive the control instructions in the activity diagram and the feedback signals of completed activities. The control module passes the control instructions received by the communication component to the corresponding component or device of the simulation model to achieve a dynamically simulated system based on the activity diagram.

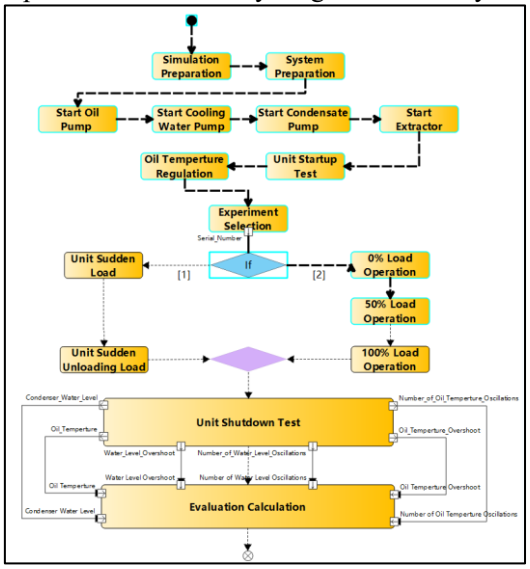
DOI 10.3384/ecp218727



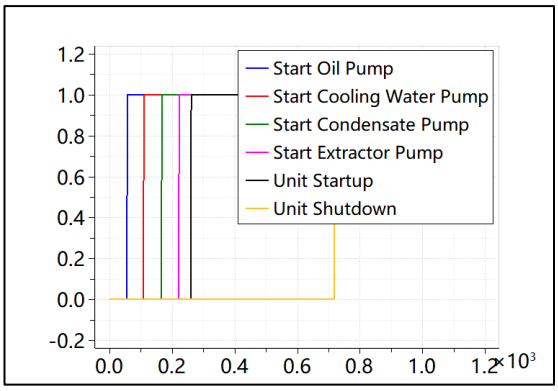
(a) Communication Components (b) Control Components

Figure 9 Communication component and control module of system simulation model

After debugging the communication and control module of the simulation model according to the execution logic of the system's dynamic scene activity diagram, the simulation calculation of the activity diagram can be performed. Figure 10 shows the schematic diagram of the activity execution status in the joint simulation process of the activity diagram and the system model, and the status of the system model instruction receiving. In the activity diagram, the current activity in the execution state is highlighted by the activity model. At the same time, when the corresponding control instructions are received by the system simulation model, the curve value of the corresponding instruction changes to 1.

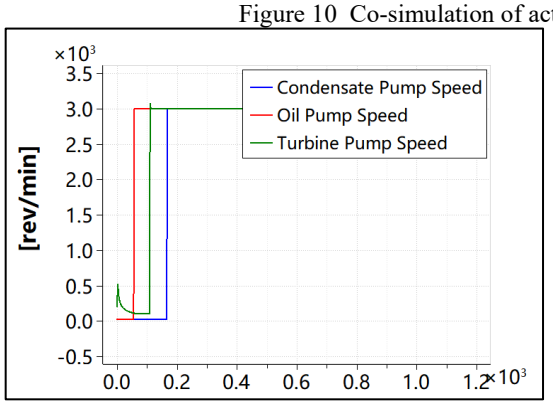


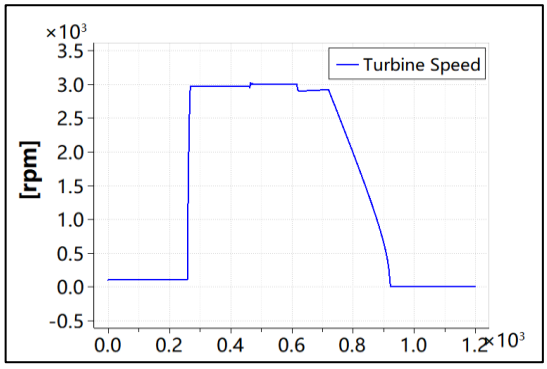
(a) Activity Diagram Simulation



(b) Command Reception

Figure 10 Co-simulation of activity diagram and system model





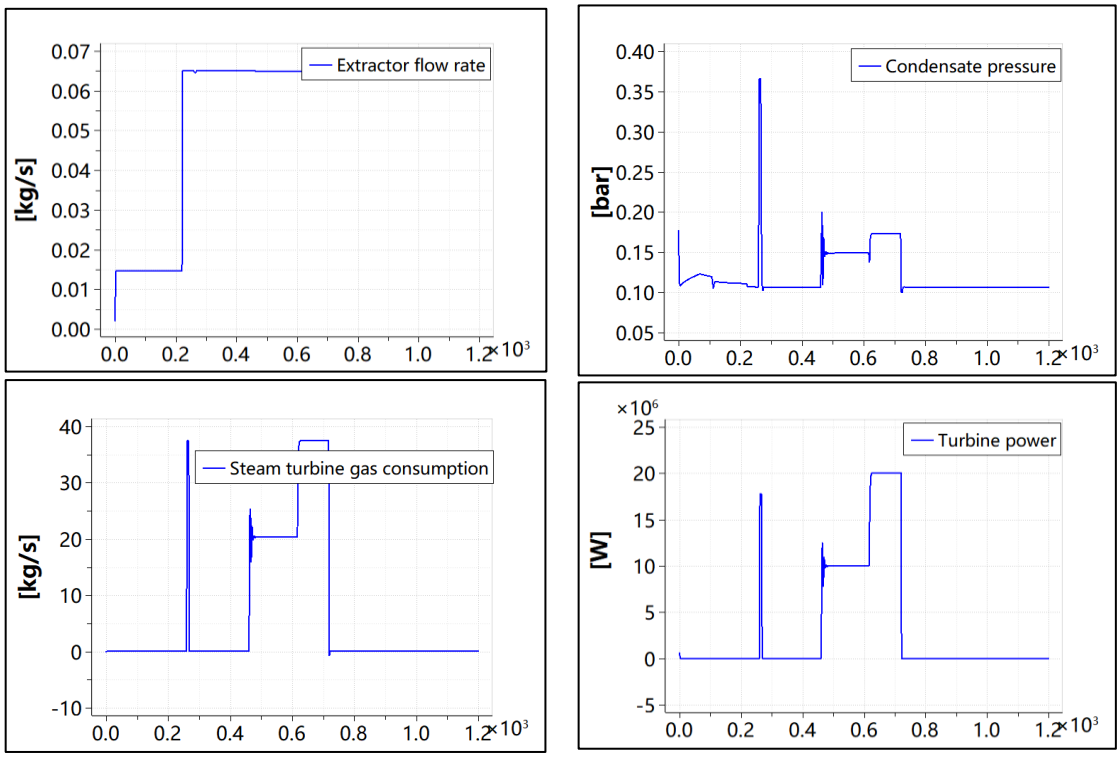


Figure 11 Key equipment parameters of the system simulation mode

Figure 11 shows the critical equipment parameter curves of the system simulation model, which primarily include the "lubricating oil pump speed," "condensate pump speed," "turbine circulation pump speed," "turbine speed," "exhauster extraction flow rate," and "condenser pressure." These curves are analyzed in comparison with the control command changes shown in Figure 10.

When the simulation model's "lubricating oil pump" receives the "start lubricating oil pump" command from the activity diagram, the lubricating oil pump speed rapidly increases to 3000 r/min. Similarly, at approximately 275 seconds, the "turbine unit" receives a "start-up" signal, causing the steam control valve to open and the steam flow to surge instantly. The turbine speed rapidly increases from a low standby speed to 3000 r/min. After the speed stabilizes, the valve is throttled to maintain 3000 r/min. During the start-up phase, the sudden increase in steam flow causes the condenser pressure to rise briefly before quickly decreasing.

At approximately 750 seconds, the "turbine unit" receives a "shut-down" signal, and the steam control valve closes. The steam flow into the condenser drops rapidly, leading to a decrease in condenser pressure. The turbine speed gradually decreases from the rated speed to 0 r/min.

3.2 System indicator validation

In the previous section, the steam power system simulation model executed dynamic calculations in sequence according to the activity diagram scenarios, accurately obtaining dynamic operating parameters for the system and equipment. These state parameters are fed back to the system architecture design software via UDP communication components, enabling comprehensive verification and validation of the system architecture design performance metrics.

This paper uses the verification of performance metrics for condenser water level and lubricating oil temperature control in the steam power system as an example. The steam system imposes strict performance requirements on water level control in the condensate system and lubricating oil temperature control in the lubricating oil system, specifically regarding overshoot and oscillation count. During dynamic operation, these parameters must meet the limits for overshoot and oscillation count. As shown in Figure 12, the overall performance metrics for the control system specify a maximum condenser pressure of 1 bar, with a maximum allowable oscillation count of 5 for water level; for lubricating oil temperature control, the maximum allowable overshoot is 0.3, and the maximum allowable oscillation count is also 5.

需求属性

参数名称	值	单位
Maximum condenser pressure	1	bar
Maxmum_Number_of_Water_Level_Oscillations	5	1
Maxmum_Oil_Temperture_Overshoot	0.3	1
Maxmum_Number_of_Oil_Temperture_Oscillations	5	1

Figure 12 Control system performance indicators

As shown in Figure 13, the curves represent the computed performance metrics of the simulation model.

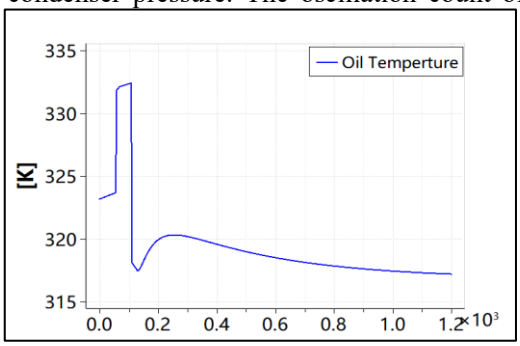
Figure 13(a-c) show the lubricating oil temperature variation curve, the oscillation count of lubricating oil temperature, and the overshoot of lubricating oil temperature. The oscillation count refers to the number of times the temperature crosses the target value under control regulation, totaling 2 times, while the overshoot is

DOI
10.3384/ecp218727

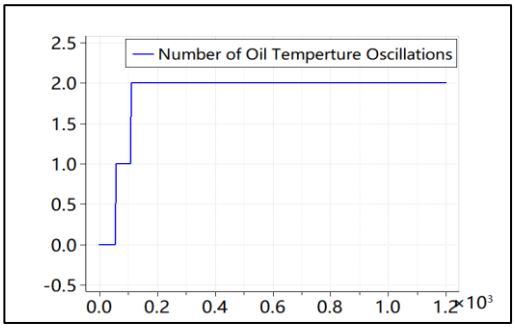
the percentage by which the maximum actual temperature exceeds the target, at 0.054%.

Figure 13(d-f) depict the oscillation count of the condenser water level, the condenser pressure, and the maximum condenser pressure. The oscillation count of the water level is the number of times it crosses the target value under control regulation, totaling 3 times.

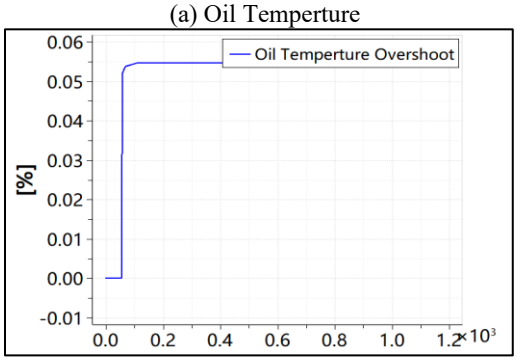
Figure 13(g) shows the state parameters received by the system architecture design software from the simulation model's feedback.



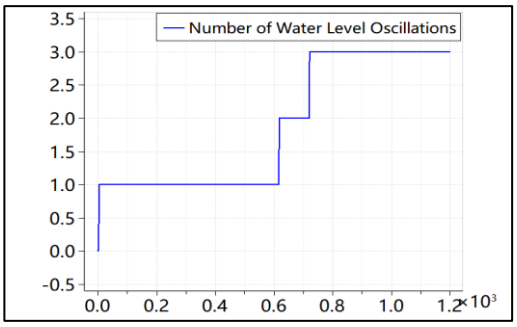
(a) Oil Temperture



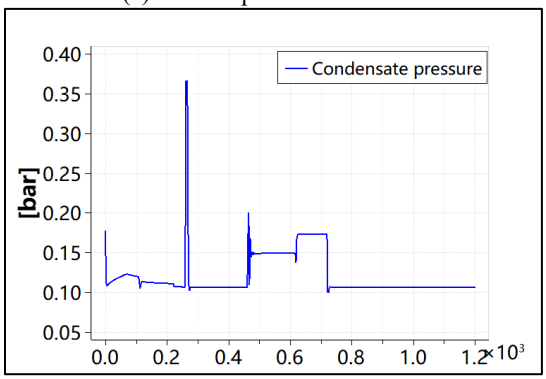
(b) Number of Oil Temperture Oscillations



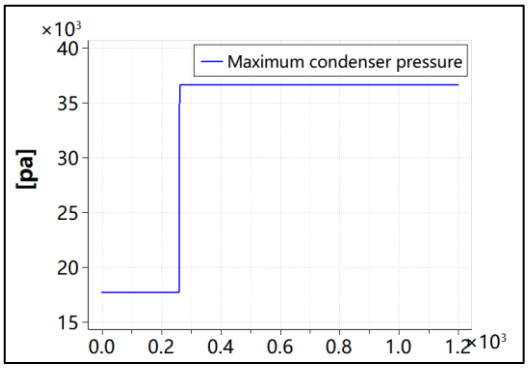
(c) Oil Temperture Overshoot



(d) Number of Water Level Oscillations



(e) Condensate pressure



(f) Maximum_condenser_pressure

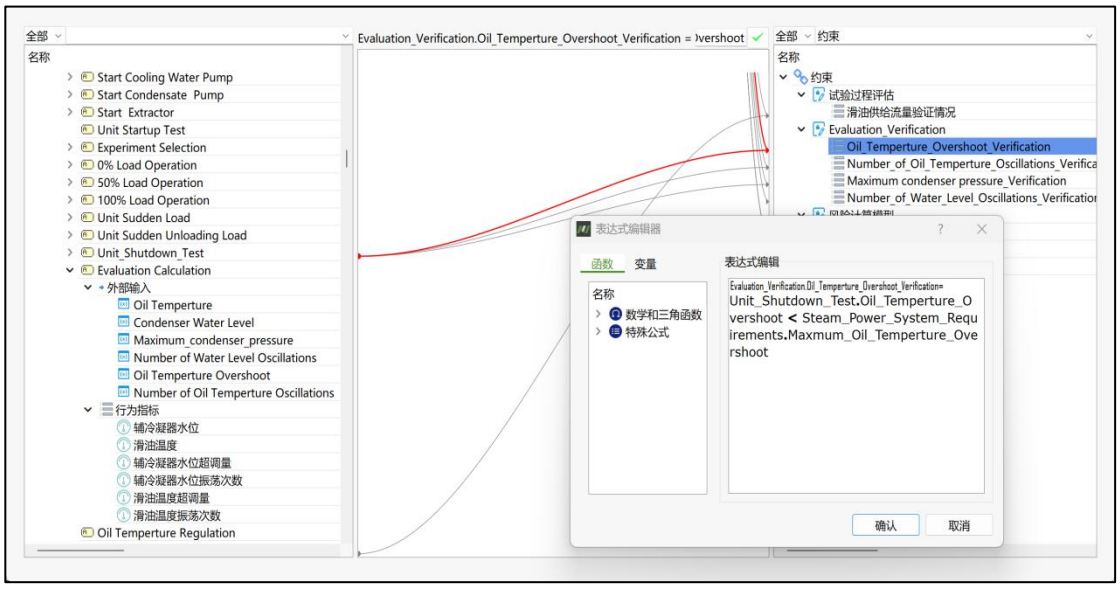
参数名称	值	单位
Condenser_Water_Level	0.384663	m
Oil_Temperture	313.148	K
Maximum_condenser_pressure	0.3662109	1
Number_of_Water_Level_Oscillations	3	1
Oil_Temperture_Overshoot	0.0551731	1
Number_of_Oil_Temperture_Oscillations	2	1

(g) Fedback Data of Simulation Model

Figure 13 Calculated parameters of the control system

To further validate whether the system performance metrics are met and determine the feasibility of the system architecture design, metric constraints can be established based on the overall performance metrics from the requirements model and the state parameters returned by the simulation calculations. Figure 14 illustrates the performance metric validation process. Figure 14(a) shows the constraint relationship between the performance metrics and the state parameters. Here, the actual maximum overshoot of the lubricating oil temperature is calculated and assigned to the constraint model value property. Figure 14(b) presents the results of all metric constraint validations. A metric constraint value of 1 indicates that the performance metric is satisfied. As shown in the figure, the actual maximum overshoot of the lubricating oil temperature (0.054%) is less than the metric value (30%), the actual oscillation count of the lubricating oil temperature (2 times) is less than the metric value (5 times), the actual maximum pressure of the condenser (0.366 bar) is less than the metric value (1 bar),

and the actual oscillation count of the condenser water level (3 times) is less than the metric value (5 times). All performance metrics are thus satisfied.



(a) Constraint Relationship Construction

约束要求

参数名称	值	约束要求	满足情况
Oil_Temperture_Overshoot_Verification	1	Unit_Shut...	满足
Number_of_Oil_Temperture_Oscillations_Verification	1	Steam_Po...	满足
Maximum condenser pressure_Verification	1	Steam_Po...	满足
Number_of_Water_Level_Oscillations_Verification	1	Steam_Po...	满足

(b) Verification Results

Figure 14 Performance indicator verification

4 Conclusion

Based on the model-based systems engineering approach, this paper has completed the architecture design and system simulation verification of a steam power system, realizing a closed-loop process of design-simulation-verification. The conclusions are as follows:

1) The paper innovatively proposes a closed-loop technical approach integrating steam power system design, simulation, and verification. This approach combines system design with simulation verification, overcoming the limitations of traditional separation and boosting design efficiency. Relying on Modelica's powerful functions for precise modeling and efficient simulation, it quickly identifies design issues, optimizes designs early, reduces iterations, cuts costs, and shortens development cycles, offering a new method for complex system design.
2) Based on the architectural model, this approach enables seamless transformation from system design architecture to system simulation architecture. By applying well-defined mapping rules, it accurately converts elements such as composition, interface, and connection relationships from SysML architectural models to Modelica simulation model architecture. This ensures consistency between design and simulation architectures in terms of interfaces and parameter passing. The process not only integrates design and simulation but also leverages Modelica's power in multi-domain physical system modeling, facilitating a smooth transition from conceptual design to performance verification.
3) Dynamic scenario simulation and verification of the steam power system are done via combined simulation of activity diagrams and system models. Using Modelica simulation technology with SysML-defined dynamic activity diagrams, it achieves precise dynamic simulation of the system's operation. This combined simulation not only reproduces the dynamic process but also provides reliable real-time state parameter feedback for performance metric verification, enhancing the closed-loop system of steam power system design, simulation, and verification.

References

[1] Lui Wang. Effort to Accelerate MBSE Adoption and usage at JSC[C]. SPACE Conference and Exposition 13-16 September 2016, Long Beach, California AIAA SPACE, 2016: 5542-5552.

DOI 10.3384/ecp218727

[2] Mitchell C, Steven W. Transitioning the SWFTS Program Combat System Product Family from Traditional Document‐Centric to Model‐Based Systems Engineering[J]. Systems Engineering, 2014, 17(3): 313-329.
[3] ZHANG Jian-hua, YANG Zhi-yi, SHEN Zhen-hui. Research and Application of Ship EW System Based on Model-Based Systems Engineering Method [J]. Command Control & Simulation,2015,37(03):101-104.
[4] Xinggang W, Shuai S H I, CHENG X, et al. Simulation of dynamic characteristics of marine steam power system under rapid power generation disturbance[J]. Chinese Journal of Ship Research, 2021, 16(Supp1): 1-6.
[5] Zhang Y, Hoepfner G, Berroth J, et al. Towards holistic system models including domain-specific simulation models based on SysML[J]. Systems, 2021, 9(4): 76.
[6] Hinkelman K, Wang J, Zuo W, et al. Modelica-based modeling and simulation of district cooling systems: A case study[J]. Applied Energy, 2022, 311: 118654.
[7] Nigischer C, Bougain S, Riegler R, et al. Multi-domain simulation utilizing SysML: state of the art and future perspectives[J]. Procedia CIRP, 2021, 100: 319-324.
[8] Feng H, Chen L, Tang W, et al. Optimal design of a dual-pressure steam turbine for rankine cycle based on constructal theory[J]. Energies, 2022, 15(13): 4854.
[9] Pondini M, Signorini A, Colla V. Steam turbine control valve and actuation system modeling for dynamics analysis[J]. Energy Procedia, 2017, 105: 1651-1656.

DOI
10.3384/ecp218727

The Fundamental Modeling Practices and Specifications to support the Preservation and Reuse of Analytical Simulations

Mark Williams[1] Hubertus Tummescheit[2] Ajaykumar MST[3]
Jose María Alvarez-Rodríguez[4]

[1]International Council of System Engineering, USA,
mark.williams@incose.net

[2]Model Based Innovation LLC, USA,
hubertus.tummescheit@modelbased.cloud

[3]Boeing India Private Limited, India,
ajaykumar.mst@boeing.com

[4]Carlos III University of Madrid, Spain
josemaria.alvarez@uc3m.es

Abstract

Significant opportunities exist to preserve and reuse simulation and analytical models and data. Based on the capabilities developed by the Modelica Association (MA), LOTAR International, and other contributing tool developers, multiple engineering and manufacturing industries can compile extensive archives of reusable and interoperable performance, behavior and integrated product information. Through the development and implementation of a preservation plan, the use of compliant off-the shelf software applications, and packaging using compatible data standards, a repository of analytical interactions, simulations and functional prototypes can be archived, maintained, and resourced by future users. The scope of this paper is the exploration of alternative data standards that might support LOTAR requirements. This paper identifies progress made since the publication of (Coïc 2021, Coïc 2023) with respect to maturity of the "Long Term Archiving and Retrieval" (LOTAR) draft standards – ASD-STAN 9300 series, (ASD-STAN, 2025) and accompanying prototype implementations. There has also been significant progress by the Modelica Association with the introduction of layered standards in new versions of the FMI and SSP standards (Modelica Association 2024-11., Modelica Association 2024-12). The soon to be released layered standard SSP Traceability (Modelica Association 2025-04) provides mechanisms to document and preserve relevant metadata for model archiving in support of LOTAR. The FMI layered standard FMI-LS-Ref standardizes parameter sets and reference results, which are an important subset of the LOTAR International requirements.

Keywords: Model reuse, SSP, FMI, LOTAR, Traceability, Modelica

1. Introduction

Model-Based Systems Engineering (MBSE) has become the de facto standard for developing complex systems, where models are first-class members of the engineering process. Rather than relying on documents, MBSE encourages the use of descriptive and analytical models to support key lifecycle activities such as requirements definition, design, analysis, and verification. This model-centric approach is critical to enabling the Digital Thread—a connected and consistent flow of engineering information that supports collaboration across disciplines and lifecycle phases.

As digital engineering practices mature, simulations are essential to representing, verifying and validating complex Cyber-Physical Systems (CPSs). Analytical simulations capture domain-specific behaviors and performance characteristics. However, while these simulations are widely used during a product's development and manufacture, they are not commonly preserved in a reusable or interoperable format. Traditional Product Lifecycle Management (PLM) tools focus on managing static artifacts, configurations, and project milestones. However, they usually lack the features needed to manage alternative formats, expose and supplement metadata or maintain contextual links and dependencies needed for reuse across different lifecycle stages and related projects.

Recently, engineering has seen the definition of different metamodels that express the configuration and relevant information needed to share, exchange and execute analytical models. The Digital Data Package, DDP (Prostep ivip 2024) provides a structured container to bundle models, input/output data, documentation, and related metadata. This packaging approach enables traceability, validation, and long-term reuse across lifecycle phases. The analytical models contained in a DDP are further enhanced by using the Model Identify

Card, MIC-Core**,** (IRT SystemX 2020). The relationships with other models can be further elaborated by the ISO 1030-243 standard, known as AP 243, (ISO 2021) which serves as a formal metadata structure to capture the key characteristics of a model; such as its purpose, inputs, outputs, assumptions, limitations, and applicable domains. The AP 243 - MOSSEC standard defines a metamodel "*enabling the sharing and exchange of Modelling and Simulation contextual metadata"*. It acts as a technical "passport" for models, supporting collaboration and enhanced content identification, empowering engineers to assess a model's fitness for use without deep inspection of its internals. Note that the SRMD metadata format (Modelica Association 2025-3), defined in the SSP Traceability draft standard (Modelica Association 2025-1), can efficiently store metadata information from MIC-Core, MOSSEC, or any of the various other model metadata specifications currently in use.

1.1 LOTAR MBSE Standards - Overview

All forthcoming LOTAR MBSE standards are still in a draft format or under review for approval. It is, however, useful to present their overall structure here.

- **Part 500**: Fundamentals and Concepts for long term archiving and retrieval of Model-Based Systems Engineering information.
- **Part 510**: Long term archiving and retrieval of Requirements, and their management schema, as text, graphics, tables, models, or "parameter based" information.
- **Part 515**: Long term archiving and retrieval of Validation and Verification "text based" and "parameter based" information (expanding Part 510).
- **Part 520**: Long term archiving and retrieval of system or component level analytical behavior models described by specification or executable code, containing differential, algebraic and discrete equations. Includes causal or acausal models not addressed by the Part 600 series.
- **Part 530**: Long term archiving and retrieval of models that use system architecture descriptions and architecture description languages (ADLs) specified by ISO 42010/SAE AS5506 [9].
- **Part 540:** Long term archiving and retrieval of data and models specifying the Logical Bill of Materials (LBOM) (Design implementation requirements).
- **Part 550**: Long term archiving and retrieval of models or features describing digital or relational links. Methods for specifying highly integrated and interrelated models and elements across numerous software tool applications.

In preparation of defining, demonstrating, and publishing future preservation process standards, the LOTAR Model Based Systems Engineering Workgroup (LOTAR MBSE 2025) has introduced the application of the P510 and P515 metamodel (see above). Draft schemata are also available for P520 and P550, as well as a common core schema. The current state of Systems Engineering in virtual environments emphasizes the increasing need for early verification and validation of system specifications, often described as the stakeholder, functional, and technical requirements, and the necessity for configuration management and traceability of all related design data. The preservation of this information will be described in the P515 process standard and will be supported by preserving the data traceability features utilizing the P550 standard for linking information. As systems become more complex and interconnected, ensuring that specifications align with both system behavior and stakeholder needs throughout the development lifecycle represents a cornerstone of the MBSE process. This paradigm shift highlights the importance of good practices for model-based approaches that can facilitate the continuous tracking and validation of requirements, their allocation to design features, and model specific test results, ensuring that every decision and change is captured and verifiable at all stages of the system's lifecycle. Note that the ISO 26262-Standard (ISO 2028) for automotive functional safety has very similar traceability requirements and often needs the same type of behavioral models for verification and validation.

In this paper, we introduce the modeling practices and specifications needed to preserve and reuse analytical simulations effectively. This includes defining standard ways to document simulation intent, manage contextual metadata, and ensure consistency across tools and formats. By formalizing these aspects, we can move toward an executable and dynamic engineering environment, where simulations are not isolated efforts but integrated assets within a sustainable and reusable digital ecosystem.

Note the significant overlap of these purposes with finished and ongoing standardization efforts and process practices within the Modelica Association:

- **Part 515**: Part 515 aims at formalizing the informal process that is currently used by the Modelica Library project of verification and validation of the Modelica Standard Library. Similar methods are also used by commercial Modelica vendors. The ongoing effort of standardizing the capture of modeling results and parameter sets in FMI-LS-REF covers the same end user needs. In the ongoing prototype work, the proposed P515 schema was merged with the draft FMI-LS-REF schema for demonstration purposes, but developing the framework for a common schema would be preferable.
- **Part 520**: Part 520 addresses the core topics of Modelica Association standards, notably the behavior and simulation models. Both Modelica and

DOI 10.3384/ecp218741

FMI generally cover the P520 model type referenced here. However, the potential future role of the SSP-standard will need to be evaluated. Until recently, the P520 has focused on FMI mostly due to the much broader adoption of FMI by the software tool vendors. Also note, to avoid repeated translations, that the long-term preservation benefits represented by the LOTAR process are potentially easier to achieve with native Modelica models than with proprietary code repackaged into reusable FMUs. Nevertheless, using FMI as a preservation format will still accommodate many of the models developed using software applications that are alternatives to Modelica.

- **Part 530**: Through the use of Modelica components and acausal connectors, the SSP-standard utilizes a different philosophy than the ISO/SAE ADLs and is potentially useful as a substitute representation of a logical or functional design architecture. An execution oriented SSP system model would be easier to verify. Taking this view, the combination of Part 530 and Part 520 form a similar hierarchical and practical relationship as the SSP and FMI standards.
- **Part 550**: Combining the functionality of the SSP container format with the features of SSP Traceability exposes many of the characteristics defined by the P550 process. Together, they establish and specify the need for relational links between the models and their internal artefacts. They are both successful at formalizing the model integration results and providing auditable evidence of design traceability.

It therefore seems obvious that the Modelica Association and the LOTAR MBSE working group would recognize the significant benefits of developing a closer collaboration relationship. With the Modelica Association's focus on data standards, and LOTAR MBSE's focus on process standards, there is an obvious synergy and mutual benefits.

2. The Benefits of Preservation

Data preservation should be a leading consideration for all engineering models, and preservation planning should be a leading activity in all modeling endeavors. While the specific advantages of digital data preservation vary by industry, maintaining an engineering data archive offers broad benefits to any engineering enterprise or effort. The obvious renumerations include: the documentation of design decisions, permitting flexibility in the contributions of future personnel, historical evidence of process management, and sustaining the traceability of interrelated data from all applicable domains. Other practical but important factors include facilitating warranty investigations, product maintenance planning, a design history to address potential part obsolescence, and support for future modifications and enhancements.

A concise preservation plan typically includes the following elements: a repository with configuration management and robust data search capabilities; use of modeling techniques that promote data interoperability and integration; tools or processes to expose and capture model metadata; generation of summary model reports for future results verification; and the establishment of traceable links to source the original technical requirements and any reference data.

2.1 Preservation Process

Preservation begins with the formulation of a data preservation plan that identifies the model types, dependencies, applicable standards, and intended future uses. Archiving is performed at defined maturity milestones—such as design reviews or configuration releases—capturing fully verified and consistent data packages known as configured baselines.

Each archived package targeted for preservation includes a Model Manifest, which documents the model's structure, purpose, and metadata; and a Model Report, which records the model's execution, results, and verification and validation (V&V) evidence. Verification ensures the technical completeness and integrity of the archived data, while validation confirms that the model meets its intended purpose, and viability after storage and subsequent retrieval. To facilitate interoperability and futureproofing, data is translated into a standardized neutral format, such as those defined by STEP protocols.

As prescribed by formal Systems Engineering standards, all archived models must maintain clear traceability to their source requirements, design rationales, and validation artifacts. The preservation infrastructure itself must support robust search functions, metadata management, and integrity checks to ensure that the data remains unaltered and usable. This paper has already identified the key metadata standards, MIC-Core and

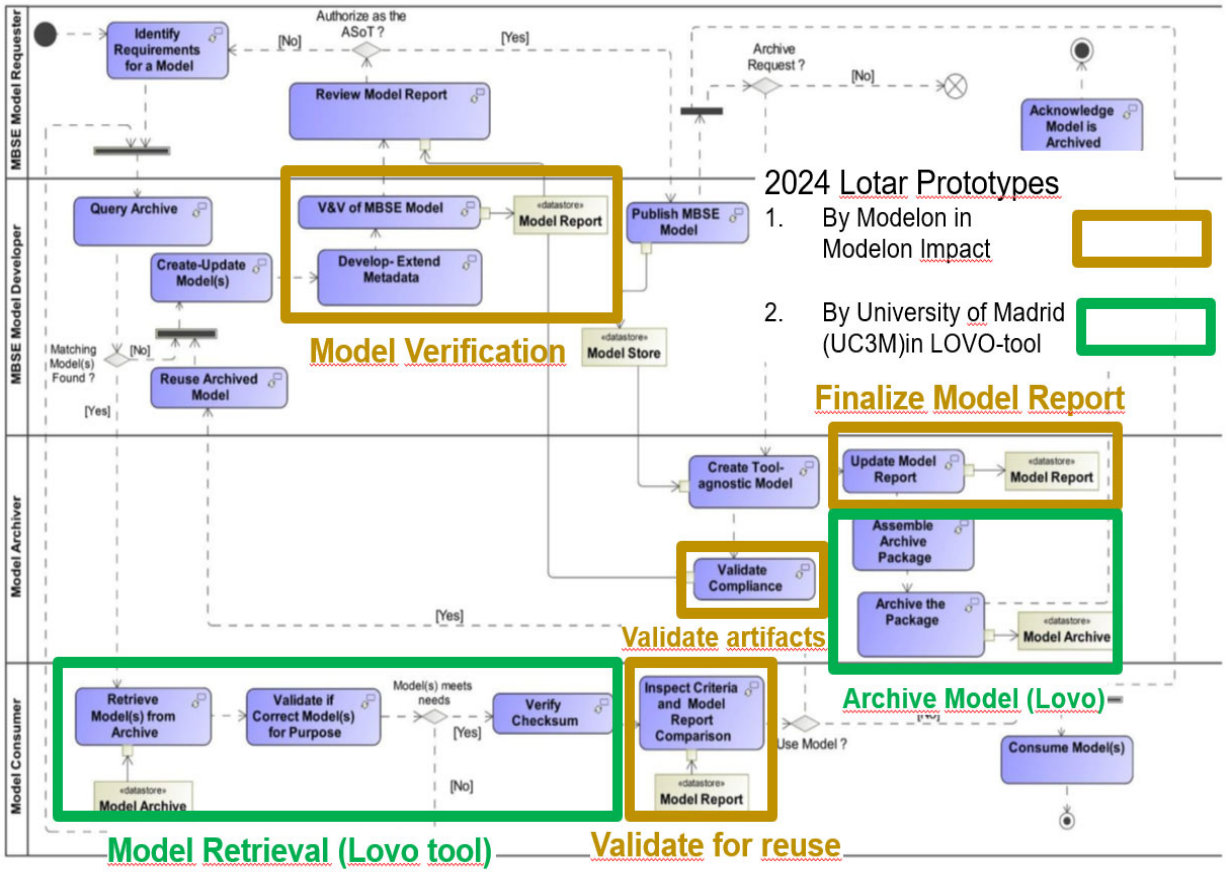


Figure 1: The LOTAR MBSE process flow with role-based swim lanes (Coïc et al, 2023).

MoSSEC, and the DDP as a packaging mechanism. The LOTAR standards identify a packaging framework, as described by the OAIS standard (ISO 2012), that compliments and formalizes the DDP.

The LOTAR preservation process, showing swim lanes for different enterprise roles for both the archival and retrieval process is depicted in Figure 1.

3. Types of Standards

The LOTAR domain specific Parts, previously defined as the P510, P515, P520, and P550 standards, are process standards that rely on widely available and robust data modeling standards. Data modeling standards support data interoperability between multiple brands of tools and users. They can also support translations into neutral or alternative formats. Data Modeling standards support model preservation if subsequent versions maintain compatibility through future software tool revision cycles. Some of the best examples of modeling standards are produced and maintained by the Modelica Association. Other notable modeling process standards for modeling and simulation are by NASA, (NASA 2024, NASA 2019).

3.1 Data Modeling Standards

The Functional Mock-up Interface (FMI) facilitates the exchange of simulation and analytical models, while System Structure and Parameterization (SSP) supports system assembly. Recent updates made them more appropriate for LOTAR purposes than they were originally, and the introduction of "Layered Standards" supports the concept and implementation of LOTAR even better. Two layered standards under development by their respective FMI and SSP projects stand out in how much they align with LOTAR needs:

- The SSP Traceability standard (to be released shortly) supporting model and process traceability; and the
- FMI-LS-REF standard that aims to add a standardized way to include simulation experiment definitions, multiple parameters sets and simulation result files to an FMU.

Together they will further enable the identification and management of metadata and traceability features in both FMUs and SSP containers.

DOI 10.3384/ecp218741

3.2 The Behavioral Model Results Report

During 2024, major progress was made to implement and test a prototype that enhances the Part 515 verification and validation process. The deliverables developed by Modelon AB included python code and a draft xsd-schema. This expanded earlier efforts to prototype the Part 520 process and integrate the Part 550 linking mechanisms directly into a Modelica model (Coïc et al. 2021, Coïc et al. 2023). The 2024 prototype added detailed checks on the completeness and correctness of all available FMU and SSP assets, as well as the model verification capability. The process break-through evolved by formalizing the existing infrastructure, used by the Modelica Association and modeling community, that automates the regression testing of behavioral models. Unfortunately, even with more than a decade of experience in automated model verification, the Modelica Association has never agreed on formalizing the definition of solver types and terminology, which are necessary for quantitative verification of simulation models. In addition to other features, the Part 515 xsd-schema formalizes the following elements of model verification:

- The link to the model to which it applies
- The required test cases
- The context of the test case
- The actual scenarios used
- The experiment definition, such as solver type, tolerances, start and stop times, and number of result points that should be stored
- The referenced variables
- Verification tolerances in the x and y direction
- Boundary conditions
- Modified parameters
- Links to the reference result files

This is a superset of the information currently envisioned for the FMI-ls-ref layered standard. The intention of the FMI-ls-ref standard was to enable a so called “smoke test” for exchanging FMUs, whereas the needs for verifying archived models for design reuse requires precisely quantified tolerances. Nevertheless, the development, agreement and utilization of a basic verification infrastructure as a layered standard to FMI would simplify future adoption.

The SSP Traceability standard by the MA provides a modular data standard that enables an in-depth and customized metadata definition. It does, without additions, not provide the precise definitions that are needed for long term archival.

4. Encoding Metadata

The LOTAR Working group’s standards include encoding the processes steps and defining the necessary metadata to enable reuse of models after potentially long periods of archive storage. In parallel to the LOTAR developments, the German national research project SET level (Otter et al. 2022, Setlevel 2023) initially developed what has become the SSP Traceability standard. As enhancements to the product development process, SET Level starts with defining the “Credible Decision Process” and the “Credible Simulation Process” (CSP). It also established a metadata format in support of SSP Traceability. It evaluates the credibility of the process by documenting the steps and linking and integrating all required artifacts. While the purpose for collecting and integrating the metadata is different, there are a lot of commonalities. SSP Traceability (MA 2025-1) has been defined as a feature that facilitates a large degree of process variation. An observer’s ability to understand the model’s purpose and then generate identical results at different times is common to both efforts. SSP Traceability is designed to capture metadata throughout the entire design process, whereas for LOTAR only the final maturity status must be set before an archiving decision is considered. However, model preservation for a longer period likely requires additional metadata that might not be needed for a shorter lifecycle. In any case, with the recently added ability to add “layered standards” in a compatible way to MA standards, the LOTAR MBSE standards could be built as a layered standard on top of SSP Traceability. The integration of this additional capability could simplify the implementation and future adoption.

The SSP Traceability standard contains several separate namespaces and parts, which don’t map 1:1 to the LOTAR specification, but would still make it possible to cover LOTAR’s intended results. The main elements of SSP Traceability are (MA 2025-1):

- **Decision Task Meta Data (DTMD)** is an XML format representing process-relevant information, as defined by the Credible Decision Process, with a file extension ending in dtmd.
- **Simulation Task Meta Data (STMD)** is an XML format representing process-relevant information, as defined by the Credible Simulation Process, with a file extension ending in stmd.
- **Simulation Resource Meta Data (SRMD)** is an XML format representing metadata for resources, with file extension ending in srmd.

Note that according to the SSP standard these metadata files can also be embedded in FMU files, not only in the SSP packaged files.

To use recurring description elements consistently, they must be defined separately from the three XML Schemas

above. As a result, these description elements are also available in the DTMD, STMD and SRMD XML Schema. The XML Schema for common content is named **Simulation Traceability Common (STC)** XML Schema representing recurring description elements of the other three XML Schemas. An important concept of SSP Traceability is the Glue Particle approach:

- The **Glue Particle Approach** is a concept for packaging data and the file-based transfer of process-relevant information and resources from a Credible Decision Process or a Credible Simulation Process.
- A **Glue Particle** is a package of process-relevant information and resources from a Credible Decision Process or a Credible Simulation Process. The form in which this bundle exists, e.g. whether it is stored as a file or in a database or a specific data format, is not specified.
- A **Glue Particle file** is a Glue Particle that is in a file-based representation and can therefore be transferred between tools or organizational units such as people, departments, or companies.
- File-based resources linked by or to a Glue Particle file should be exchanged with the Glue Particle file itself to take full advantage of the Glue Particle transfer.

SSP Traceability reuses the W3C XLink language (XLink 2012) for data element linking, whereas the P550 draft schema uses a simple URI schema that differentiates between local and global links. Both linking schemata have the same goal: to provide the "Authoritative Source of Truth" (ASoT) by creating traceable links between requirements, needs, and the artifacts to be verified or validated.

Part 520 metadata provide a set of classifications describing the model's pedigree and usage. The same classifications could equally be expressed with the ClassificationEntry elements in the STC schema. Some of the Part 520 data elements map more clearly with elements in the STMD schema, such as the Integration_Process_Type.

The Part 515 schema focusses on the data needed to verify the model results after retrieval from an archive. The data may be different than what was used to verify the original analysis requirements and results. As mentioned earlier, this is similar to the current draft fmi-ls-ref specification but with additional considerations. STMD on the other hand focuses on the entire verification process of the original model. Undoubtedly, STMD data would be of great use for an engineer trying to reuse a model after it spent a decade or more being buried in an evolving archive.

Prototypes for creating and validating the P515 and P520 LOTAR MBSE archiving process were implemented in the Modelon Impact software application to demonstrate that they can easily be accomplished and repeated using Modelica-based tools.

5. Representing Analytical Models

Utilization of the Modelica language satisfies many of the needs of the P520 process, but in today's environment the majority of models are developed using proprietary tools. However, most of these tools now support the export of models as Functional Mock-up Units (FMUs). In principle, text-based language models have fewer intricate implicit dependencies on details of the execution environment such as Application Binary Interfaces (ABIs) and dynamically linked libraries (sometimes referred to as "dll-hell"), which doesn't go to the full depth of the second problem, major revisions of the underlying operating system. The volume of metadata needed to define, let alone guarantee, a compatible execution environment is quite high. However, the containerization of computational resources has at least made it easier to recreate the original settings, but preservation planners must still account for open-source or proprietary operating systems, plus any additional system libraries and software platforms. A partial remedy to the execution environment lifecycle is to use "source-code FMUs", but they are not supported by many tools. The detailed "Build Configurations" available in FMI 3.0 will improve portability, ease of use, and expand the archiving alternatives. There are further improvements in FMI 3.0 that "sharpen" the semantic precision of models and therefore accommodate the packaging of more sophisticated simulations of integrated systems and controls. The most relevant features here are:

- Clocks
- Transition events in co-simulation FMUs
- Interface provisions for scheduled executions
- New data types, including binary data
- Methods to manage the build configuration
- A new implementer's guide to improve the user's interpretation of the standard.

In earlier work prototyping the P550 concepts (Coïc et al 2023), Modelica records were used to embed interactive traceability links to elements (e.g., requirements parts, attributes) directly in the Modelica code. They were modelled as Strings in a traceability record. Now that the Modelica Association has exposed the SSP Traceability standard (current status - release candidate, but several commercial tools appear to be early adopters), it seems reasonable to tag elements with a Modelica Resource, enabling a standardized annotation link to point to the

DOI 10.3384/ecp218741

required resources. Such resources can even be embedded in the SSP.

6. Model archive and retrieval: verification and traceability

As discussed previously, preservation of engineering artifacts is a critical capability for organizations seeking to enable long-term traceability, reuse, and maintain compliance in complex system development. The preservation process is not limited to mere storage. As illustrated in Figure 1, data preservation requires a robust workflow for model identification, verification, archival, and retrieval. Combining these steps ensures the integrity and accessibility of engineering data over time. In this section, we briefly describe significant process enhancements to the archive and retrieval of engineering artifacts by leveraging metadata verification, packaging standards, and knowledge graphs for queries and model identification.

1) The archival process starts with the verification of a **metapackage** described through an instance of the P550 schema. This metapackage-information serves as the entry point for validating the consistency of the engineering artifacts to be preserved. Each metapackage comprises references to one or more subpackages (e.g., P510, P515, P520), each associated with a manifest and a corresponding engineering artifact. Before indexing any package into an archive, two levels of verification are required:
   - **Traceability level**: the archival process must ensure that all traceability links between the engineering artifacts actually exist and are all accessible. This includes validating cross-references between requirements, models (logical and physical), and model elements.
   - **Artifact level**: the content of each artifact must also be correct in terms of data formats. For instance, artifacts of type **P510** (e.g., requirements) must comply with the ReqIF standard. Artifacts of type **P520** (e.g., physical/analytical models) must be executable with the associated FMI components and produce the results specified in the manifest and model report. In general, each manifest must declare a valid checksum and timestamp to verify content immutability.

   Once these two aspects are verified, packages can be stored in the archive. The process supports updating the manifest metadata as necessary to reflect any corrective or clarification changes. This guarantees the long-term preservation of both the engineering artifact and its descriptive. context
2) The retrieval process for preserved engineering artifacts follows a similar set of checking actions. Given a package is contained in an archive, the system starts from a P550 instance and traverses all associated subpackages recursively. Each subpackage is unpacked, and its manifest is opened and validated using the same rules as in the archival phase. If the type of artifact is known (P510, P515, P520, etc.), the appropriate toolchain is invoked to:
   - Parse the manifest and validate schema conformance.
   - Check that the artifact's format (e.g. ReqIF, FMI) is as expected.
   - Run simulations or syntax checks as needed to validate behavior or structure with an external tool.

Once verified, both metadata and artifact data can be exposed for further processing, modification, search, or visualization.

6.1 Knowledge Graphs and Search Queries

While the verification process ensures and maintains model integrity, precision identification, suitability and selection of engineering artifacts requires more powerful query and exploration capabilities. To address this, we implemented a semantic layer on top of the archive based on knowledge graph representations. Two strategies were implemented: 1) a Label Property Graph (LPG) using Neo4j Desktop 1.5.9 and 2) an RDF Graph using Apache Fuseki 5.2.0**.** These representations were derived directly from the XML-based manifests using the following transformation rules:

1) XML to Label Property Graph transformation.
   - Each XML element becomes a **node** with a unique identifier and a type-classification derived from the element's type in the XML Schema.
   - Attributes of elements are mapped to their respective node properties (e.g. attribute_name:attribute_value).
   - Hierarchical relationships are transformed into directed edges between nodes, named as `has_<element_name>`. For simple (literal) child elements, values can also be modelled as node *properties* of the parent node rather than separate nodes to avoid too many edges.
2) XML to RDF. The second approach makes a representation of the XML content in RDF (an OWL/RDFS ontology was previously created to derive all XML Schema definitions as classes and properties):
   - Each XML element is an instance of an RDF resource, where the resource type corresponds to an XML Schema Complex Type represented as a class.
   - For each element, a triple is created linking it to its parent using a property named `has_<element_name>`, and then typed either as

an ObjectProperty, AnnotationProperty or DatatypeProperty depending on its origination and whether the value is complex or literal (user defined or from a pre-defined dictionary of terms).

- Every resource is annotated with a rdfs:label to ensure a human-readable identifier.

3) As illustrated in Figure 2, a graph structure enables the integration of the model manifests and the opportunity to perform traversal-based queries using languages such as SPARQL or Cypher. Through the use of queries engineers can explore model features, data relationships, or perform consistency checks. A sample of query results is depicted in Figure 3. A query example, "***Find all scenarios that have a verification credibility level 'Low'***" is defined using SPARQL in Figure 4.

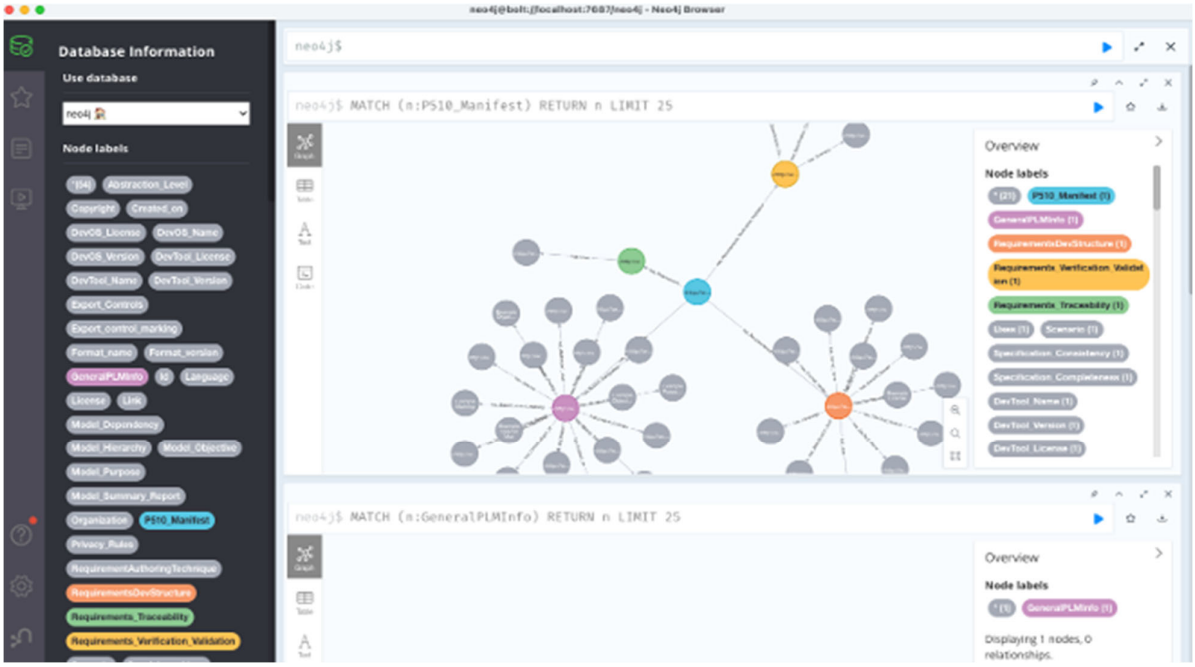


Figure 2: Visualizing a Manifest in Neo4J

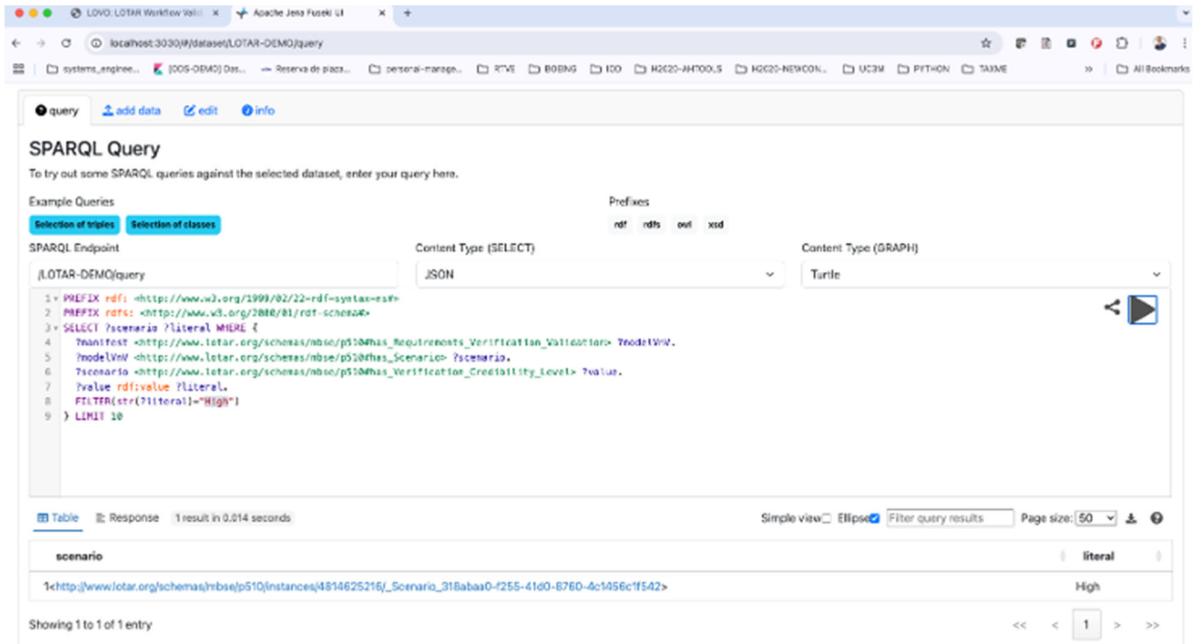


Figure 3: Results of the SPARQL Query

DOI
10.3384/ecp218741

```
PREFIX rd.:
http://www.w3.org/1999/02/22-rdf-
syntax-ns#
PREFIX rdfs: http://www.w3.org/2000/01/
rdf-schema#
PREFIX 510: http://www.lotar.org/schemas
/mbse/p510#
SELECT ?scenario ?literal WHERE {
?manifest
p510:has_Requirements_Verification_
Validation ?modelVnV.
?modelVnV p510:has_Scenario ?scenario.
?scenario
p510:has_Verification_Credibility_Level
?value.
?value rdf:value ?literal. FILTER(str(?literal) =
"Low")
}
```

Figure 4: Sample SPARQL query

The engineering artifacts and sample data used for these examples are hosted in the Stratoliner Project (MBE-Demonstrator-RM 2024). The sample source models and manifests included a text-based technical requirements specification in a ReqIF model format, and two FMUs depicting the airplane's alerion actuators. All of the verification and traceability capabilities presented in this section are implemented within the LOVO (LOtar Validation tOols) framework, included in the Stratoliner Project. LOVO will be released under license in a common Python package repository by the authors in 2025.

7. Summary

Significant progress has been made in defining the fundamental parts of the LOTAR MBSE archiving process. This includes multiple tool prototypes to develop and demonstrate feasibility, code to automate many of the process steps, and to-be-released process documentation to standardize the procedures. Many of the techniques necessary for a proper archival and retrieval process are currently un-documented "best practices" developed by experienced simulation experts. But these process advances are usually implemented as an ad-hoc methodology, and with inconsistent technologies across different companies. The LOTAR MBSE schemata formalize these practices and thereby facilitates their eventual integration into familiar off-the-shelf tools.

The initiatives for the SSP Traceability standard, primarily motivated by the needs of the automotive industry, and the aerospace driven LOTAR MBSE standards, were developed in parallel at about the same time. Both industries have similar needs for data traceability and model interoperability. It is therefore assumed that increased coordination and dialog between the respective representative organizations will encourage and accelerate their adoption by the software tool vendors. Although the current Standards produced by the Modelica Association have significantly enhanced the LOTAR process options, multiple challenges remain. The Modelica Language should include the prescriptive identification of the common metadata elements. If the SSP Traceability standard could also apply to FMU containers, and the source analytical models, traceability would be significantly improved. The true advantage would be integration within the Modelica language.

The LOTAR MBSE standard originally was designed to use only the FMI standard, but the extension to SSP is a logical step. Unfortunately, industry's adoption of SSP is lagging behind FMI. The metadata that extends to the SSP container will be slightly different and needs its own unique set of system level classes. Many of the metadata elements could be captured automatically by the original modeling and simulation applications and automatically reproduced in the XML-files used for model integration and traceability. When exported and exposed to the repository, the model metadata would support queries across the LOTAR MBSE preservation archives.

Layered standards, such as SSP Traceability, are quite new in the Modelica community. They offer significant opportunities for LOTAR-type initiatives to satisfy their specific process needs on top of the well-established standards of the Modelica Association (MA) (e.g. FMI and SSP). However, basic instructions and knowledge of how to specify them are not generally available. A published guide about how to develop and coordinate layered standards by the MA would notably improve their adoption. Layered standards, if designed and used correctly, will make implementation of related data standards much easier and facilitate their widespread use.

8. Acknowledgements

This work has been supported by the yearly project prototypes (2022, 2023, and 2024) developed under the sponsorship of LOTAR International, a consortium of industry volunteers sponsored by PDES, Inc. In the United States, and prostep ivip in Europe, with publication services provided by the Aerospace Industries Association (AIA) in the U.S. and the Aerospace and Defense Industries Association of Europe for Standardization (ASD-STAN) both under the supervision of the International Aerospace Quality Group (IAQG).

The authors would also like to thank:

- Jesse Gohl, Modelon AB, for his work in implementing the prototype archiving mechanism

using FMI and SSP in Python and integrated in Modelon Impact.
- Model Based Innovation LLC, Hartford CT, USA, a company dedicated to Modelica standards, supporting the investigation into analytical and simulation tools.

References

ASD-STAN, Aerospace and Defense Industries Association of Europe (2025), standards published at https://www.asd-stan.org/en

AIA, Aerospace Industries Association (2025), in partnership with the American National Standards Institute (ANSI) standards published at https://www.aia-aerospace.org/

Coïc, Clément Mark Williams, Juan Carlos Mendo, José María Alvarez-Rodríguez, Marcus Richardson (2023). "Linking Design Requirements to FMUs to create LOTAR compliant MBSE models". In: "2023 FMI User Meeting". URL: https://fmi-standard.org/assets/literature/FMI_User_Meeting_2023/04_Co%C3%AFc-Linking_Design_Requirements_to_FMUs_to_create_LOTAR_compliant_mBSE_models-168_a.pdf

Coïc Clément, Adrian Murton, Juan Carlos Mendo, Mark Williams, Hubertus Tummescheit, Kurt Woodham (2021-10). Modelica, FMI and SSP for LOTAR of Analytical mBSE models: First Implementation and Feedback. DOI: https://doi.org/10.3384/ecp2118149

IAQG, International Aerospace Quality Group, (2025) The certification and registration authority for commercial aerospace standards and suppliers, https://iaqg.org/

IRT SystemX (2020). Model Identity Card (MIC): Towards a standardization of the specification and description of simulation models. https://mic.irt-systemx.fr

ISO 2023, ISO/IEC/IEEE 15288:2023 Systems and software engineering — System life cycle processes. https://www.iso.org/standard/81702.html

ISO 2018, ISO 26262-1:2018, Road Vehicles – Functional Safety. https://www.iso.org/standard/68383.html

ISO 2021, ISO 10303-243:2021, Industrial automation systems and integration — Product data representation and exchange — Part 243: Application protocol: For modelling and simulation information in a collaborative systems engineering context (MoSSEC), https://www.iso.org/standard/72491.html

ISO 2012, ISO 14721:2012, Space data and information transfer systems – Open archival information system (OAIS)

LOTAR MBSE 2025, LOTAR Model Based Systems Engineering Workgroup, LOTAR Internation. URL: https://lotar-international.org/lotar-workgroups/model-based-systems-engineering/

MBE-Demonstrator-RM 2024, https://github.com/MBE-Demonstrators/MBE-Demonstrator-RM, a GitLab prototype and testing environment using a Stratoliner's structure and modern aerospace systems

Modelica Association (2023-03). Modelica – A Unified Object-Oriented Language for Systems Modeling. Language Specification Version 3.6. Tech. rep. Linköping: Modelica Association. URL: https://specification.modelica.org/maint/3.6/MLS.pdf.

Modelica Association (2024-12). SSP, System Structure and Parameterization Standard – Document version: 2.0 URL: https://ssp-standard.org/docs/2.0/

Modelica Association (2024-11). FMI Functional Mockup Interface Specification – Document version: 3.0.2. URL: https://fmi-standard.org/docs/3.0.2/

Modelica Association (2025-1). SSP Traceability Standard Specification, Document version 1.0.0 beta 2. https://modelica.github.io/ssp-ls-traceability/master/, accessed April 16, 2025

Modelica Association (2025-2). FMI Layered Standard REF draft standard, https://modelica.github.io/fmi-ls-ref/main/ accessed April 16, 2025

Modelica Association (2025-3). SRMD metadata format, part of the SSP Traceability Standard, version 1.0.0 beta 2, https://modelica.github.io/ssp-ls-traceability/master/#sec-srmd, accessed April 16, 2025

Modelon 2025, Modelon Impact Modelica platform, https://help.modelon.com/latest/, accessed April 29, 2025.

NASA 2019. NASA Handbook for Models and Simulation: An Implementation Guide for NASA-STD-7009. https://standards.nasa.gov/standard/NASA/NASA-HDBK-7009

NASA 2024. NASA STD 7009B. Standard for Models and Simulations. https://standards.nasa.gov/standard/NASA/NASA-STD-7009

Otter, Martin, Matthias Reiner, Jakub Tobolář, Leo Gall, Matthias Schäfer, 2022 Towards Modelica Models with Credibility Information, https://doi.org/10.3390/electronics11172728

Prostep ivip 2024, Digital Data Package (DDP), https://www.prostep.org/en/projects/digital-data-package, accessed April 16, 2025.

Radio Technical Commission for Aeronautics 2011, DO-331 - Model Based Development and Verification Supplement to DO-178C and DO-278A, https://my.rtca.org/productdetails?id=a1B36000001IcfjEAC

Radio Technical Commission for Aeronautics 2011-2, DO-333 - Formal Methods Supplement to DO-178C and DO-278A https://my.rtca.org/productdetails?id=a1B36000001IcffEAC

Setlevel 2023, Das SET Level Projekt, https://setlevel.de/

XLink (2012). World Wide Web Consortium, Inc. (W3C): XML Schema 1.1. April 2012. https://www.w3.org/XML/Schema

Automation Nation: Taming Complex V&V Workflows

Erik Rosenlund[1,2] Robert Hällqvist[1,2] Robert Braun[2] Petter Krus[2]

[1]Saab Aeronautics, Linköping, Sweden
[2]Fluid and Mechatronic Systems, Linköping University, Sweden

Abstract

In the realm of model verification, ensuring traceability and repeatability is paramount for achieving Modeling & Simulation (M&S) credibility and development efficiently. This paper explores challenges and solutions for expressing complex model Verification & Validation (V&V) workflows that ensures readability while still enables automation, parallelization and advanced customization of verification activities. This paper contributes a practical solution that is evaluated through a detailed implementation within an in-house developed multi-purpose automation and Verification & Validation (V&V) framework.

Keywords: Model Verification, Machine-Interoperable Traceability, Workflow Automation, Model-driven Engineering, Simulation Credibility, Verification and Validation

1 Introduction

To quote Balci and Ormsby (2000): “any use of a model in a context where it can not be proved to be credible is by definition not credible”. Credibility is, to say the least, a broad concept involving many aspects of model development and use (*Standard for models and simulations* 2024). As systems and models become larger and more complex (Ahmed, Shah, and Umar 2016), so does the need to establish model credibility (Arthur et al. 1999). Where previously a few verification scenarios were executed and verified manually (Arthur et al. 1999), there is now a move towards extensive model exploration (Bouskela et al. 2023) with increasingly complex, multistep verification activities being implemented (Rosenlund et al. 2024).

In several ongoing industrial research projects, there is a large focus on traceability and frameworks such as System Structure and Parameterization (SSP) Traceability and the Credible Simulation Process (Ahmann et al. 2022). Where traceability of verification activities was previously based on manual documentation, as verification activities increase in size and complexity, this approach is no longer plausible (Arthur et al. 1999). The growing number of experiments increases the need to automate the verification process to maintain development efficiency. And just as in research, one of the core pillars for motivating trust in results is the reproducibility of experiments (Dalle 2012), where traceability of data, both regarding the conducted activity and its results, enables decisions to be taken on "known knowns" and "known unknowns" (Nallaperumal and Krishnan 2013).

To address these needs, the primary research question is formulated as follows:

> **Research question**
>
> How can automated model verification be conducted in a manner that is both traceable, repeatable and comprehensible?

Our proposal couples a lightweight workflow schema based on the Extensible Markup Language (XML) with Git-based change management to satisfy both repeatability and traceability (Figure 1). Because every transformation and activity are explicit in the workflow and intermediate artifacts are version-controlled, stakeholders can audit exactly how a simulation result was obtained. In proposing a schema for representing complex model verification workflows, we build upon insights from existing solutions used in automated data processing while adhering to requirements specific to model verification. Furthermore, evaluation of existing change management methods seek to enhance traceability throughout the verification process; potentially leveraging a mature ecosystem for data management.

To enable close collaboration between industry and academia, the established method denoted as "Industry-as-laboratory" by Potts (1993) was utilized. A number of use cases, established from industrial needs, were used for evaluation of different model verification and exploration

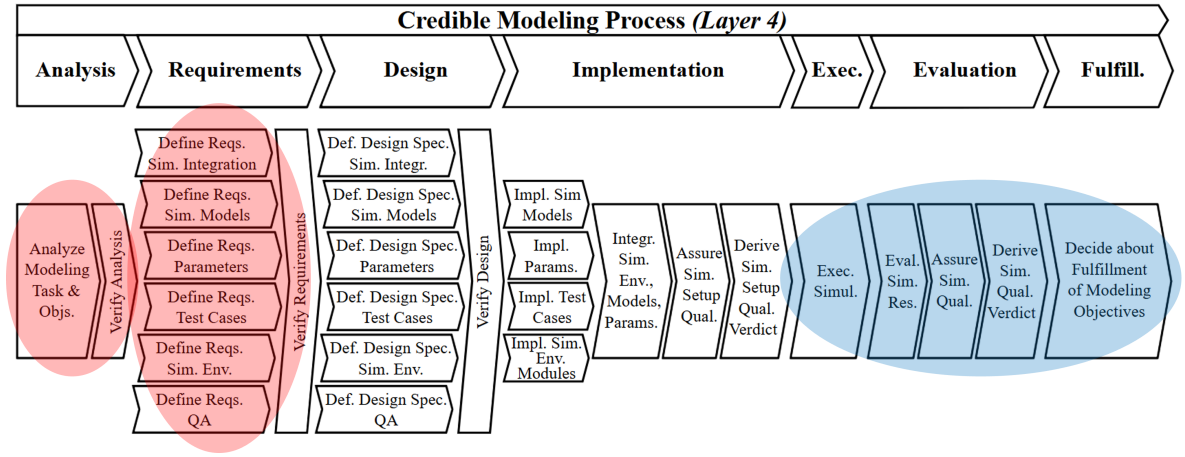


Figure 1. Visualization of the Credible Modeling Process Layer of the Credible Simulation Process Framework (Modelica Association 2022). The red areas showing the verification design phase and the blue area the verification execution phase

scenarios in an in-house automation and testing framework. The focus is here on verification, but the solution can equally well be applied to validation, e.g. as described in (Hällqvist 2023). Looking ahead, our findings offer promising avenues for further refinement and extension in future research, addressing ongoing challenges and exploring new opportunities in model management and V&V.

Contributions This study contributes to the advancement of model V&V through the integration of both novel and established solutions, aiming to promote greater transparency in the process. This by introducing a traceable and repeatable workflow representation—as a simple XML schema—that promotes readability and enabling automation.

The work is demonstrated within an in-house V&V automation framework, integrating established version control methods within model archives. It explores aligning model change management with existing software practices and the SSP Traceability framework, offering a scalable approach to managing changes and enhancing traceability and automation both during and after initial model development.

2 Theoretical Background

This section introduces the key frameworks and concepts that support the development and traceability of simulation models. These include the Credible Simulation Process Framework, version control systems, automated verification and validation mechanisms, and data pipeline architectures. Together, they form the basis for ensuring model credibility, enabling traceable workflows, and supporting scalable, reusable simulation practices.

Credible Simulation Process Framework and SSP Traceability To ensure that design decisions are based on credible data, SSP Traceability (which acts as an enabler of the Credible Simulation Process) was developed. This research focuses on a set of activities that are part of the Credible Modeling Process layer within the Credible Simulation Process. As illustrated in Figure 1, the Credible Modeling Process is largely analogous to the model development and delivery process used at Saab Aeronautics (Andersson and Carlsson 2012). The activities highlighted in blue in Figure 1 are addressed in this work, while those marked in red provide essential artifacts that serve as prerequisites for the activities under focus. A subset of these artifacts is discussed in (Rosenlund et al. 2024).

The Credible Simulation Process Framework allows the tracking of modeling activities, digital artifacts, and decisions to provide a solid foundation for model credibility assessment tasks. It provides the means of documenting data related to simulation tasks in a transparent and comprehensive manner (Modelica Association 2022). The related metadata is stored alongside the model using mechanisms implemented in SSP for coupling metadata related to the model within the same container. It employs various specialized metadata formats, such as Simulation Task Meta Data (STMD) for tracing tasks and Simulation Resource Meta Data (SRMD) for resource data.

DOI 10.3384/ecp218751

Version Control System Extensive usage of Version Control System (VCS) is employed by software companies to maintain collaboration and manage evolving products (Khleel and Nehéz 2020). They are used to storing code, documentation, and related artifacts in a traceable manner, where each change is accompanied by metadata detailing its relationship to other versions, the date, and the author (Khleel and Nehéz 2020). A transition from Centralized Version Control System (CVCS), such as Apache Subversion (SVN), to Decentralized Version Control System (DVCS), e.g. Git, is in progress (Muşlu et al. 2014), largely to enable more fine-grained control during development. The key benefit, in our context, of this transition is that it enables using VCS for tracking model changes even when the model is packaged within archives, as is the case for Functional Mock-up Interface (FMI) and SSP.

Verification and Validation Automation Framework An internal automation framework for model verification and validation, which supports extensive model exploration—the term here referring to the activity of subjecting the model to multiple scenarios while its behavior is checked against requirements or validated against test data (Hällqvist 2023)—serves as an application playground for this research. Model exploration in this context serves to strengthen model credibility thus enabling efficient and large scale model development, model-based decision-making in any life-cycle phase, and improve anomaly detection in model or physical product. The framework provides mechanisms for documenting both model and simulator Operational Domains (ODs) and Domains of Validation (DoVs)[1]. The DoV are all operating conditions where the model has been tested, including outcomes of verification or validation activities, and it serves as a foundation for assessing various aspects of the model's *credibility* (*Standard for models and simulations* 2024). In contrast the OD (or *applicability domain*) defines where the model can be used in regard to a set of requirements, including acceptance criteria, that reflect the model's intended use. A result of this is that the OD will always be a subset of the DoV, since requirement fulfillment outside the DoV is per definition unknown. More detailed descriptions of mode ODs and DoVs can be found in the literature, see for example Oberkampf and Roy (Oberkampf and Roy 2014) and Beisbart et al. (Beisbart and Saam 2019). Establishing both ODs and DoVs is crucial for ensuring the appropriate and credible application of a model, promoting traceability with respect to verification activities and facilitating model reuse by clearly communicating model limitations (Rosenlund et al. 2024).

Data Pipeline / Workflow Frameworks A data pipeline—or workflow—is based upon the architectural pattern **pipes and filters**, in this case more specifically the **tee and join** variant, where filters correspond to activities and pipes to the chaining of these activities (Buschmann et al. 1996). In short, pipes are a set of activities arranged in a specific order, where inputs and outputs are connected between the activities, often creating a Directed acyclic graph (DAG). The pattern's main advantage, when constructing verification workflows, is that "changes should be possible by exchanging processing steps or by recombination of steps, even by users" (Buschmann et al. 1996). The pattern also provides determinism regarding execution order and data flows; from the activity Directed acyclic graph (DAG), parallelization opportunities can be identified. Due to the often computationally expensive operations related to running simulations, this can result in considerable performance gains.

There are a number of existing software solutions that employ this pattern in various forms; two examples of more specialized solutions are **Jenkins** (*Jenkins (software)* 2025), for setting up build flows, and **dask** (*Dask (software)* 2025) for data processing and computation workflows. These solutions employ various methods for expressing workflows, using specific Domain-specific languages (DSLs), general programming languages such as python or existing data storage solutions such as JavaScript Object Notation (JSON) or XML.

3 Methodology

The methodology adopted in this work was driven by the need to define a verification workflow representation that could balance traceability, repeatability, readability, and automation possibilities. The foundation of the implementation lies in a set of verification activities that must be supported, as well as in the coupled requirements governing the verification process. The activities and requirements, collected from verification engineers, are in several cases supported by other sources to strengthen their validity (Cederbladh, Cicchetti, and Suryadevara 2024; Andersson and Carlsson 2012). The resulting methodology is presented in

[1] Or Domain of Verification

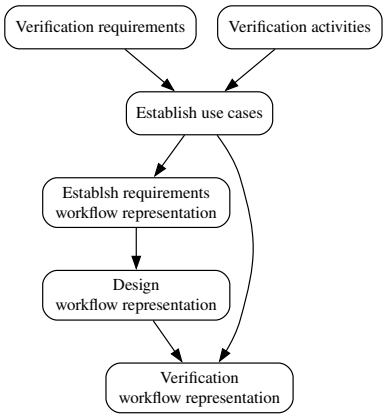


Figure 2. The research process employed

Figure 2, which illustrates the research and development process: starting with the collection of key verification activities and requirements, moving toward well-defined use cases, and culminating in a fully specified, validated workflow.

Requirements

- **Traceability** - Create traceability regarding verification activities and its results, striving for alignment against SSP Traceability
- **Repeatability** - Repeatability of simulation results, this implies that any operation must be deterministic
- **Automation** - The workflow needs to be automated to streamline development and enable effective regression testing
- **Human and machine-readable** - To facilitate both manual design and verification together with a high level of automation, any workflow representation must be both human understandable and machine-readable
- **Customization** - To enable reuse of similar operations between models
- **Resource utilization** - Computationally expensive operations require parallelization to ensure a quick feedback loop during V&V and development

Verification activities

- **Custom activity** - Easy implementation of verification activities
- **Activity comparison** - Compare the outcome of different activities, including simulation results for different scenarios or parameter sets
- **Result aggregation** - Allow aggregation of results

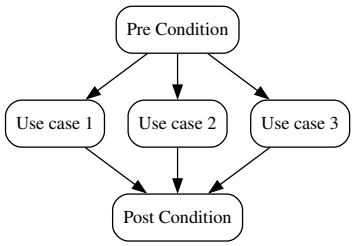


Figure 3. Multiple use cases sharing a common pre and post conditions

Table 1. Mapping between verification activity and use cases

	UC1	UC2	UC3
Activity Comparison	x		
Result aggregation	x	x	
Model Exploration		x	x
Design Optimization			x
Design of Experiments			x

- **Model exploration** - Enable methods of model exploration to evaluate different scenarios or parameter sets
- **Design optimization** - Provide means of using iterative optimization methods such as Bayesian or evolutionary algorithms
- **Design of experiments** - Evaluate what experiments would provide the most value in regard to what we currently know, often implicitly fulfilled by **Design optimization** since this is the first step when using iterative optimization methods

3.1 Use case

Several different use cases were formulated to capture the functionality needed to support the verification activities while fulfilling the defined requirements. Due to the broad nature of the activities, creating multiple smaller use cases enables a better coverage of the activities while also enabling a tighter connection to the final verification, see Figure 3. A mapping between the verification activities and use cases is in Table 1 ensures coverage.

Pre Condition The cases start with a model and a need to verify, validate or optimize certain aspects of the model or system.; they all begin with the establishment of the verification workflow representation itself where all activities are defined and it is decided what additional artifacts are to be saved for traceability. This representation is stored within the model archive for traceability and enabling repeatability, in alignment with SSP Traceability.

DOI 10.3384/ecp218751

Post Condition The tasks all have a final activity where the additional traceable artifacts relating to the verification are packaged within the model (FMI) or system (SSP) container, this to enable a strong traceability connection between the verification activity and data to the model. The post condition states that there should be a containerized model that can be utilized and simulated.

3.1.1 Application Use Case 1

Description The first use case is designed to evaluate multiple product designs over a number of test scenarios. A real world example may be when evaluating different pipe sizes in a system during different load conditions. The process begins by establishing what test scenarios are to be evaluated. For each test scenario, multiple model variants are created and simulated, and the results are subsequently compared. Since all simulation variants are independent of one another, they can be executed in parallel.

Scenario

1. Load existing scenarios from a file, store the scenarios or its location
2. Create the different design variants, store meta data of the design variations
3. Simulate the different variants in parallel
4. Collect and compare the Quantities of Interest (QoIs), store the results associated with the QoIs

3.1.2 Application Use Case 2

Description The second use case is designed to evaluate different behaviors of a model, this by creating multiple combinations of input stimuli and evaluating the results. This is often applicable as verification after model adjustments or prior to deployment to verify that the model is fully functioning. May involve using a number of preset scenarios or creating randomized permutations of scenarios based on methods such as Monte Carlo, then evaluate the resulting behavior in regard to one or multiple requirements and finally generate a OD that corresponds to the fulfillment of that requirement. Just as in the first case, the generated scenarios are not dependent on each other and all can be executed in parallel. To fulfill the requirement of repeatability, any randomization based method must either store the full scenario or metadata enabling recreation of the full scenario. Metadata could for example include reference to the software used for scenario generation, randomization method and/or seed.

Scenario

1. Create the different scenario variants, store meta data regarding the scenarios
2. Simulate the different variants in parallel
3. Collect the QoIs, evaluate and aggregate the fulfillment of the requirement against the DoV as an OD, store the resulting OD using Simulation Task Meta Data (STMD)

3.1.3 Application Use Case 3

Description In the third case, the goal is to find the optimal design of a geometry under some predefined constraints by using Bayesian methods for optimization, where the outputs of previous simulation are used to find better sampling positions. May be applicable in cases were complex interactions between models make optimization challenging, for example in optimizing cooling capabilities in relation to usage scenarios. Using iterative methods creates difficulties when it comes to determinism and repeatability of results. Synchronous batch-sequential methods have previously been employed in other areas using Bayesian search methods (Tran et al. 2022). In these the simulations are grouped in batches and when a batch is completed the next one is allowed to start. This allows each batch to utilize the outcome of previous batches to calculate new sampling positions while ensuring determinism. The main downside of this method is that it is often less effective than using methods that do not take batching into consideration, since the variance of the model simulation time can greatly influence the utilization ratio of the computational resource.

Scenario

1. Store the optimization algorithms and corresponding parameters, variables and, if applicable, randomization seed
2. Create the batch of design variants to evaluate based upon known information, store meta data of the design variations
3. Simulate the batch in parallel
4. Collect the QoIs and evaluate if the design goal is fulfilled, if not return to Item 2
5. Store the parameter set of the final design

3.2 Requirements of Workflow Representation

The workflow representation is to provide clarity as to what, and in which order, any verification

activity is to be executed. It should help maintain a clear trail of artifacts, enabling determinism, traceability and repeatability. To enable complex scenarios, it should support nested workflows, where sub-workflows have a clear connection to parent activities. Moreover, it should enable the execution of independent tasks concurrently, whether locally or in a distributed environment such as High Performance Computing (HPC), to optimize resource usage and efficiency.

The representation should have the ability to encapsulate common verification actions, such as simulation runs or result comparisons, into reusable components, to reduce code duplication, enhance maintainability, and lower the risk of human errors. To further enhance the possibility for re-use of components, it should possess the capacity for easy customization of activities, such as test conditions, parameter sweeps, and simulation scenarios. The workflow should also enable adaptation to intermediate results by using conditional rules that change the next steps based on factors like missing artifacts or whether goals were met. Finally, it should allow for comments, fostering clarity and collaboration by documenting the purpose and logic of each part of the workflow.

3.3 Design of Workflow Representation

We identified existing solutions for representing verification workflows and considered established domain-specific languages and workflow engines, such as Jenkins, Dask, Nextflow, Prefect, Luigi, Airflow, and Snakemake. While these tools offer advanced parallelization and dependency management features, our analysis came to the conclusion that their complexity and overhead often exceeded what was necessary for this scope.

Instead, we landed in a simplified Application Programming Interface (API) to express these tasks and dependencies. This approach maintained the option to layer on more sophisticated workflow management tools in the future, should complexity or scalability demands increase. Once the interface was established, the next decision was how to represent the workflows themselves: using an existing General-purpose language (GPL), such as python or develop a custom Domain-specific language (DSL) in the form of existing markup languages such as xml or json. Using a GPL offers the benefit of leveraging a mature ecosystem with readily available tools. However, it also requires that users are proficient in the chosen language and capable of effectively make use of existing tools. To keep complexity down and simplify access, both to users and potential no code/low code applications, the solution landed on limiting supported features though the utilization of an existing markup language. But to cater the more advanced users, it was also made possible to exploit the API directly as showcased in the Appendix under Listing 10 in cases where access and readability to non developers is deemed less important.

Selection of markup language began by evaluating common existing key-value pair data formats, including YAML Ain't Markup Language (YAML), JSON, Tom's Obvious, Minimal Language (TOML) and XML to determine how they fit the requirements. The final selection of XML was guided by three main considerations. First, the chosen format had to be easily interpretable and editable by engineers, ensuring that the workflow could be understood, maintained, and customized without invoking a specialized programming knowledge. Second, the storage solution needed to supply flexibility and be extensible to support a diverse range of verification tasks, including dependency management, parameterization, and dynamic adaptations. Third, we prioritized easy to use parsing possibilities and schema validation capabilities, all of which contributed to improved maintainability and scalability. Fourth, the selected solution should enable interoperability with traceability frameworks sush as SSP Traceability. To ensure that the verification workflow remains accessible and robust, we imposed guidelines on the structure and complexity of the XML files. Using standardized field naming conventions, encouraging descriptive comments and employing schema validation provides immediate feedback and clarity on correctness.

3.4 Version Control

To enable traceability of the evolving model and not just the final version, VCS—specifically Git—was integrated into existing model packaging formats such as FMI/SSP archives. In this use case, version control and packaging standards serve **orthogonal purposes**. The solution does not change the internal structure of the archive, it merely adds data for tracking changes. The combination a symbiosis; embedding a repository inside an archive enables tracking changes, ensures reproducibility, and allows auditing of the lineage of verification tasks. This approach used familiar version control mechanics, enabling seamless collaboration, comparison of different versions, and

DOI
10.3384/ecp218751

storage of historical results for review.

3.5 Verification of Workflow Representation

Finally, the methodology was verified through tests that confirmed its ability to meet the stated requirements. The approach was stress-tested by scaling the number and complexity of verification scenarios according to the use cases, with observations showing stable performance, clear error-handling mechanisms, and effective parallelization strategies. These tests also examined the impact of version control integration on model archive size and complexity.

4 Results

The solution consists of two distinct parts: first, the markup schema for defining the verification workflow, allowing for easy customization and repeatability of verification activities, its inherent portability making it optimal for providing traceability by integration into FMI or SSP archives; and second, a python function library—contained within the in-house V&V automation framework—defining the verification activities, providing a static point that multiple workflows can operate against.

4.1 Resulting Workflow Representation

The structure and attributes of the resulting XML schema are described in this section along with the connection to the initial requirements.

Action Starting with the smallest artifact of the schema, the link to the automation framework library of predefined functions. It specifies a named reference inside the library, any additional fields within the scope are custom parameters that will be passed to the library function as seen in Listing 1. A function together with its parameters effectively creates templates for simplified re-use to reduce code duplication and maintenance overhead. Instead of rewriting actions for similar tasks, engineers can adapt templates by altering a few parameter values, making the verification process more flexible, modular, and cost-effective.

Listing 1. Activity example

```
<action function="simulation">
    <parameter_1 value="f"/>
</action>
```

Grouping mechanisms Due to the often computationally expensive operations involved in executing simulations, the speed-up factor of parallelizing very slow activities—such as simulations—can often be close to optimal since any overhead will almost always be much smaller than the actual simulation. Therefore, enabling parallelization is essential for scalability, where multiple simulations should be processed simultaneously. To facilitate this, two different grouping types were introduced: **sequential** (Listing 2) and **parallel** (Listing 3). These grouping types enable clustering of activities, the former defining a dependency chain of execution in sequential order and the latter that all activities within the group are independent and could be executed individually.

Listing 2. Sequential example, creating a sequential dependency chain between the actions, forcing execution in a specific order

```
<sequential>
    <action function="action_1"/>
    <! -- potential additional actions -->
</sequential>
```

Listing 3. Parallel example, all actions can be invoked independently of each other

```
<parallel>
    <action function="action_1"/>
    <! -- potential additional actions -->
</parallel>
```

Structure The overall structure is based on two main principles. First, a hierarchical structure that enables nested grouping operations, allowing for complex scenarios while maintaining coherence and clarity (see Listing 5). The second principle is dynamic adaptation, which allows any action to change or add actions downstream.

Dynamic adaptation has two main objectives, the first being to facilitate adaptation to unknown factors. For example, an action that searches for scenarios in a folder has no way to know how many, or what kind of, scenarios it will find before execution. To enable this, the action, after finding the scenarios, will make adjustments to downstream actions to reflect the new information, Listing 5 showcases a workflow that corresponds to this scenario, simulating and verifying results for each input data file it finds in a directory at runtime, the definition of the function called shown in Listing 4. The second objective of dynamic adaptation is to enable the workflow to take into account intermediate results such as failures or fulfillment of conditions and automatically branch to alternative paths. This adaptability is often connected to handling iterative tasks where the number of iterations may not be known beforehand.

Listing 4. Python signature of `find_files`

```
def find_files(path: str,
               pattern: str = "*.csv") ->
                   list[Node]:
    """Creates corresponding nodes to
        availible files that match *pattern
        * under *path*."""
```

Listing 5. Dynamic adaptation, adapting to the number of files found

```
<sequential>
  <action function="find_files">
    <path value="file_storage/"/>
  </action>
  <sequential>
    <action function="simulate"/>
    <action function="evaluate_results"/>
  </sequential>
</sequential>
```

Combining these two principles enable the building of almost any workflow, including generating scenarios and design variations, comparing results, and executing different actions depending on the outcome, all while maintaining the readability of both simple and complex workflows. In the end, the basis for enabling these requirements, while keeping the representation compact and understandable, was to outsource much of the complexity to the automation framework's function library; this by creating helper functions for advanced use cases. The ability to construct templates of common verification scenarios also creates reusability opportunities and limits code duplication between workflows.

Global parameters Outside the scope of the workflow, it is possible to place **global parameters**, as seen in Listing 6. Any number of global parameters can be created, and they will be propagated to all functions, this to enable some common understanding of context without explicitly feeding the parameters, for example model location, to each action. As seen in Listing 6, the top workflow definition consist of a single sequential or parallel grouping.

Listing 6. Workflow example

```
<VerificationWorkflow>
    <model value="./model_1"/>
    <sequential>...</sequential>
</VerificationWorkflow>
```

Limitations The order of execution is strictly downstream, meaning that each action can only be executed once in a specified sequence. This results in that an action should not create a downstream dependency to any action that is upstream. Data, on the other hand, can only be accessed upstream where a dependency exists to ensure a deterministic result. Figure 4 showcases how requirements regarding execution order, parallelization opportunities and data access relate.

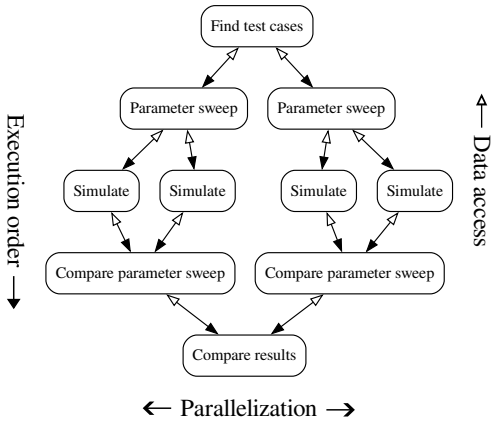


Figure 4. Showcasing execution order, parallelization possibilities and data flow

To mitigate some complexities related to merging branches executed in parallel, there is a defined limit to the change scope for dynamic adaptations. An action may only adjust actions or groups that are within its own grouping scope. For example, in Listing 7, **outer_action_1** can alter everything downstream, including the inner actions. However, when moving into the inner scope, the **inner_action_1** can only alter actions downstream within its grouping scope, more specifically **inner_action_2**.

Listing 7. Dynamic adaptation scope

```
<sequential>
    <action function="outer_action_1"/>
    <sequential>
        <action function="inner_action_1"/>
        <action function="inner_action_2"/>
    </sequential>
    <action function="outer_action_2"/>
</sequential>
```

4.2 Implementation

The V&V framework parses the workflow representation and constructs a corresponding Directed acyclic graph (DAG). Any actions grouped by a clustering mechanism are enclosed by start and end nodes to facilitate simpler graph manipulations. This design is chosen for the same reason that most programming languages use notations such as {} or tabs to clarify the beginning and end

DOI 10.3384/ecp218751

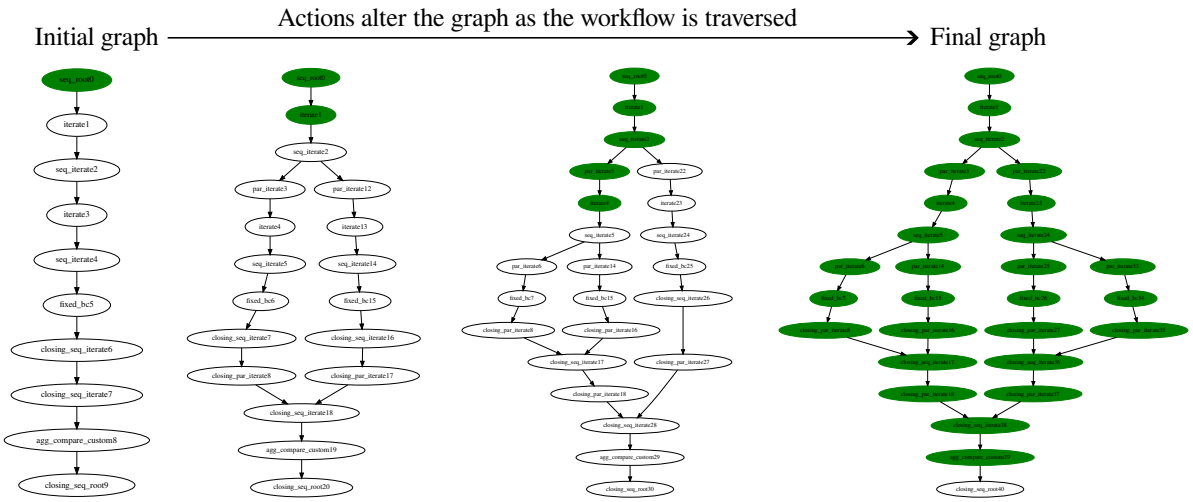


Figure 5. Graph adaptation during execution for use case 1. Filled node marks completed actions

of constructs. Although the start of a construct can often be inferred from context, it is often challenging to distinguish where the construct ends without a closing notation.

After finding the starting node, the application begins traversing the DAG downstream, morphing the graph as the actions execute, see Figure 5 for a description of how the graph alters after that actions are processed. To enable this, the full DAG is passed to each action for modification and when returned, it is merged back into the original DAG before continuing execution. The graph corresponding to use case 1 is showcased in Figure 5; the traceable artifacts of the activity and workflow representation are described as SSP Traceability artifacts in the Appendix, see Listing 8 and Listing 9.

4.3 Version Control

By integrating version control systems within model archives, each verification activity and its results could be linked to specific model versions, parameters, and configurations, thereby creating a comprehensive audit trail, see Figure 6 for a visualization of the concept. This enhanced traceability facilitated root-cause analysis in cases where discrepancies arose and provided a robust foundation for conducting credible and reproducible verification processes. But despite these successes, some limitations surfaced. Handling large binary artifacts within the version control system was challenging, inflating repository sizes and complicating storage. Future improvements might involve adopting specialized large file storage solutions or more effective strategies.

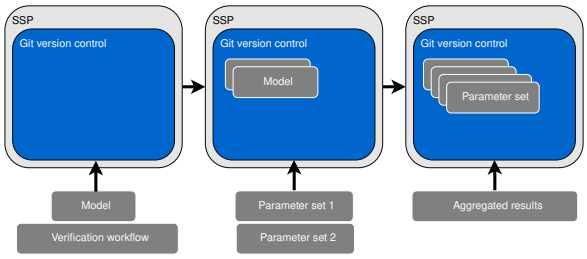


Figure 6. Showcasing the version control methodology, each step a snapshot in time enabling traceability of model creation and alterations

4.4 Verification

When applied to the use cases, the workflow consistently produced traceable and reproducible verification results. Stakeholders perceived the enhanced traceability as increasing the trustworthiness of verification outcomes. Preliminary evaluations suggested that the new workflow format offered clearer logic navigation and easier interpretation than previous solutions. The workflow representation was perceived to minimize the need for code duplication between projects. Additionally, the options for using templates for common functions to further simplify the workflow design are promising and warrant further evaluation.

5 Discussion

Workflow representation By comparing the verification method to established practices highlights some improvements. Traditional methods, such as the *Saab Aeronautics Handbook for Development of Simulation Models* (Andersson and Carlsson 2012), often rely on non machine readable, document-centric tracking, which may be error-prone and sometimes difficult to maintain. By consolidating workflow definitions and

model data into a consistent, machine-readable form, the proposed approach reduces human error, enhances discoverability, and aligns well with growing industry trends toward continuous integration and continuous delivery.

The function library contained within the automation framework offloads much of the complexity from the workflow representation and direct manipulation of the DAG, enabling a simpler representation of the workflow. Offering fine-grained control via arguments or flags enabled teams to adapt the workflow to various conditions without creating entirely new test definitions. However, this raises questions whether it might sometimes be too simplistic. For example, in Listing 5, it is not possible to determine how the function **find_files** will alter the downstream actions by only looking at the representation. This places requirements on the function library to provide clarity regarding what functions do and how they affect the workflow; Underscoring the need for intuitive defaults, comprehensive documentation, and clear guidance to ensure broad and consistent user adoption. It is identified that a possible mitigative action is to foster collaboration best practices regarding workflow adaptations among users.

During the design, the discussion rose early to include the possibility to let workflow designers work directly in python, skipping the markup schema. In some cases one is preferred, in others not. It almost always comes back to how to enable easy access for all stakeholders, not only developers, to the verification workflows. We identified that XML would align more closely with standards such as SSP and FMI but may in some cases be quite explicit provide less readability for use cases such as this. If a port to some other format would be deemed necessary in the future, XML is highly convertible into almost any other key-value pair format.

Implementation By breaking large verification tasks into parallel segments, total execution time was reduced; however, coordinating these tasks introduced new challenges in synchronizing data states and results. Many of these challenges relate to how branches should be merged; for example, how to handle situations where two branches have made different changes to future common activities. To cope with this, it was necessary to place restrictions on how dynamic adaptation functions in order to mitigate some complex cases. Prohibiting feedback loops in the graph is one such restriction that minimizes the chance of overwriting artifacts at consecutive invocations. Disallowing loops does lead to some complexity when taking iterative workflows into account, and creative workarounds are sometimes needed to achieve certain complex cases. For example, in iterative workflows to achieve conditional effects the solution could be to copy the desired actions to the end when a condition is triggered, thus creating the next iteration.

The chosen implementation solution for parallelization has created a high level of complexity when coupled to the ability for dynamic adaptation of the graph. The design decision not to use a central graph handler to orchestrate the parallelization, but to let parallel threads traverse the graph independently of each other, enables a very high degree of parallelization. In hindsight, most likely unnecessarily high since any simulation activities will be magnitudes slower, a clear case of premature optimization. The implication of the design is that each parallel branch has its own notion of the DAG. Without a central authority, the branches must negotiate the new notion of truth before continuing when merging branches.

Traceability The intention originally was to utilize Simulation Resource Meta Data (SRMD) for storing and the resulting workflow definition. After delving deeper into the definition of SSP Traceability, it was later adjusted to utilizing an external resource together with STMD. Both SRMD and STMD are connected to the **Credible simulation process**, but more fitting would be to have a stronger connection to the **credible modeling process** by utilizing Modeling Task Meta Data (MTMD). But at the moment this is not included into the released version of the standard and the content between the metadata formats are in large interchangeable, thus the choice went to using the released STMD

Embedding version control systems directly within the model archives offers a transparent, maintainable history that aligned well with established practices such as FMI/SSP. The resulting benefits, including easier reproduction of results, better understanding of model evolution, and stronger stakeholder confidence, generally outweighed the overhead and complexity of managing multiple branches, commits, and binary data within a repos-

DOI
10.3384/ecp218751

itory. Data management and storage considerations emerged as another critical aspect of the solution. Storing binary artifacts within version control systems allowed for a single source of truth and integrated traceability, but also increased repository sizes, potentially affecting performance and long-term maintainability. Strategies like introducing binary files early, employing artifact repositories or Git LFS, and limiting unnecessary duplication can mitigate these issues. Nonetheless, as verification activities scale up, organizations will need to carefully balance repository complexity against the benefits of having all artifacts co-located and tracked within the same environment.

6 Conclusion

This work has presented a method for managing model verification activities that meet the stated research objective of achieving traceability and repeatability in verification workflows, while ensuring readability and enabling automation. By preserving the complete history of verification activities, providing a transparent and evidence-based record that in the end supports more informed decision-making during and after development. Binding verification logic closely to model versions, harnessing parallel execution, and embracing dynamic adaptation, the approach supports more rigorous and scalable verification scenarios. By integrating traceable verification workflows with version control embedded into model archives, the approach provides a platform for managing complex and evolving verification processes.

The Modelica association standards FMI, SSP, and SSP Traceability serve as key enablers to the presented work. Relying on such standards allow the presented functionality to interplay, through portable and stand-alone simulation packages, with similar functionality provided in both open source and commercial tools, such as Dymola (Bruck 2023). Work to initiate this described interplay between testing frameworks and development tools has been initiated in the OpenSCALING project and at Saab through, for example, the work of Matstoms (Matstoms 2025). Matstoms realizes SSP Traceability, and automation, in an existing regression testing framework including model development tools in the loop, particularly Dymola, OpenModelica (Mengist et al. 2015), and easySSP (exxellent Solutions 2025).

The positive reception, strong performance gains, and improved user experience provide a solid foundation for ongoing refinements and broader integration into current software development practices, continuous integration processes, and simulation ecosystems that use standards such as FMI or SSP. As best practices and tools continue to evolve, integrated, traceable, and machine-readable verification workflows are likely to continue evolving and becoming more mature, in the end benefiting a wide range of simulation and modeling applications.

Acknowledgments

This research was conducted within the scope of the Advanced digitization ERATO projects as well as the ITEA4 OpenSCALING project, funded by Vinnova and Saab AB. The authors would also like to thank Dan Louthander for his work with the in-house verification framework.

References

Ahmann, Maurizio et al. (2022-11-22). “Towards Continuous Simulation Credibility Assessment”. In: *Proceedings of Asian Modelica Conference 2022, Tokyo, Japan, November 24-25, 2022*. Asian Modelica Conference 2022, Tokyo, Japan, November 24-25, 2022, pp. 171–182. DOI: 10.3384/ecp193171. URL: https://ecp.ep.liu.se/index.php/modelica/article/view/572 (visited on 2025-03-28).

Ahmed, R., M. Shah, and M. Umar (2016-06-15). “Concepts of Simulation Model Size and Complexity”. In: *International Journal of Simulation Modelling* 15.2, pp. 213–222. ISSN: 17264529. DOI: 10.2507/IJSIMM15(2)2.317. URL: http://www.ijsimm.com/Full_Papers/Fulltext2016/text15-2_213-222.pdf (visited on 2025-04-02).

Andersson, Henric and Magnus Carlsson (2012-12-12). *Saab Aeronautics Handbook for Development of Simulation Models : Public Variant*. 12/00159. Linköping University Electronic Press. URL: https://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-86281 (visited on 2025-03-26).

Arthur, J.D. et al. (1999-12). “Verification and validation: what impact should project size and complexity have on attendant V&V activities and supporting infrastructure?” In: *WSC’99. 1999 Winter Simulation Conference Proceedings. ’Simulation - A Bridge to the Future’ (Cat. No.99CH37038)*. WSC’99. 1999 Winter Simulation Conference Proceedings. ’Simulation - A Bridge to the Future’ (Cat. No.99CH37038). Vol. 1, 148–155 vol.1. DOI: 10.1109/WSC.1999.823064. URL: https://ieeexplore.ieee.org/document/823064/?arnumber=823064 (visited on 2025-04-02).

Balci, O. and W.F. Ormsby (2000-12). “Well-defined intended uses: an explicit requirement for accreditation of modeling and simulation applications”.

In: *2000 Winter Simulation Conference Proceedings (Cat. No.00CH37165)*. 2000 Winter Simulation Conference Proceedings (Cat. No.00CH37165). Vol. 1, 849–854 vol.1. DOI: 10.1109/WSC.2000.899883. URL: https://ieeexplore.ieee.org/document/899883 (visited on 2024-06-05).

Beisbart, Claus and Nicole J. Saam (2019-04-24). *Computer Simulation Validation: Fundamental Concepts, Methodological Frameworks, and Philosophical Perspectives*. Springer. 1088 pp. ISBN: 978-3-319-70765-5. URL: https://www.ebook.de/de/product/30290750/computer_simulation_validation.html.

Bouskela, Daniel et al. (2023-12-22). "The Common Requirement Modeling Language". In: *Modelica Conferences*, pp. 497–510. ISSN: 1650-3740. DOI: 10.3384/ecp204497. URL: https://ecp.ep.liu.se/index.php/modelica/article/view/960 (visited on 2025-01-08).

Bruck, D. (2023-09). "SSP in a Modelica Environment". In: *Proceedings of the Modelica Conference*. Linköping University Electronic Press. DOI: 10.3384/ecp204711.

Buschmann, Frank et al. (1996-08-16). *Pattern-Oriented Software Architecture Volume 1: A System of Patterns*. Volume 1 edition. Chichester: Wiley. 476 pp. ISBN: 978-0-471-95869-7.

Cederbladh, Johan, Antonio Cicchetti, and Jagadish Suryadevara (2024-03-31). "Early Validation and Verification of System Behaviour in Model-based Systems Engineering: A Systematic Literature Review". In: *ACM Transactions on Software Engineering and Methodology* 33.3, pp. 1–67. ISSN: 1049-331X, 1557-7392. DOI: 10.1145/3631976. URL: https://dl.acm.org/doi/10.1145/3631976 (visited on 2025-01-10).

Dalle, Olivier (2012-12). "On reproducibility and traceability of simulations". In: *Proceedings Title: Proceedings of the 2012 Winter Simulation Conference (WSC)*. 2012 Winter Simulation Conference - (WSC 2012). Berlin, Germany: IEEE, pp. 1–12. ISBN: 978-1-4673-4782-2. DOI: 10.1109/WSC.2012.6465284. URL: http://ieeexplore.ieee.org/document/6465284/ (visited on 2025-04-04).

Dask (software) (2025-01-12). In: *Wikipedia*. Page Version ID: 1268901352. URL: https://en.wikipedia.org/w/index.php?title=Dask_(software)&oldid=1268901352 (visited on 2025-04-07).

exxellent Solutions (2025). *Orchideo| easySSP, developing and simulating complex systems*. https://www.easy-ssp.com/. (Visited on 2025-07-01).

Hällqvist, Robert (2023). "On The Realization of Credible Simulations: Efficient and Independent Validation Enabled by Automation". PhD thesis. Linköping University, Division of Fluid and Mechatronic Systems. ISBN: 978-91-7929-597-4.

Jenkins (software) (2025-03-11). In: *Wikipedia*. Page Version ID: 1279860802. URL: https://en.wikipedia.org/w/index.php?title=Jenkins_(software)&oldid=1279860802 (visited on 2025-04-07).

Khleel, Nasraldeen and Károly Nehéz (2020-01-01). "Comparison of version control system tools". In: *Multidiszciplináris Tudományok* 10, pp. 61–69. DOI: 10.35925/j.multi.2020.3.7.

Matstoms, Axel (2025). "Automating Modeling and Simulation Verification Testing in a Credible Simulation Process: Leveraging SSP and SSP Traceability". MA thesis. Department of Computer and Information Science, Linköping University.

Mengist, A. et al. (2015-09). "An Open-Source Composite Modeling Editor and Simulation Tool Based on FMI and TLM Co-Simulation". In: *Proceedings of the 11th International Modelica Conference*. Linköping University Electronic Press. DOI: 10.3384/ecp15118181.

Modelica Association (2022-10-27). *SSP Traceability Specification. Version 1.0.0-Beta2*. Standard, unreleased.

Muşlu, Kıvanç et al. (2014-05-31). "Transition from centralized to decentralized version control systems: a case study on reasons, barriers, and outcomes". In: *Proceedings of the 36th International Conference on Software Engineering*. ICSE 2014. New York, NY, USA: Association for Computing Machinery, pp. 334–344. ISBN: 978-1-4503-2756-5. DOI: 10.1145/2568225.2568284. URL: https://doi.org/10.1145/2568225.2568284 (visited on 2025-04-03).

Nallaperumal, Krishnan and Annam Krishnan (2013-12-07). *Engineering Research Methodology A Computer Science and Engineering and Information and Communication Technologies Perspective*.

Oberkampf, William L. and Christopher J. Roy (2014-01-13). *Verification and Validation in Scientific Computing*. Cambridge University Press, pp. 378–381. 790 pp. ISBN: 978-0-521-11360-1. URL: https://www.ebook.de/de/product/11824763/william_l_oberkampf_christopher_j_roy_verification_and_validation_in_scientific_computing.html.

Potts, C. (1993-09). "Software-engineering research revisited". In: *IEEE Software* 10.5, pp. 19–28. DOI: doi: 10.1109/52.232392.

Rosenlund, Erik et al. (2024). "Objectively Defined Intended Uses, a Prerequisite to Efficient MBSE". In: *Modelica Conferences*, pp. 29–42. ISSN: 1650-3740. DOI: 10.3384/ecp20729. URL: https://www.ecp.ep.liu.se/index.php/modelica/article/view/1128 (visited on 2025-04-02).

Standard for models and simulations (2024). NASA STD-7009B.

Tran, Anh et al. (2022-04-01). "aphBO-2GP-3B: a budgeted asynchronous parallel multi-acquisition functions for constrained Bayesian optimization on high-performing computing architecture". In: *Structural and Multidisciplinary Optimization* 65. DOI: 10.1007/s00158-021-03102-y.

Appendix

Listing 8. Simulation task metadata, verification task for use case 1

```
<stmd:SimulationTaskMetaData>
  <stmd:EvaluationPhase>
    <stmd:AssureSimulationQuality id="eval_step_01" description="Evaluate simulation results
        against requirements">
      <stc:Input>
        <stc:Resource kind="system" source="../../SystemStructure.ssd" type="application/x-ssp
            -definition"/>
        <stc:Resource kind="testcase" type="application/xml" source="eval_task_01.xml"
            description="Evaluation task definition"/>
      </stc:Input>
      <stc:Procedure>
        <stc:Resource kind="method" type="application/xml" source="procedure.xml" description=
            "Evaluation procedure document"/>
      </stc:Procedure>
      <stc:Output>
        <stc:Resource kind="result" type="application/xml" source="results/evaluation_report.
            xml" description="Evaluation report"/>
      </stc:Output>
      <stc:LifeCycleInformation>
        <stc:Validated date="2025-04-11T09:30:00+01:00">
          <stc:Responsible organization="Team A" name="John Doe" role="Evaluation Engineer"/>
        </stc:Validated>
      </stc:LifeCycleInformation>
    </stmd:AssureSimulationQuality>
  </stmd:EvaluationPhase>
</stmd:SimulationTaskMetaData>
```

Listing 9. Verification workflow use case 1. Referenced above as eval_task_01.xml

```
<VerificationWorkflow>
  <model value="./model_1"/>
  <sequential>
    <action function="find_test_cases">
      <path value="./test"/>
    </action>
    <sequential>
      <action function="parameter_sweep">
        <parameter_name value="parameter_1"/>
        <values value="5, 10, 15"/>
      </action>
      <sequential>
        <action function="simulate"/>
      </sequential>
      <action function="compare_parameter_sweep"/>
    </sequential>
  </sequential>
  <action function="compare_results"/>
</VerificationWorkflow>
```

Listing 10. Verification workflow as Python for use case 1

```
common_parameters = {"model": "./model_1",}

simulate_seq = Sequential(
    Action("simulate")
    )
sweep_seq = Sequential(
    Action("parameter_sweep", parameter="parameter_1", values=[5, 10, 15]),
    simulate_seq,
    Action("compare_parameter_sweep")
    )
top_seq = Sequential(
    Action("find_test_cases", path="./test"),
    sweep_seq,
    Action("compare_results")
    )

result = Test_Framework(parameters = common_parameters, seq=top_seq)
```


DOI
10.3384/ecp218751

Integration of Geometric Tolerance Analysis in System Simulations via Functional Mock-up Units

Christoph Steinmann[1] Konstantin Wrede[2] Jens Schirmer[1] Jens Lienig[1]

[1]Institute of Electromechanical and Electronic Design, Dresden University of Technology, Germany
christoph.steinmann@tu-dresden.de
[2] Fraunhofer Institute for Integrated Circuits IIS, Division Engineering of Adaptive Systems EAS, Dresden, Germany
konstantin.wrede@eas.iis.fraunhofer.de

Abstract

This paper presents a concept for integrating geometric tolerance analysis into system simulations using Functional Mock-up Units (FMUs). While various methods and tools for tolerance analysis exist in the mechanical domain, there is currently no standardized or widely established approach for their integration into multi-domain system simulation. This work proposes a structured FMU interface based on the Functional Mock-up Interface (FMI) standard, enabling a modular and reusable representation of tolerance behaviour. The concept is demonstrated in a case study, in which an FMU for static tolerance calculation was implemented and successfully verified against a commercial analysis tool. Furthermore, the use of the FMU with FMUGym illustrates the potential of the FMI ecosystem to flexibly combine tolerance models with other simulation environments and analysis methods. The results demonstrate that FMUs can provide a suitable and tool-independent interface for integrating geometrical tolerance effects into system-level simulations and model-based engineering processes.

Keywords: geometric tolerances, ISO GPS, tolerance analysis, MBSE, system simulation, Functional Mock-up Interface

1 Introduction

Geometrical tolerances of mechanical parts, in particular shape and position tolerances, play a central role in the precise definition of technical systems. They are defined by the geometric product specifications (GPS). Although there are already numerous simulation approaches for these tolerances in the mechanical domain, there is a lack of standardised approaches for integrating them into multi-domain simulations which are used for Model Based System Engineering (MBSE). The aim of this work is therefore to investigate a possible solution to enable the integration of geometrical tolerances for such simulations and to evaluate Functional Mock-up Units (FMU) (Blochwitz et al. 2011) as a possible interface. FMUs could serve as the connection between commercial Computer-Aided Tolerance (CAT) management programs or other analysis models and the system simulation, as shown conceptually in Figure 1.

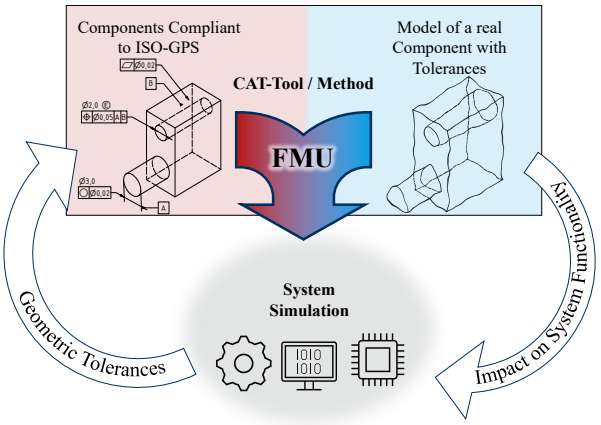


Figure 1. Our concept for the integration of geometric tolerance simulations into the product development process with system simulation via FMUs, influenced by ISO-GPS (red) and the modelling method used (blue).

Several challenges show the necessity of such an approach. Firstly, CAT with commercial tools is often regarded as a black box by the user, as the underlying methods are not accessible in detail (Kosec, Skec, and Miler 2020). This makes it difficult to adapt the models to new applications under changed boundary conditions. In addition, there is still no standardised methodology for setting up mechanical tolerance simulations, which impairs the comparability and reproducibility of results (Morse et al. 2018). The usability of existing scientific solutions also poses a problem, as many approaches are either too complex, not accurate enough for certain problems or not sufficiently user-friendly (Walter et al. 2021). There are also approaches from the field of MBSE to solve geometric problems using library elements (Moers et al. 2025). However, these are much less flexible and less precise than tolerance simulations that are specifically tailored to the problem of mechanical part geometry.

Precise tolerance simulation requires expert knowledge and a deep understanding of the respective domain (Ramnath et al. 2018). Modularisation of the modelling task therefore appears to be a promising approach to reduce complexity and facilitate access for users.

This work focuses on the simulation of rigid mechanical assemblies, with the aim of enabling static tolerance

analysis in the context of system simulation - similar to existing commercial tools such as *CETOL*, *MECAmaster*, or *3DCS*. While dynamic and transient simulations of tolerances have been addressed in initial research (Husch and Walter 2024), this paper deliberately focuses on static scenarios as a first step. Future extensions could build on these results to support more complex, dynamic applications.

The main contributions of this paper are:

- An analysis of the state of the art in tolerance simulation methods, with a focus on their interfaces and applicability within system simulation.
- The introduction of a new FMU-based concept for transferring specialised tolerance analysis methods to system-level simulations, demonstrated through a case study.
- The demonstration of additional use cases for FMUs within the FMI ecosystem, illustrated by performing a Monte Carlo simulation using the FMUGym toolbox.

2 Related Work and Methods

In order to effectively implement a mechanical tolerance model, it is essential to first consider both the standards and regulations that define geometric specifications, as well as the scientific methods used to simulate and analyse them. The relevant aspects of FMUs in relation to this application are important as well and thus, are also summarized below.

2.1 Mechanical Tolerance Analysis

Mechanical tolerance simulation in this paper's context specifically refers to the simulation of geometric deviations in components arising from their dimensions (size), shape (form), orientation, position (location), profile, and run-out in relation to one another. The underlying standard is the ISO-GPS system (ISO International Organization for Standardization 2011). The unique aspect here is that these tolerances go beyond pure scalar dimensions and therefore require special calculation methods for linking several tolerances.

Before the established analysis models are presented, two terms that are equally relevant for all of the methods need to be explained:

- *Functional Requirement (FR):* This is sometimes also referred to as a functional condition and describes the geometric entity that is relevant for the correct function of the assembly in the technical application and has to stay inside of certain limits. Depending on the calculation method used, this describes either a scalar (a distance) or the three-dimensional boundary of a resulting position in space. Defining this condition is the task of the user for all available methods. There are approaches for systematically or even automatically determining this condition, which can be used when solving specific tasks (Anselmetti 2006). FRs still have to be interpreted afterwards. Frequent target variables are, for example, the clearance of a fit or the misalignment in an assembly. The FR is an edge in the contact graph.
- *Contact Graph:* Or assembly graph is used to model the network of mating contacts and constraints between features in an assembly. This graph defines the relationships that form the tolerance chains - the paths through which individual part variations accumulate to affect a functional requirement. In commercial CAT software, the contact graph can be derived from the 3D CAD model, sometimes with the help of additional user input. For manual analyses according to the methods presented below, it must be created by the user, for which systematics are suggested in the literature (Haghighi et al. 2014).

The following paragraphs describe six analysis methods as well as groupings of tolerance models with their primary literature sources. There exist various modifications of most of the methods, for which numerous reviews (Hallmann, Schleich, and Wartzack 2020; Chen et al. 2014; Cao, Liu, and Yang 2018; Shah et al. 2007; Shen et al. 2005) provide a good overview. A summary of the most relevant aspects can be found in Table 1.

Vector Loop
(Chase et al. 1996)
The Vector Loop method models assemblies as closed kinematic loops, using vectors to represent nominal dimensions and constraints for mating conditions between parts. Tolerances are represented by small adjustments at the joints. Nominal dimensions, tolerance values, and assembly constraints are required as inputs. The calculated result is a point to point distance, which is interpreted as the FR. The method is able to calculate worst-case as well as statistical results with consideration of input-distributions. It can handle serial tolerance chains well but is not capable to represent complex, parallel tolerance chains in its basic form due to its limited ability to model intricate geometric interactions.

Matrix TTRS
(Desrochers and Riviere 1997)
The Matrix TTRS (Technologically and Topologically Related Surfaces) method uses homogeneous transformation matrices to model tolerances as small permissible displacements. It requires the nominal geometry, tolerance zone constraints, and assembly connections. The output is a range of variation in a critical assembly dimension based on solving inequality constraints. This represents the worst case calculation. It can theoretically handle complex parallel tolerance chains, but solving large constraint sets is computationally expensive, making it impractical for very detailed assemblies.

DOI
10.3384/ecp218765

Table 1. Overview of tolerance analysis models with their relevant characteristics.

Method / Model	*Result*	*Tolerance Representation and Calculation*	*Comment*
Vector Loop	distance, point to point	solution for joint conditions, concatenation of linearized transformation matrices	point based method (not well suited to represent form deviations), no parallel chain calculations
Matrix TTRS	distance, point to point	TTRS with non-linear constraint inequalitys	point based method, statistics only from Monte Carlo
Jacobian Torsor	interval of FRs position and rotation (6D)	multiplication of jacobian matrix with SDT (interval arithemtic)	partly represents form tolerances, handles certain parallel chains
Deviation-Domain	boundaries of tolerance zone in deviation space (6D)	coordinate transformation and polytope calculation	good compatibility with ISO-GPS, complicated implementation
Parametric Variation & Skin Model Shape	one or multiple measurements with statistic distribution	variation of parameters and 3D-contact solving with solids or mesh bodies	requires a sophisticated solution to solve the 3D contact problems

Jacobian Torsor
(Desrochers, Ghie, and Laperrie‘re 2003)
The Jacobian Torsor method combines Jacobian matrix transformations with Small Displacement Torsors (SDTs) to represent tolerances in six degrees of freedom. It requires assembly constraints, SDT representations of tolerances, and a functional requirement definition. The output is the variation range of a key dimension, computed using interval algebra. It handles complex and to some extent also parallel tolerance chains, making it usable for larger assemblies while maintaining computational efficiency.

Deviation Domain Methods
(T-Map (Davidson, Mujezinovic, and Shah 2002) and Polytope (Homri, Teissandier, and Ballu 2015))
The methods working in the Deviation Domain explicitly compute the geometric region of variation using operations on the tolerance space, which represents the possible translation and rotation of a feature (6D). The required input includes tolerance zones, assembly constraints, and mating feature definitions. The output is a geometric volume of possible variations, enabling worst-case and statistical analysis. It can theoretically handle parallel tolerance chains but suffers from exponential computational complexity, making it suitable for small critical assemblies rather than large systems.

Parametric Variation
(Gupta and Turner 1993)
The Parametric Variation method applies Monte Carlo simulation to a parametric CAD or analytical model, varying dimensions based on statistical distributions. It requires a parametrised assembly model, tolerance distributions, and assembly constraints. The output is a probabilistic distribution of functional variation. It can handle complex and parallel tolerance chains within the limits of the used geometric contact solver (for parametric CAD) but may require thousands of runs, making it computationally expensive.

Skin Model Shape
(Schleich et al. 2016)
The Skin Model Shape method (SMS) simulates tolerance effects by generating randomized part surfaces, assembling them virtually, and measuring the outcome. It requires a detailed CAD model, tolerance distributions, and assembly constraints. The output is a statistical distribution of assembly variations, considering realistic form deviations. It can handle complex and parallel tolerance chains, but the computational burden is extremely high, making it impractical for routine design work but useful for high-precision simulations. It can be considered a further development of the parametric variation approach.

All analysis methods have in common that they take information from both the nominal geometry (3D model) and the defined tolerances as input - including positions, interactions according to the contact graph, tolerance types and tolerance zone widths. The output is an FR in one or more dimensions, either as a scalar value or as a value range.

2.2 FMU

An FMU is a self-contained executable file that facilitates integration and simulation within various development environments that adhere to the FMI standard (Blochwitz et al. 2011), such as Modelica based tools or MATLAB Simulink. The idea behind the FMI standard is to enable comprehensive digital modelling of complex systems. If a real-world product consists of multiple interacting com-

ponents, each governed by intricate physical laws and associated control systems - including electronic, hydraulic, and software-based controls - then it should likewise be possible to digitally assemble a virtual counterpart from individual models representing these subsystems. Each FMU is packaged as a zip archive comprising an XML file with static meta-information, and a C library used for evaluating model equations and their derivatives. This library can be provided in source form or as a dynamically linked compiled library (DLL), and is suitable either for static or dynamic linking. FMI defines two primary forms of interconnection: model exchange, in which interactions occur at the level of a differential-algebraic system of equations (DAE), and co-simulation, which handles discrete input-output interactions at specific communication time points. The FMI standard 3.0 further introduces the scheduled execution format supporting concurrent computation of model partitions on a single computational resource (e.g. specific CPU-core). Although primarily designed for transient simulation scenarios, the FMI standard has been applied in previous research to perform steady-state (design-oriented) simulations, typically involving static single-step computations (Gohl et al. 2024). Similar to our intended use-case, FMUs have also been used as transfer interfaces for three dimensional FEM models (Gödecke et al. 2012).

While the FMI standard and FMUs are well suited for modular modelling and simulation, integrating uncertainty handling and automation features remains an active area of development. The Modelica Credibility Library (Otter et al. 2022), for instance, offers structured mechanisms to represent uncertainty directly within Modelica models and supports traceability, credibility assessment, and transparent documentation of assumptions. However, it has not yet been used in our concept for two main reasons: first, the library lacks interfaces for automated simulation workflows or integration with Python-based environments; second, the current data is derived solely from predefined GPS, so detailed traceability is not yet a priority. Recent contributions such as FMUGym (Wrede et al. 2024) have extended FMI applicability to machine learning scenarios. FMUGym is a Python-based interface that supports reinforcement learning-based control of FMUs and accommodates model uncertainty. It handles co-simulation FMUs compatible with FMI 1.0, 2.0, and 3.0, linking simulation environments with reinforcement learning libraries. In a simplified manner, FMUGym also allows a standardized evaluation of simulation model outputs under specified input distributions and varying system parameters. This makes it suitable for analysing the tolerances of technical systems.

3 Proposed Concept for Geometric Tolerance FMUs

This chapter presents how FMUs can be set up for the simulation of geometric tolerances of mechanical components

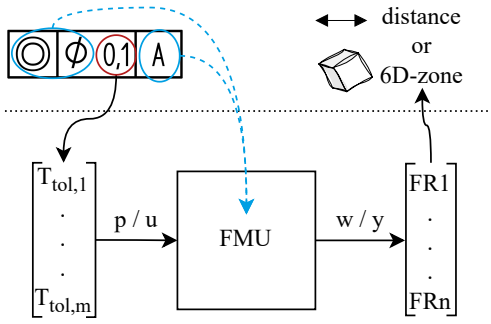


Figure 2. FMU Concept: Tolerance zone size T_{tol} from ISO feature control frame as input and Functional Requirement FR as output depending on the method used for tolerance calculation.

and which interfaces they provide. The actual computation of the tolerance values is not subject of this paper, as this depends heavily on the selected method and the considered application - a brief explanation and assessment of the suitability of various methods can be found in subsection 2.1. The goal is rather to formulate a generic specification of interfaces in line with the FMI standard, in particular with regard to possible inputs and outputs as shown in Figure 2. For the static case considered in this paper, both parameters (p) and input variables (u), as well as local variables (w) and output variables (y), can be treated equivalently to represent mechanical tolerances. It should be noted, that a differentiation between these categories would be necessary in the context of dynamic simulations (e.g. with external forces).

The idea for structuring the interfaces is based on the elements of the tolerance feature control frame according to ISO-GPS. The scalar values of the tolerance field variables are provided as input variables - i.e. the numerical value in the second box of the ISO tolerance field (see Figure 2). Further information, such as the specific tolerance type, geometric datum references or other model-dependent details, are only included in the internal structure of the FMU. This separation makes it possible to keep the external interface stable and generic, while the internal modelling can be flexibly adapted to different analysis methods. The FMU must also be capable of modelling several FRs at the same time, as complex systems can rarely be reduced to a single FR. This results in the need to integrate several analysis models within an FMU. To the outside world, each tolerance defined in the CAD-model or drawing only appears as a single parameter T_{tol} representing the size of the tolerance field. An exemplary realisation of this structure can be found in the case study in section 4.

The output of the FMU are the individual FRs, which can take different forms depending on the model type and analysis objective. In the simplest case, the output is a scalar, e.g. to describe a distance between two points. However, even in these simple approaches, the result is typically given as a range with statistical or worst-case limits for this distance - effectively requiring two output variables per FR to represent minimum and maximum

DOI
10.3384/ecp218765

values. For more complex models such as the Jacobian torsor or the deviation domain, deviation ranges or three-dimensional spaces can be expected as outputs. The actual reduction to interpretable individual values, such as worst-case values, then proceeds outside the FMU. Probability distributions could also theoretically be the output. However, we suggest to carry out the actual statistical evaluation outside of the FMU, as the FMU should only provide the geometric tolerance calculation.

The choice of FMUs as the basis for this approach is suitable in particular because they allow modular, standardised integration into existing simulation environments and promote the reusability of complex analysis logic.

4 Case Study

The following case study shows an example of how a geometric tolerance simulation can be integrated into an FMU and subsequently used as such. The starting point is the analysis of an assembly with the commercial simulation tool CETOL. Based on this analysis, an FMU is created that calculates the same FRs as the reference analysis internally and uses the interfaces described in section 3.

4.1 Application Example

The example shown in Figure 3 shows a simplified model of a stepper motor with translational output. A magnetic field is generated in the stator (S), which causes the magnetic rotor (R) to rotate. The rotor is held between two balls (B) and drives the nut (N) via a spindle (not shown here), which converts the rotary movement into a linear movement. The assembly is completed by the end cap (C), which also serves as a linear bearing for the nut.

The contact graph in Figure 4 illustrates the relationships between the individual components and contains the labels of all relevant connection points, whose interactions are explained in more detail in the following subchapter. The values and types of tolerances are listed in Table 2.

The study focuses on the gap widths $FR1$ and $FR2$ marked in Figure 3. $FR1$ describes the air gap between the stator and the rotor through which the magnetic field flows, while $FR2$ relates to the mechanical fit of the linear bearing on the output. The upstream analysis in CETOL also serves to identify those tolerances within the defined mechanical chain that have a relevant influence on the FR under evaluation. Parameters that are not categorised as relevant in this analysis are not taken into account in the subsequent FMU implementation. This approach is useful as only the relevant parameters should be included, even in the event of a future automated FMU export. For the present case study, this reduces the manual calculation effort when implementing the FMU.

The following additional boundary conditions for the implementation of the FMU result from the example application and the setup of the CETOL analysis. These are particularly relevant in regard to the comparison of the calculations implemented here with the previously presented analysis methods from the literature:

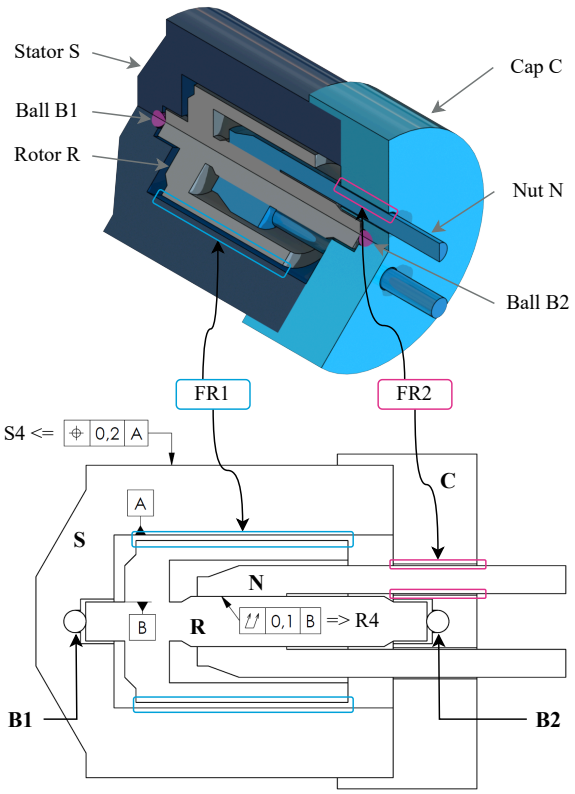


Figure 3. CAD model of a stepper motor with component labels and FR positions used for the case study. The assignment of the labels for the geometric tolerances is shown as an example for S4 and R4.

- Rigid solids without deformation are assumed.
- Contacts between components only occur at defined connection points.
- Shape tolerances are not fully mapped in accordance with ISO standards.
- The assembly sequence specifies manually defined kinematic dependencies between the components.
- Parallel tolerances are simplified by making assumptions about their degrees of freedom.

4.2 Implementation as FMU

In this example, the analysis of the mechanical tolerances is limited to a 2D plane. This simplification is acceptable due to the predominantly rotationally symmetrical components; minor deviations in the non-rotationally symmetrical nut (N) and end cap (C) are accepted for the benefit of clarity. It is not the scope of this paper to demonstrate how complex 3D analysis methods can be automatically translated into executable code, but rather to show the basic process of using an FMU to transfer the tolerance analysis model. Therefore, the internal content of the FMU must still be manually created and verified. A simplified case such as the one presented here is particularly beneficial

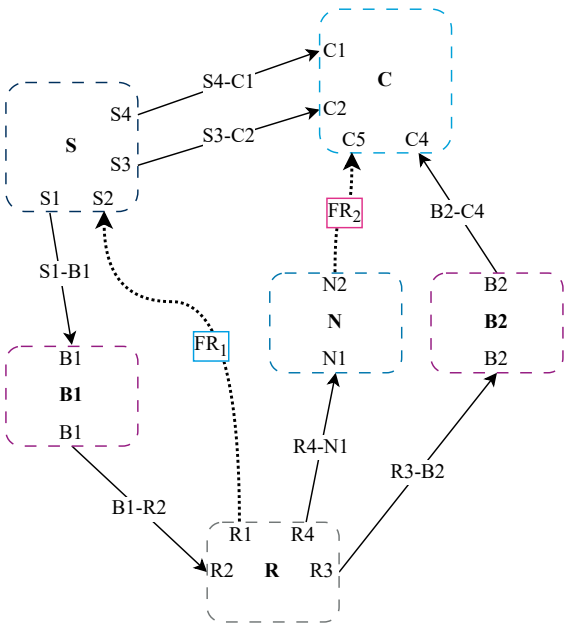


Figure 4. Contact graph of the motor with part labels and links between tolerances

in this context, as it allows for transparent modelling and easier validation of the implemented logic.

As described in subsection 2.1, the existing analysis methods differ fundamentally in their calculation logic. Accordingly, the internal code within an FMU also varies depending on the underlying method. For this case study, the following procedure was chosen to build the FMU:

1. Analysation of the tolerances and joints defined in CETOL
2. Derivation of the kinematic chain and constraints
3. Analytical calculation of the influence of all part tolerances on the investigated FR
4. Implementation of the resulting equations as an Modelica algorithm, including the summation of the final results
5. Export of the Modelica model as FMU

Table 2. Type and size of tolerance zone T_{tol} of all tolerances relevant for calculating the FRs

Tol	*Type*	T_{tol}	*Kinematics*	*FR*
R1	diameter	0.1 mm	linear	$FR1$
S2	diameter	0.1 mm	linear	$FR1$
S4	position	0.2 mm	tilt	$FR1$, $FR2$
C4	profile	0.1 mm	tilt	$FR1$, $FR2$
C5	diameter	0.2 mm	linear	$FR2$
R4	runout	0.1 mm	tilt	$FR2$
N2	diameter	0.1 mm	linear	$FR2$

In combination with the previously defined boundary conditions, the approach is similar to the vector loop method. To simplify the implementation, a solver algorithm for determining the linking states is not used. Instead, all relevant cases are solved analytically and are integrated into the FMU as equations.

For $FR1$ and $FR2$, this results in a compact set of equations for the relevant tolerances listed in Table 2. R1 and S2 describe the simple linear variation of the air gap $FR1$ on both sides. The same applies to C5 and N2 with regard to the fit of $FR2$. The tolerance variations of C4 and S4 result in different tilting movements of the kinematic chain, which influences both FRs. R4 is also a tilting move only affecting $FR2$.

To determine the resulting worst-case tolerance, the effects on the target dimension are calculated separately for each individual tolerance along the kinematic chain - as shown below. Finally, all partial solutions FRn_i are summed up and added to the nominal value of FRn.

Our calculation is based on the radius (y-axis in Figure 5), which is why all diameters are halved. In the linear cases (see Table 2), the tolerances in this example - due to the symmetrical tolerance zones - are each taken into account in the calculation with half the value of their tolerance zone width. The individual contributions thus result directly from the width of their corresponding tolerance zone T_{tol}:

$$FRn_{i,max} = +T_{tol}/4 \tag{1}$$

$$FRn_{i,min} = -T_{tol}/4 \tag{2}$$

The rotational deviations can be derived geometrically from the kinematic joints by combining several vectors from the loop. The tilting cases with relevant geometric parameters for deriving the equations are shown in Figure 5.

Changes in the tolerance of C4, as defined by the joint configuration, result in a tilting of the rotor R around the contact point at B1 (Point O in Figure 5), caused by moving B2 by $t = T_{C4}$ along the y-axis. Under this assumption, the tilt angle can be calculated:

$$\alpha_{C4} = \arcsin\left(\frac{t}{l_R + r_B}\right) \tag{3}$$

Changes in R4 are a simple rotation of part N on part R around the center of the cylinder axis. This results in a direct influence of the tolerance zone rotation on the final result. The total rotation angle is determined by the characteristic length c_{R4}, which corresponds to the length of the cylindrical surface defined by the geometry of part R for the runout tolerance R4 (see Figure 3):

$$\alpha_{R4} = \arctan\left(\frac{T_{R4}}{c_{R4}}\right) \tag{4}$$

DOI
10.3384/ecp218765

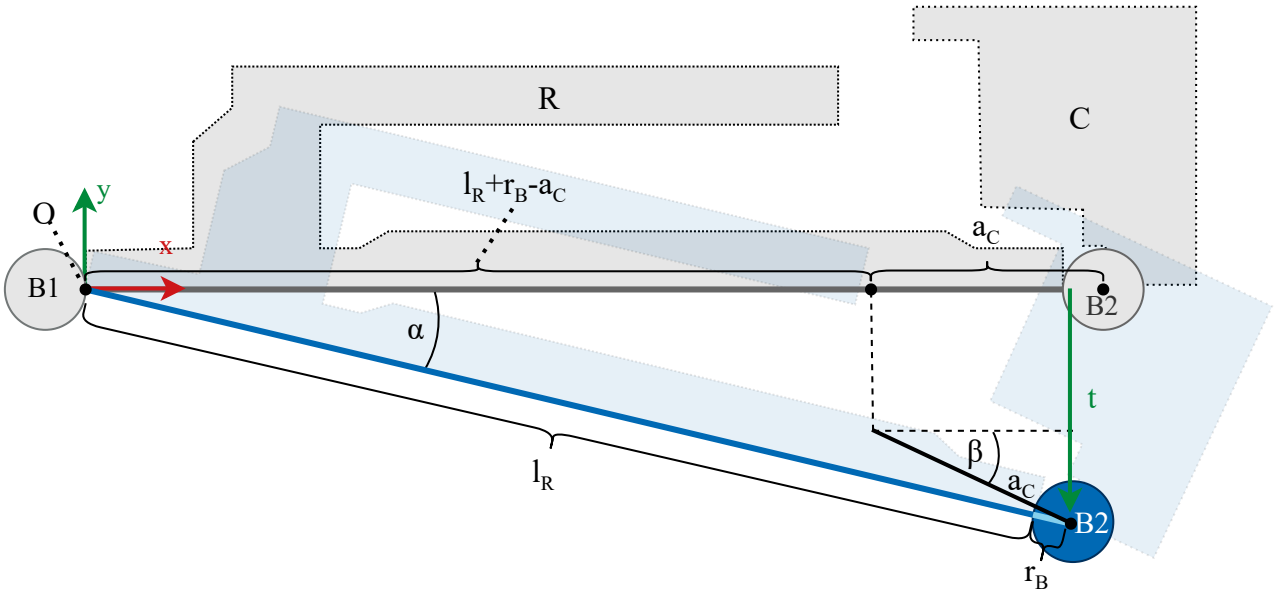


Figure 5. Geometric relationships for calculating the tilt angle α of the rotor between the bearings B1 and B2

Changes in the tolerance of S4 result in a tilting of the rotor R around the contact point at O at Ball B1, as well as a tilting of the cap C around the ball B2. During these movements, contact between the cap C and the stator S in the x-direction is maintained (with the contact defined at a fixed distance a_C). The tolerance variation affects the angle β in this case, which is calculated by using the tolerance zone width T_{S4}: $\beta_{S4} = \arctan(T_{S4}/c_{S4})$. The characteristic feature length c_{S4} can be derived from the geometry of part S. S4 defines the tolerance of the outer cylindrical surfaces shown in Figure 3. Corresponding to Figure 5, α_{S4} can then be determined using this equation:

$$\alpha_{S4} = \arccos\left(\frac{a_C \cdot \cos(\beta_{S4}) + l_R + r_B - a_C}{l_R + r_B}\right) \quad (5)$$

The tilt angles α can be used to determine the change of FRn_i in relation to these measuring points via the position vector of the FRn. For $FR2$, it should be noted that in the case of S4 the reference point for measurement in part C is also rotated by the angle β_{S4}.

Finally, the partial contributions determined in this way are summed up:

$$FRn_{max/min} = FRn_{nominal} + \sum FRn_{i,max/min} \quad (6)$$

The values of FRn_{max} and FRn_{min} calculated in this way form the output of the FMU for this example. The equations are transferred to a Modelica model (Appendix Listing 1) and subsequently exported as an FMU using OpenModelica.

4.3 Results and Verification

A comparison of the results calculated with the FMU and CETOL is shown in Figure 6. The worst-case calculation is performed directly within the FMU based on a predefined set of model inputs, which correspond to those used in the CETOL analysis. The values calculated by the method described above are very close to the result of CETOL with deviations from the worst case values of less than 3%. These minor deviations result from the fact that the assumptions made in this paper regarding the kinematic chain do not exactly reproduce the specific behaviour of the numerical solver used for the joint configuration in CETOL. This FMU can now be integrated into a system simulation to calculate the two FRs and the influence of the mechanical tolerances can be analysed across domains.

To further demonstrate the applicability of the FMU and at the same time present a complementary verification method for the results, additional testing was carried out using FMUGym as a feasible tool for this purpose. The FMU implementation presented above calculates only worst-case values and does not perform any statistical analysis internally. Therefore, a Monte Carlo simulation with 1000 samples was conducted with FMUGym to compare the statistical results with those from CETOL. As in the reference analysis, a process capability index (Cpk) of 1.0 (equivalent to $\pm 3\sigma$ limits) was assumed for all input variables, with random variation around the mean value following a normal distribution.

The results of this Monte Carlo simulation are also plotted in Figure 6. It can be seen that the statistical results from the FMU simulation also match the reference values from CETOL very well. While CETOL computes statistical parameters and sensitivities using the method of moments, based on first-order approximations derived from the partial derivatives with respect to individual variables, our approach leverages Monte Carlo simulation, which may lead to slight differences in the results.

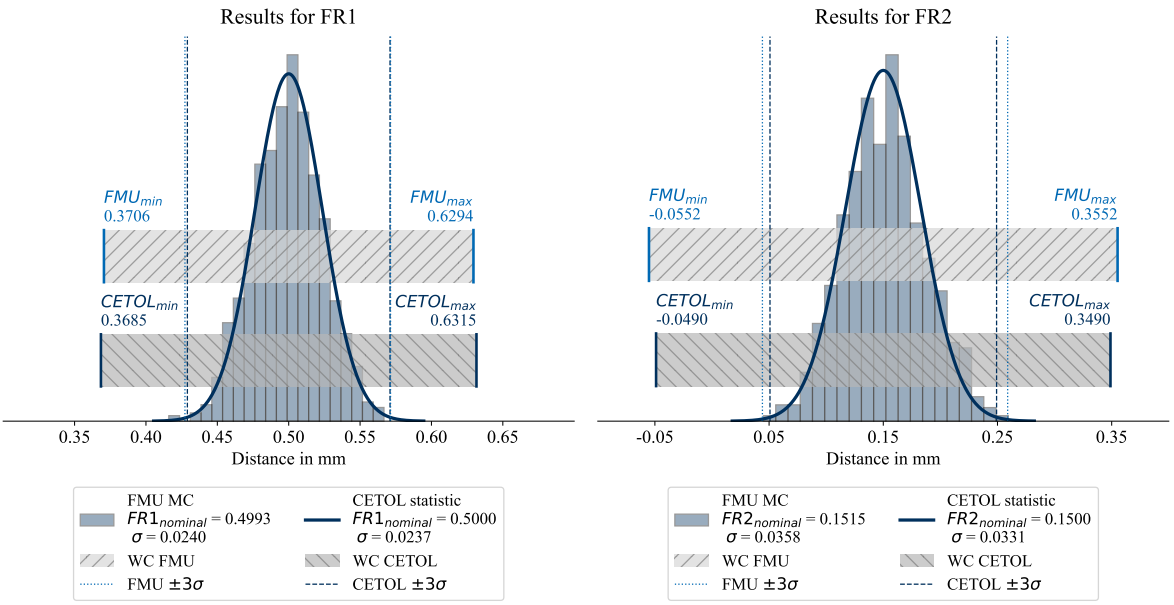


Figure 6. Results of FMU for Worst Case (WC) analysis and Monte Carlo Simulation (MC) as well as the results of the analysis with CETOL. It can be seen that both WC and statistical results of the two methods are close to each other. The deviations result from the assumptions regarding the kinematic chain and the different approaches to determine the statistical results.

5 Discussion

5.1 Evaluation of the Implemented Approach

In the presented case study, the FMU was successfully used to perform a Monte Carlo simulation via the external tool FMUGym. While this demonstrates one specific application scenario, the general concept of using FMUs for tolerance analysis is not limited to this context. All functionalities of FMUs - ranging from simple parameter sweeps to integration in full system simulations - remain applicable.

The FMU developed in this work focuses solely on computing geometric quantities related to the defined FRs. However, in a broader system simulation, these results could be interpreted and used across domains. For example, as described in the case study, the air gap ($FR1$) influences torque generation in the magnetic circuit of the stepper motor, while the guide clearance ($FR2$) can introduce friction forces at the mechanical output. These interactions imply how geometrical tolerances can have a measurable impact on system-level behaviour and performance.

5.2 Usefulness of the FMU Approach

The use of FMUs enables seamless integration of tolerance analysis into complex system models, facilitating cross-domain simulation workflows. This modularization allows tools like FMUGym or Modelica-based environments to interact with geometrically defined tolerances without reproducing detailed internal logic on the system level. The FMU outputs can thus be interpreted either in purely geometrical terms or as system-relevant input variables, depending on the simulation context.

The benefits of standardized, automatable interfaces for solving geometry-related problems in 3D system models has also been supported by recent literature in the field of MBSE (Lehner et al. 2025).

In the future, FMUs offer significant potential for extended applications, such as dynamic simulations with time-dependent input and output variables. This could include non-rigid components or evolving relationships between assembly parts, enabling the modelling of effects like deformation under load, wear, or reconfiguration of mechanical constraints during operation. In the long term, the FMU approach may also allow internal processes and models to be manipulated externally for example, by selecting tolerance types, defining contact graphs, or detecting contact zones. These advanced capabilities, however, are highly dependent on the specific analysis method used and would need to be evaluated individually for each application.

5.3 Limitations

The approach demonstrated in this case study is subject to several limitations that currently restrict its broader applicability. Most notably, the FMU represents a static analysis of rigid-body behaviour and does not account for dynamic effects, deformation, or time-dependent interactions. As such, the FMU output reflects only a snapshot of the geometric relationships under worst-case or statistical assumptions. Without dynamic analyses or automated FMU exports, the approach therefore currently only involves additional implementation effort.

Moreover, the internal structure of the FMU is tightly coupled to the specific tolerance scheme used during its generation. Any modification to the tolerancing concept

DOI
10.3384/ecp218765

- such as changes in the type or arrangement of tolerance elements, beyond the mere numerical values - requires the FMU to be re-generated manually. This limits flexibility, particularly in early design phases where such schemes are subject to frequent change.

Based on the results presented in this paper, a fully automated export of FMUs is not yet feasible. Possible approaches to enable such automation could include systematically translating the methods from subsection 2.1 into DAE systems or representing their behavior using response surfaces as meta-models. A key prerequisite for enabling such automation would be the availability of freely accessible toolchains for geometric tolerance analysis. To our knowledge, the only freely available tool in this domain is *politopix* (Delos and Teissandier 2015). However, since politopix is based on Deviation Models, integrating it into an automated FMU generation workflow would be significantly more complex than with methods such as the Vector Loop approach. Nevertheless, its open accessibility could make it a valuable starting point for future developments in this direction.

6 Conclusion and Outlook

This work introduced a structured concept for integrating geometric tolerance analysis into system simulations using FMUs. A consistent interface based on the FMI standard was proposed, enabling the encapsulation of tolerance logic in a modular and reusable format. Through a case study, the practical feasibility of implementing a manually derived FMU was demonstrated and successfully verified against a commercial analysis tool.

While the current implementation is limited to static, rigid-body models and requires manual derivation of code, it shows that FMUs can represent mechanical tolerances with sufficient precision for system-level applications. With future development, including automation of model generation and support for dynamic or deformable systems, FMUs could serve as a practical bridge between detailed mechanical analyses and multi-domain system simulations.

In the long term, enabling FMU export directly from commercial CAT tools would be a desirable step to facilitate a broader application of geometric tolerance analysis within MBSE processes.

References

Anselmetti, Bernard (2006). "Generation of Functional Tolerancing Based on Positioning Features". In: *Computer-Aided Design* 38.8, pp. 902–919. DOI: 10.1016/j.cad.2006.05.005.

Blochwitz, Torsten et al. (2011). "The Functional Mockup Interface for Tool Independent Exchange of Simulation Models". In: *Proceedings of the 8th int. Modelica Conf.* Linköping University Press, pp. 105–114. DOI: 10.3384/ecp11063105.

Cao, Yanlong, Ting Liu, and Jiangxin Yang (2018). "A Comprehensive Review of Tolerance Analysis Models". In: *The Int. Journal of Advanced Manufacturing Technology* 97, pp. 3055–3085. DOI: 10.1007/s00170-018-1920-2.

Chase, Kenneth W. et al. (1996). "Including Geometric Feature Variations in Tolerance Analysis of Mechanical Assemblies". In: *IIE Transactions* 28.10, pp. 795–807. DOI: 10.1080/15458830.1996.11770732.

Chen, Hua et al. (2014). "A Comprehensive Study of Three Dimensional Tolerance Analysis Methods". In: *Computer-Aided Design* 53, pp. 1–13. DOI: 10.1016/j.cad.2014.02.014.

Davidson, Joseph K., Alma Mujezinovic, and Jami J. Shah (2002). "A New Mathematical Model for Geometric Tolerances As Applied to Round Faces". In: *Journal of Mechanical Design* 124, pp. 609–622. DOI: 10.1115/1.1497362.

Delos, Vincent and Denis Teissandier (2015). *PolitoCAT and Politopix*. http://i2m.u-bordeaux.fr/politopix. University of Bordeaux, Talence, France, accessed April 6, 2025.

Desrochers, Alain, Walid Ghie, and Luc Laperrie're (2003). "Application of a Unified Jacobian—Torsor Model for Tolerance Analysis". In: *Journal of Computing and Information Science in Engineering* 3.1, pp. 2–14. DOI: 10.1115/1.1573235.

Desrochers, Alain and Alain Riviere (1997). "A Matrix Approach to the Representation of Tolerance Zones and Clearances". In: *The Int. Journal of Advanced Manufacturing Technology* 13.9, pp. 630–636. DOI: 10.1007/BF01350821.

Gödecke, Andreas et al. (2012). "FEM models in System Simulations using Model Order Reduction and Functional Mockup Interface". In: *Proceedings of the 9th Int. Modelica Conf., Munich, Germany*. Linköping University Electronic Press, pp. 565–570. DOI: 10.3384/ecp12076565.

Gohl, Jesse et al. (2024). "Steady-state Optimization of Modelica Models and Functional Mockup Units with Pyomo". In: *Proceedings of the American Modelica ConferenceStorrs, Connecticut, USA*, pp. 109–118. DOI: 10.3384/ecp207109.

Gupta, S. and J. U. Turner (1993). "Variational Solid Modeling for Tolerance Analysis". In: *IEEE Computer Graphics and Applications* 13.3, pp. 64–74. DOI: 10.1109/38.210493.

Haghighi, Payam et al. (2014). "Automatic Detection and Extraction of Tolerance Stacks in Mechanical Assemblies". In: *Proceedings of the ASME Int. Design Engineering Technical Conf. and Computers and Information in Engineering Conf.* New York, NY: ASME. DOI: 10.1115/DETC2014-35315.

Hallmann, Martin, Benjamin Schleich, and Sandro Wartzack (2020). "From Tolerance Allocation to Tolerance-cost Optimization: A Comprehensive Literature Review". In: *The Int. Journal of Advanced Manufacturing Technology* 107, pp. 4859–4912. DOI: 10.1007/s00170-020-05254-5.

Homri, Lazhar, Denis Teissandier, and Alex Ballu (2015). "Tolerance Analysis by Polytopes: Taking into Account Degrees of Freedom with Cap Half-spaces". In: *Computer-Aided Design* 62, pp. 112–130. DOI: 10.1016/j.cad.2014.11.005.

Husch, Julia and Michael S. J. Walter (2024). "Tolerances in Mechanisms". In: *Research in Tolerancing*. Ed. by Sandro Wartzack. Cham: Springer Nature Switzerland, pp. 65–99. DOI: 10.1007/978-3-031-64225-8_4.

ISO International Organization for Standardization (2011). *ISO 8015:2011; Geometrical product specifications (GPS) — Fundamentals — Concepts, principles and rules*.

Kosec, P., S. Skec, and D. Miler (2020). "A Comparison of the Tolerance Analysis Methods in the Open-loop Assembly". In: *Advances in Production Engineering & Management* 15.1, pp. 44–56. DOI: 10.14743/apem2020.1.348.

Lehner, Daniel et al. (2025). “Model-driven Engineering for Digital Twins: A Systematic Mapping Study”. In: *Software and Systems Modeling*, pp. 1–39. DOI: 10.1007/s10270-025-01264-7.

Moers, Frederik et al. (2025). “Reusable Solution Element Libraries for Accelerated Application of MBSE in Mechanical Product Development”. In: *Forschung im Ingenieurwesen* 89.1, pp. 1–14. DOI: 10.1007/s10010-025-00825-y.

Morse, Edward et al. (2018). “Tolerancing: Managing Uncertainty from Conceptual Design to Final Product”. In: *CIRP Annals* 67.2, pp. 695–717. DOI: 10.1016/j.cirp.2018.05.009.

Otter, Martin et al. (2022). “Towards Modelica Models with Credibility Information”. In: *Electronics* 11.17. ISSN: 2079-9292. DOI: 10.3390/electronics11172728. URL: https://github.com/DLR-SR/Credibility.

Ramnath, Satchit et al. (2018). “Comparative Study of Tolerance Analysis Methods Applied to a Complex Assembly”. In: *Procedia CIRP* 75, pp. 208–213. DOI: 10.1016/j.procir.2018.04.073.

Schleich, Benjamin et al. (2016). “Status and Prospects of Skin Model Shapes for Geometric Variations Management”. In: *Procedia CIRP* 43, pp. 154–159. DOI: 10.1016/j.procir.2016.02.005.

Shah, Jami J. et al. (2007). “Navigating the Tolerance Analysis Maze”. In: *Computer-Aided Design and Applications* 4, pp. 705–718. DOI: 10.1080/16864360.2007.10738404.

Shen, Zhengshu et al. (2005). “A Comparative Study Of Tolerance Analysis Methods”. In: *Journal of Computing and Information Science in Engineering* 5.3, pp. 247–256. DOI: 10.1115/1.1979509.

Walter, Michael S. J. et al. (2021). “Statistical Tolerance Analysis—A Survey on Awareness, Use and Need in German Industry”. In: *Applied Sciences* 11.6, p. 2622. DOI: 10.3390/app11062622.

Wrede, Konstantin et al. (2024). “FMUGym: An Interface for Reinforcement Learning-based Control of Functional Mock-up Units under Uncertainties”. In: *31st Int. Workshop on Intelligent Computing in Engineering, EG-ICE 2024*, pp. 647–656. DOI: 10.35869/Proceedings_EGICE2024. URL: https://github.com/Fraunhofer-IIS/fmugym.

A Appendix

Listing 1. Modelica model with all equations before FMU-export

```
model geom_tol
// removed in- and outputs to save space
// real inputs: R1, S2, C4, S4, C5, R4, N2
// real outputs: FR1min, FR1_max, FR2_min,
    FR2_max
protected
parameter Real Char_L_S4 = 27.5;
parameter Real Char_L_R4 = 17.2679;
parameter Real l_R = 30.5;
parameter Real r_B = 1.0;
parameter Real a_C = 10.0;
parameter Real Pos_FR1_R_x = 13.96;
parameter Real Pos_FR1_R_y = 7.25;
parameter Real Pos_FR2_RN_x = 31.25;
parameter Real Pos_FR2_RN_y = 1.25;
parameter Real Pos_FR2_N_x = 13.21;
parameter Real Pos_FR2_N_y = 1.25;
parameter Real Pos_P_R_x = 31.5;
parameter Real Pos_P_R_y = 0.0;
parameter Real Pos_FR2ref_C_x = -0.25;
parameter Real Pos_FR2ref_C_y = 1.25;
parameter Real FR1_nom = 0.5;
parameter Real FR2_nom = 0.15;
Real alpha_C4;
Real S4_pos;
Boolean S4_negative;
Real beta_S4;
Real alpha_S4;
Real FR1_S4;
Real XM, YM, XB, YB;
Real FR2_S4;
Real alpha_R4;
Real FR2_R4;
equation
// C4 effect
alpha_C4 = asin(C4/2 / (l_R + r_B));
// S4 sign logic
S4_negative = S4 < 0;
S4_pos = if S4_negative then -S4 else S4;
beta_S4 = atan(S4_pos / Char_L_S4);
alpha_S4 = acos((a_C * cos(beta_S4) + l_R +
     r_B - a_C) / (l_R + r_B));
FR1_S4 = (Pos_FR1_R_y*(1 - cos(alpha_S4)) +
     Pos_FR1_R_x*sin(alpha_S4)) * (if
    S4_negative then -1 else 1);
XM = Pos_FR2_RN_x * cos(alpha_S4) -
    Pos_FR2_RN_y * sin(alpha_S4);
YM = Pos_FR2_RN_x * sin(alpha_S4) +
    Pos_FR2_RN_y * cos(alpha_S4);
XB = Pos_P_R_x * cos(alpha_S4) - Pos_P_R_y
    * sin(alpha_S4) + Pos_FR2ref_C_x * cos(
    beta_S4) - Pos_FR2ref_C_y * sin(beta_S4
    );
YB = Pos_P_R_x * sin(alpha_S4) + Pos_P_R_y
    * cos(alpha_S4) + Pos_FR2ref_C_x * sin(
    beta_S4) + Pos_FR2ref_C_y * cos(beta_S4
    );
FR2_S4 = sqrt((XB - XM)^2 + (YB - YM)^2) *
    (if S4_negative then -1 else 1);
// R4 effect
alpha_R4 = atan(R4 / Char_L_R4);
FR2_R4 = Pos_FR2_N_y*(1 - cos(alpha_R4)) +
    Pos_FR2_N_x*sin(alpha_R4);
// Final outputs
FR1_min = FR1_nom - R1/4 - S2/4
- (Pos_FR1_R_y*(1 - cos(alpha_C4)) +
    Pos_FR1_R_x*sin(alpha_C4))
- FR1_S4;
FR1_max = FR1_nom + R1/4 + S2/4
+ (Pos_FR1_R_y*(1 - cos(alpha_C4)) +
    Pos_FR1_R_x*sin(alpha_C4))
+ FR1_S4;
FR2_min = FR2_nom - C5/4 - N2/4
- (Pos_FR2_RN_y*(1 - cos(alpha_C4)) +
    Pos_FR2_RN_x*sin(alpha_C4))
- FR2_S4
- FR2_R4;
FR2_max = FR2_nom + C5/4 + N2/4
+ (Pos_FR2_RN_y*(1 - cos(alpha_C4)) +
    Pos_FR2_RN_x*sin(alpha_C4))
+ FR2_S4
+ FR2_R4;
end geom_tol;
```

Shared sea-environment definition and realisation for maritime and offshore co-simulations

Karl Gunnar Aarsæther[1,3] Stian Skjong[2]

[1]Department of Technology and Safety, UiT The Arctic University of Norway, karl.g.aarsather@uit.no
[2]SINTEF Nordvest AS, Norway, stian.skjong@sintef.no
[3]SINTEF Ocean AS, Norway, karl.gunnar.aarsather@sintef.no

Abstract

Marine operations are often developed with the aid of numerical simulation, in particular lifting operations and transfer of cargo between different units at sea. The effect of the environmental conditions is often the limiting factor and must be included together with models of different components and sub-systems. This paper describes an approach to synchronize spatial and temporal environment information such as universal constants, current, wind, and wave for use in co-simulations of marine operations. Co-simulation models in marine operations will inherently use physical constants, wind and current velocities to calculate forces. Wind and current velocities can have spatial and temporal variations that require the models to synchronize the values. In the event of simulations in waves, the position of the ocean surface, wave particle velocities must be coherent between individual co-simulation models. This paper suggests a structured description of an environment for co-simulation of marine operations. This is illustrated through the implementation of a co-simulation of an offshore lifting operation where a vessel model, a crane, wire and payload model, as well as positioning system models are all integrated with a common environment.

Keywords: FMI, Environment, Maritime, Marine operations, Co-simulation

1 Introduction

Co-simulation of marine operations is increasingly popular both in industry and research. The co-simulation methodology enables increased fidelity through domain specific models. The FMI (Modelica Association 2014) based framework Open Simulation Platform (OSP) (OSP 2025) has introduced additions for easier configuration of marine co-simulations with pre-defined connection types implementing power bonds and control system signals. It is common for simulation models to implement a physical system situated in an environment where common physical quantities are shared and interfaced to individual models. For co-simulation of marine operations it is important that all models have an identical perception of the environment. For operations in waves the harmonic representation of the ocean surface and wave loads is of particular importance as out-of synch models will result in chaotic results where control systems, cranes, payloads and vessels have time-shifted harmonic excitation. It is also important to ensure that all models use the same constants and fluid properties when calculating the forces. Defining a common specification for how a sea environment should be represented in a marine operations co-simulation model is one of the objectives of the SEACo project(SINTEF 2021). An environment model for co-simulation of marine operations must have a mechanism for synchronization of harmonic motions between models. Another issue is the possible variations in current and wind during the simulation. Wind and currents have natural spatial and temporal variations in addition to the influence of the vessel hull and superstructure on these flow fields.

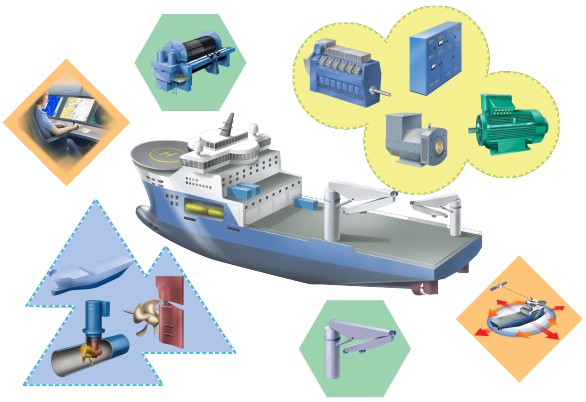


Figure 1. Systems and components important for performing safe and efficient maritime operations. Figure borrowed from the ViProMa project, (SINTEF 2025).

1.1 Marine lifting operations

Marine lifting operations are often complex and demanding, particularly when conducted from floating structures like ships. The precision of these operations is significantly influenced by environmental conditions such as wind and vessel motions. Additionally, when performing lifting operations through the splash zone and below, the lifted payload will encounter substantial environmental forces from currents, waves, and potential slamming forces upon entering the water.

To enhance operational precision, minimize environmental impact on the payload, and boost safety and efficiency, it is crucial that the entire vessel, along with all its essential systems, as depicted in Figure 1, functions seamlessly and in harmony. Therefore, the focus must shift from optimizing individual subsystems in isolation to prioritizing full-system optimization. This holistic approach is vital for improving overall performance, though it introduces significant complexity due to the intricate interdependencies between various subsystems, especially as the trend towards increased autonomy in demanding marine operations continues.

To illustrate, the *Dynamic Positioning* (DP) system must maintain the vessel's position and desired orientation with precision, while the crane operates, largely independent of the DP system, to control the payload's motion. Both the crane and DP systems draw power from the vessel's power system and are significantly affected by stochastic environmental conditions, which also impose requirements on onboard power production. This example merely scratches the surface of the complexity involved in such operations.

Minimizing the risk of operational failure, which can manifest as exceeding safety limitations or damaging the payload, is crucial. This risk can be mitigated through various methods, such as crew training, advanced onboard decision support systems (Skjong, Lars T Kyllingstad, et al. 2019; Lars Tandle Kyllingstad et al. 2023), and virtual prototyping of the operation itself (Skjong and Pedersen 2017a). The latter involves testing different operational strategies in simulators, as demonstrated in this work. Additionally, onboard decision support systems can be either data-based, physics-based, or a combination[1].

1.2 Sea environment

The important environmental conditions for marine operations are wind and current together with waves. Current is commonly represented in simulation models as vector fields with spatial variation. The temporal variation is often neglected since the time scales of changing ocean currents are usually larger than the duration of lifting operations. The temporal variations of wind cannot be neglected. Wind is described by a mean wind speed and direction, but with added gust variations in both speed and direction. Gusting describes flow phenomena that has a statistical description in the form of spectrums. Depending on the direction, the wind may be disturbed by the superstructure, creating a wake.

The surface waves that comprise a sea-state is a complex phenomena that is described by wave spectrums. Wave spectrums define the distribution of energy between different wave frequencies in a specific sea state. The distribution can be derived from oceanographic modelling, measured experimentally or from standardized wave spectra parametrized by simpler statistics such as significant wave height and peak period. For long crested seas all waves share the same direction whereas short crested seas make use of a directional spectrum to distribute wave energy among wave directions. In this paper we assume JONSWAP (K. Hasselmann et al. 1973) distributed wave energy and a simple $\cos^2$ directional distribution around a mean direction. A wave spectrum is a model of how energy is distributed between different wave frequencies in the frequency domain, while most simulation frameworks are developed for the time-domain. Wave spectrums can be converted to the time-domain by sampling the spectrum in the frequency domain and lumping the energy from the continuous spectrum into a finite set of harmonic functions. Figure 2 illustrate the process of converting a wave spectrum into a time-domain wave. To avoid an unreasonably large response at the origin, each component in the realisation must be phase shifted by a random value.

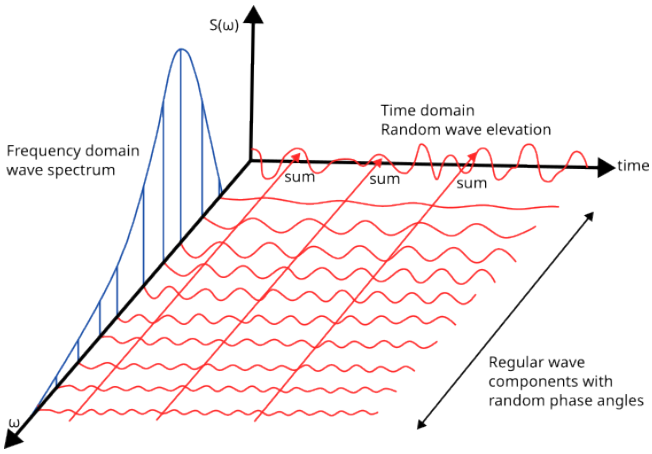


Figure 2. Relationship between frequency domain wave spectrum and time domain realisation by Fourier series. Adapted from (Faltinsen 1993)

Wind and waves are realised in the time domain in a fashion similar to Fourier series where frequencies and amplitudes are selected to replicate the energy distribution of the underlying power spectra. A shared environment model for marine lifting operations must represent:

- Ocean current magnitude and direction with possible spatial variation
- Wind speed and direction with gust variations, spatial variation and possible shielding effects
- Frequency and directional distribution of wave energy

In a co-simulation setting this information must be synchronized across all participating models. Particular attention has to be paid to the Fourier-series.

1.3 Previous work

Common parameters and calculations are easily implemented in monolithic simulation environments. Envi-

[1] A combination of data- and physics-based is preferable over pure data-based methods due to the reduced need for extensive training data required.

DOI
10.3384/ecp218775

ronments based on the Modelica language enables sharing of common resources with the inner/outer instances of generic classes. An example of a purpose built tool for marine operations is FhSim (Reite et al. 2014; Su et al. 2019) that implements numeric integration of ODE models, where the environment information and calculations are centralized in an environment model that is directly accessible by memory location as a global resource. A global singleton object is a common solution when all of the connected models are in a single process. Simulation models that are distributed and communicate across narrowly defined interfaces are nevertheless an attractive proposition in the marine industry where multiple models, and sometimes physical equipment, must be integrated in simulators for training, operation design and integration tests. Training simulators are a particular case where heterogenous software and equipment are combined to place a human operator in an immersive simulation environment operating in real-time. High-Level-Architecture (Dahmann, Fujimoto, and Weatherly 1997) has seen widespread adoption for integrating heterogenous models into a common simulator, particularly for training simulators. (McTaggart et al. 2021) shows the development of an HLA based simulator for ship-to-ship replenishment at sea. The simulator case is similar to offshore crane operations in the sense that it consists of a sea environment, a vessel, an external load, and interaction effects. The interaction effects are dissimilar in that instead of propulsors interacting with themselves, the environment and hull. The simulator in (McTaggart et al. 2021) consists of two ship federates, a replenishment equipment federate, a seaway federate and a hydrodynamic interaction federate to capture interaction effects between the two ship federates. HLA relies on common software libraries that communicate over an Run-Time-Infrastructure (RTI), and even if the HLA standard facilitates re-use by the definition of Federate Object Models, simulators must use a common RTI software to manage execution and exchange data. This is different from the FMI standard that describe co-simulation with a minimal interface between models where the same binary code can be used in different simulation environments.

There has been demonstrations of FMI based co-simulator frameworks for marine operations such as (Z. Liu et al. 2023), where a FMI based co-simulation software was used to simulate vessel motions and an attached crane. The environment in model in (Z. Liu et al. 2023) is described as affecting the vessel motions but not the crane. The crane lifting wire, slings, and payload is absent from the simulation and is therefore not connected to the environment definition. The environment model in (Z. Liu et al. 2023) describe the sea state as a collection of Fourier components.

2 The SEACo environment

A sea environment description for FMI-based co-simulation must represent quantities ranging from simple constants to power spectra. The purpose of placing the environment in a single model is to reduce errors by providing a common source of values. A single data source can provide wind, current, and waves from both conventional constant values, from pre-calculated datasets, and time-domain realisation of power spectrums on standard forms. The models in a co-simulation of marine operations will often calculate environmental forces by either assuming slender structures and applying the Morrison's equation (Eq. 1), and/or apply linear wave theory with superposition of forces generated by individual waves as described by Eq. 3.

Eq. 1 is applied to rigid bodies where the submerged volume V , drag coefficient C_d, added mass coefficient C_a are intrinsic to the shape and the fluid density ρ, the relative velocity v and relative acceleration $\dot{v}$ are properties of the environment.

$$F = \frac{1}{2}\rho A C_d v^2 + C_a \rho V \dot{v} \tag{1}$$

Fluid density, acceleration and velocity will have temporal and spatial variation in marine simulation models, i.e., when currents vary in magnitude and direction through the water column, or varying density from patches of brackish water. A shared environment model needs to provide both physical constants and fluid velocity information as a function of time and place.

Linear wave theory can represent a wave spectrum by a sum of sin-functions. models can calculate the surface elevation by evaluating Eq. 2

$$\zeta(x,y,t) = \sum_{i=0}^{N} \zeta_{a,i} \sin(\omega_i t - k_{x,i} x - k_{y,i} y - \varepsilon_i) \tag{2}$$

where $\zeta_{a,i}$,ω_i,$k_{x,i}$,$k_{y,i}$ and ε_i is the individual wave height, wave frequency, wave numbers in x and y direction and random phase angle respectively from the spectrum realisation. The choice of $\zeta_{a,i}$, $k_{x,i}$ and $k_{y,i}$ and ω_i for individual waves depends on the two spectra for the energy distribution between frequencies and the energy distribution in the wave direction. The method of converting the frequency domain wave spectrum into Fourier components and allocating the individual waves to a direction depend on the choice of number of Fourier components, random number generation and implementation of the spectra distribution functions.

A representation of the wave surface with superposition of pre-calculated amplitudes precludes the use of higher order methods to describe waves. However, the linear superposition of waves allows models to compute the instantaneous surface position. The dynamic pressure, wave particle velocity and accelerations can be derived from the individual wave velocity potentials as seen in Equation 3.

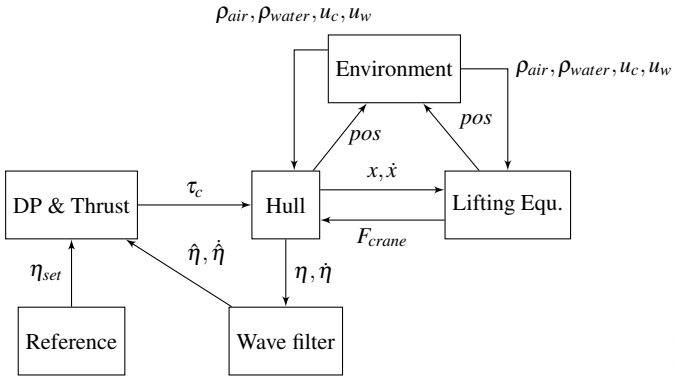


Figure 3. Co-simulation setup of vessel in a dynamic positioning operation with lifting equipment.

This makes it possible to model wave forces on slender structures such as crane wires and payloads in marine lifting operations.

$$\Phi_i(x,y,z,t) = \frac{\zeta_{a,i}g}{\omega_i} e^{-kz} \sin(\omega_i t - k_{x,i}x - k_{y,i}y - \varepsilon_i) \quad (3)$$

Eq. 3 contains the same variables as Eq. 2 and the combination of the equations can be used for calculating forces on slender structures and ventilation of propellers. Linear wave theory is also used in sea-keeping calculations that derive hydrodynamic forces as a function of wave frequency, direction and amplitude. See (Faltinsen 1993) for common applications of linear wave theory in offshore applications.

This paper propose dividing the specification sea-environment for co-simulation of marine operations into levels as seen in Table 1. Not all models need information about the waves, nor spatial and temporal variation of current and wind. The levels of specification makes it possible to classify a simulation model as supporting synchronized environments on different levels. Global current and wind can be discerned from spatial current and wind by checking model connections for a coupling of position information between the model and the environment with the origin as a default value for input port values.

Specification level 0 would make the constants required in Eqs. 1-3 available from a common source. The level of the environment description that is shared is increasingly sophisticated and the last level would include interaction between models through the environment for instance loss effects on propulsors from interaction effects between propeller jets, hull and surface.

3 Methods and Materials

3.1 Models and simulation setup

Figure 3 illustrates the co-simulation setup for the case study presented in this work. The subdivision into FMUs represent different disciplines that cooperate to establish a model of a marine operation. Control systems (DP-controller, wave-filter, thrust-allocation), Hydrodynamics (hull, propulsors), Mission specific equipment interconnected with the common Environment. Other subdivisions are possible, but the separation into FMUs reproduce real-world system boundaries where signals are sampled in control systems. ρ_{water}, represents the water density, and ρ_{air}, the air density. The vector u_c holds the water current speeds, which can be depth-dependent, and u_w holds the wind speed. τ_c is the thrust force vector in the vessel's body reference frame calculated by the DP control system. The yaw angle of the vessel is represented by ψ. The position and velocity vectors for vessel being fed to the lifting equipment model are x and $\dot{x}$, respectively. F_{crane} is the force feedback from the lifting equipment model. The desired position and orientation of the vessel are given by η_{set}, while $\hat{\eta}$ and $\dot{\hat{\eta}}$ represent the filtered position and orientation of the vessel, along with their corresponding rates. Finally, η and $\dot{\eta}$ denote the position and orientation of the vessel, and their corresponding rates. Note that information provided by the environment about the surface elevation (due to waves) are omitted in the figure. It is out of scope here to describe each model shown in the figure in detail, but a brief overview will be provided in the following.

3.1.1 Environment

A simplified environmental description was implemented that supported specification levels 1-3 from Table 1. The sharing of information is implemented with the port system and the environment output include acceleration of gravity (g) and the densities for air and water (ρ_{air} and ρ_{water}, respectively). The output of wind and current magnitude and direction is implemented with a vector input for positions with a corresponding output with velocity magnitude of wind and current. The first three values in the position input correspond to the directional components of the wind and current in the first three output values. Here, the wind is assumed constant while the current profile follows an exponential decay with increasing depth:

$$v(z)_{curr} = v(z=0)_{curr} e^{-az} \quad (4)$$

The wind, current and decay rate values are set as parameters. The waves are generated from a JONSWAP spectrum for a given significant wave height (Hs) and peak period (Tp) pair, and the mean wave direction. Here, the number of Fourier components are set to 24 and the spectrum is realised by random sampling of the wave frequency in the interval $(0, 2\pi]$. The wave direction is randomly selected from a cos^2 distribution centred on the mean wave direction. The wave directions are assigned to the individual waves by descending wave amplitude and distance from the mean wave direction. Wave phase angles was randomly selected. Each wave require four values, as shown in Eq. 2, to be specified resulting in 4 vector ports or 96 individual ports to be connected to the other models. The values on the output ports of the simplified environment model are constants. But usage of the port

DOI 10.3384/ecp218775

Table 1. Specification levels for synchronized environment.

Level of Specification	*Description*	*Example*
0	Basic constants	Identical constants are used from a common source. Can be achieved with constant value signals.
1	Global current and wind	Global wind and current values as input coupled to a common source. Can be achieved with constant value signals.
2	Spatial current and wind	Wind and current based on individual positions. To model spatial and temporal variations in current and wind around the vessel and in the water column. Requires input of position where wind and current is observed, eg. payload position as input results in output of current and wind velocities at payload position. Centralized implementation of wind gust spectrum.
3	Synchronized wave definition	Centralized implementation of wave energy and direction spectrum formulas and realisation of individual waves. Output list of wave components allows connected models to calculate surface elevation, wave particle motions and utilize sea-keeping results from linear wave theory.
4	Propulsor interaction	Propulsor losses can be introduced by phenomena such as the Coanda effect, inflow disturbance and propeller jet interaction. There are no general methods for including such effects in a flexible reconfigurable environment such as co-simulations.

system ensures that changes in environment values will be reflected in the interconnected models.

3.1.2 Hull

The hull geometry is analysed using *HAMS*, an open-source computer programme for the analysis of wave diffraction and radiation of three-dimensional floating or submerged structures (Y. Liu 2019). The hydrodynamic results from the analysis are transformed into a state-space time-domain formulation as presented in (Thor I. Fossen 2005). The equation of motion for the hull is given in Eq. 5.

$$\dot{\eta} = \mathbf{J}(\Theta)\nu \tag{5a}$$

$$\mathbf{M}\dot{\nu} + \mathbf{C}_{RB}\nu + \mathbf{D}\nu + \mu + \mathbf{g}(\eta) = \tau_{env} + \tau \tag{5b}$$

In (5) the vector η represents the generalized position and orientation of the vessel given in the NED-, *North, East, Down*, reference frame. The matrix $\mathbf{J}(\Theta)$ is the transformation matrix dependent on the Euler angles Θ, which describe the vessel's orientation. The vector ν represents the generalized velocity of the vessel, including both linear and angular velocities. The matrix $\mathbf{M}$ is the generalized mass matrix, which includes both the rigid-body mass and the added mass due to hydrodynamic effects. The matrix $\mathbf{C}_{RB}$ represents the Coriolis and centripetal forces due to the rigid-body motion of the vessel. The matrix $\mathbf{D}$ is the linear damping matrix, which accounts for the hydrodynamic damping forces acting on the vessel. The vector μ represents the memory effect of the fluid, which is modelled as an integral term involving the retardation function. Note that μ can be approximated by a linear reduced-order state-space model, as described in (Perez and Thor I. Fossen 2011). The vector $\mathbf{g}(\eta)$ represents the restoring forces and moments due to gravity and buoyancy, which depend on the position and orientation of the vessel. The vector τ_{env} represents the environmental forces acting on the vessel, including wave, wind, and current forces. The vector τ represents the control input forces and moments applied to the vessel.

The hull model outputs position, orientation, and corresponding rates for its centre of gravity, as well as for fixed positions on the hull. It takes forces as input, both described in the NED reference frame and in the body-fixed reference frame. The hull also gets environmental information from the environmental model, such as information about the water density (ρ_{water}), the wave components (direction, amplitude, frequency, phase, etc.), and the wind conditions (speed and direction).

3.1.3 Lifting Equipment

This lifting equipment model consists of a crane-tip, a wire model, four slings and a rectangular payload, and the model is developed based on the framework presented in (Skjong, Reite, and Aarsæther 2021), which focuses on lumped, constrained cable modelling using an explicit state-space formulation. This model simulates the dynamics of a payload being lifted by a crane-wire, accounting for various forces acting on the system, including wind forces, buoyancy, first-order wave forces, and drag forces. The drag force is calculated with Eq. 1 and the first-order wave force is calculated by assuming a fully submerged payload and using Eq. 3 to calculate the wave force from

each wave component as

$$F_j = \sum_{i=0}^{N} \rho_{water} V \frac{\delta^2 \Phi_i}{\delta j \delta t} \quad \text{for j=x,y,z} \tag{6}$$

Where V is the payload volume and Φ is given in Eq. 3.

The model employs a lumped parameter approach, where structural elements are represented by discrete mass nodes connected by constraints. These constraints are solved explicitly using an elastic version of Baumgarte stabilization (Baumgarte 1972), ensuring numerical stability and avoiding singularities in matrix inversions. The Baumgarte stabilization method reformulates constraint forces to handle differential-algebraic equations, allowing efficient numerical integration. The model receives the vessel position, orientation and corresponding rates and offset from the center of gravity as inputs and calculates a force vector that is applied to the vessel in the centre of gravity through a port coupling.

3.1.4 DP, thrust and references

The DP controller is here assumed to be a simple PID-controller that controls the global thrust force vector needed to keep the vessel in position, as presented in (Skjong and Pedersen 2017a). The equations in the DP-controller are given as

$$\tau_c = (\mathbf{R}_z(\psi))^\top \left[\mathbf{K}_P \mathbf{e}_P + \mathbf{K}_D \dot{\mathbf{e}}_P + \mathbf{K}_I \int_0^t \mathbf{e}_P \, dt \right] \tag{7}$$

In the equation the control force vector is denoted by τ_c. The rotation matrix $\mathbf{R}_z(\psi)$ is a function of the yaw angle ψ, which describes the vessel's orientation about the vertical axis. The transpose of this rotation matrix is $(\mathbf{R}_z(\psi))^\top$. The proportional gain matrix is $\mathbf{K}_P$, and the error in the position vector is $\mathbf{e}_P$. The error is defined as the difference between the desired position and orientation of the vessel, and the measured (filtered measurements) position and orientation of the vessel. Note that the desired position and orientation of the vessel could be fixed scalars or slowly varying variables, and are modelled in the *Reference* model shown in Figure 3. The derivative of the error in the position vector is $\dot{\mathbf{e}}_P$, and the derivative gain matrix is $\mathbf{K}_D$. The integral of the error in the position vector over time is $\int_0^t \mathbf{e}_P dt$, and the integral gain matrix is $\mathbf{K}_I$.

In this co-simulation study, propulsors and thrust allocation are omitted. The global thrust command vector from the DP controller is acting directly as the thrust force on the hull. This can be argued for since the main focus is the lifting equipment and the payload being lowered into the sea with a common envionment. For a well designed thrust allocation algorithm and well-tuned control system, the dynamics of the complete propulsion system should be comparable to the output of the DP-controller.

3.1.5 Wave filter

The wave filter is used to filter out first order motion oscillations for the vessel, which in general, is not needed to compensate for, nor possible in realistic operations without spending an enormous amount of energy. Hence, the wave filter provides the dynamic positioning system with the filtered vessel position, orientation, and corresponding rates. The filter is of a passive non-linear observer design, as elaborated in (Thor I Fossen and Strand 1999), and its dynamics are described by the equations listed in (8).

$$\dot{\xi} = \Omega \xi + \mathbf{K}_1 \bar{\mathbf{y}} \tag{8a}$$

$$\dot{\eta} = \mathbf{J}(\mathbf{y}) \hat{\mathbf{v}} + \mathbf{K}_2 \bar{\mathbf{y}} \tag{8b}$$

$$\dot{\mathbf{b}} = -\mathbf{T}^{-1} \mathbf{b} + \frac{1}{\gamma} \Lambda \bar{\mathbf{y}} \tag{8c}$$

$$\mathbf{M} \hat{\mathbf{v}} = -\mathbf{D} \hat{\mathbf{v}} + \mathbf{J}^T(\mathbf{y}) \dot{\mathbf{b}} + \tau + \frac{1}{\gamma} \mathbf{J}^T(\mathbf{y}) \mathbf{K}_3 \bar{\mathbf{y}} \tag{8d}$$

$$\hat{\mathbf{y}} = \hat{\eta} + \Gamma \xi \tag{8e}$$

In the filter, the state vector ξ represents the wave-induced motion. The matrix Ω describes the dynamics of this motion, while $\mathbf{K}_1$ is the observer gain matrix for the wave-induced motion. The estimation error vector is denoted by $\bar{\mathbf{y}}$. The position and orientation vector of the vessel is η, and $\mathbf{J}(\mathbf{y})$ is the transformation matrix dependent on the vessel's position and orientation. The estimated velocity vector of the vessel is $\hat{\mathbf{v}}$, with $\mathbf{K}_2$ being the observer gain matrix for the position and orientation. The bias vector $\mathbf{b}$ accounts for slowly varying environmental disturbances. $\mathbf{T}^{-1}$ is the inverse of the diagonal matrix of bias time constants, and γ is a scalar tuning parameter. The matrix Λ scales the amplitude of the estimation error. The mass matrix of the vessel is $\mathbf{M}$, and $\mathbf{D}$ is the damping matrix. The transpose of the transformation matrix $\mathbf{J}(\mathbf{y})$ is $\mathbf{J}^T(\mathbf{y})$. The control input vector is τ, and $\mathbf{K}_3$ is the observer gain matrix for the velocity estimation. Finally, $\hat{\mathbf{y}}$ is the estimated measurement vector, and Γ is the matrix representing the relationship between the wave-induced motion and the measurements.

This filter gets the vessel position and orientation as inputs, and is tuned with knowledge about the wave conditions and the vessel dynamics. The output of the filter is the filtered vessel position and orientation, along with corresponding rates. Figure 4 shows illustrates the performance of the wave filter, where the vessel's first order motions in the north position has been filtered.

3.2 Co-simulation setup

The Open Simulation Platform programme `cosim` (OSP 2025) was used to run simulations. cosim includes a `VariableGroup` mechanism to ease interconnection of models with a large number of ports, and the wave component parameters were combined into `VariableGroups` for ease of use. Main parameters used in the co-simulation is shown in Table 2

Simulations were run on a Intel I9-12900 with 64GB RAM and the simulation consumed 1006[s] CPU time for 1200 [s] simulation time (790[s] wall-clock time due to parallel execution).

DOI 10.3384/ecp218775

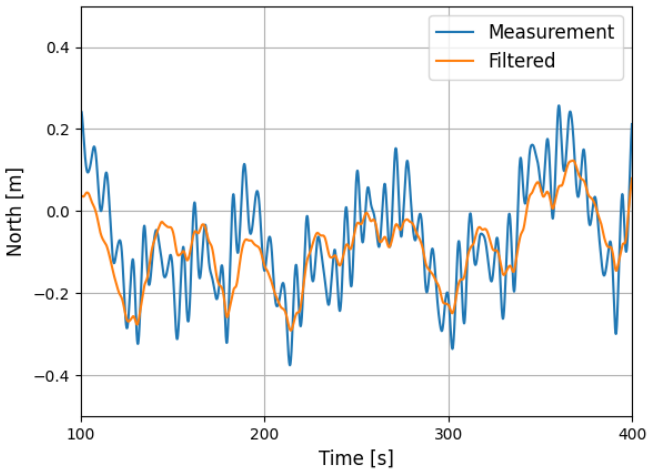


Figure 4. Wave filter performance example where the vessel's first order motions in the north position has been filtered.

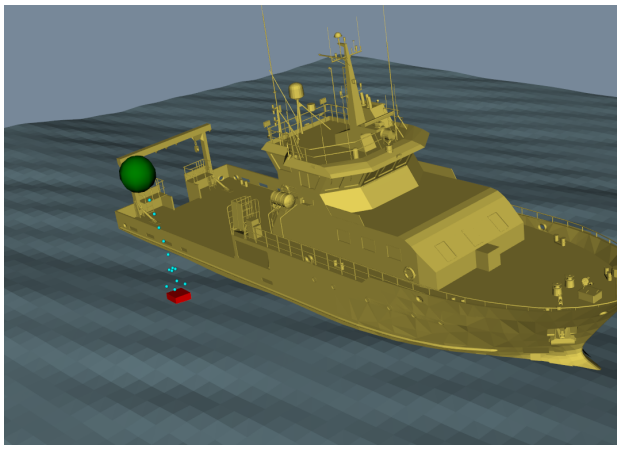


Figure 5. VTK visualization of the position traces in Figure 6. The image shows values derived from the environment model, ship model, crane point with wire and load. Red colour on load signifies water contact

Table 2. Main parameters in co-simulation.

Parameter	*Value*
Ship size	$36.25[m]$ x $9.6[m]$ x $2.7[m]$
Ship mass	574 $[tons]$
Crane height	8 $[m]$
Crane boom length	5 $[m]$
Crane base offset	5.5 $[m]$, -4.5 $[m]$, 1.15 $[m]$
Payload size	$1[m]$ x $1[m]$ x $1[m]$
Payload mass	5 $[tons]$
Wire diameter	0.03 $[m]$
Winch speed	0.05 $[m/s]$

4 Results

The system model shown in Figure 3 was defined in `cosim` with separate FMUs for each box in the figure. The communication stepsize was $0.01[s]$ and results from the individual FMUs were saved in separate csv files. A VTK based visualization was developed that animated the visual components from the different FMUs in 3D based on the contents of the individual csv files. The results of a hull with attached crane & payload in a irregular wave field is shown in Figure 5. The parameters of the individual waves in Figure 5 is transmitted to both vessel an crane & payload model. The resulting position trace for the hull, crane tip and payload from a $120[s]$ simulation with the payload in the wave zone for $Hs = 1[m]$ and $Tp = 7[s]$ is shown in Figure 6. The position trace shown the harmonic oscillations of the vessel in linear and angular degrees-of-freedom and amplified by the lever arm at the crane-tip, the figure also shows the resulting motions of the payload at the end of the crane wire. The payload will experience wave forces in the wave zone together with buoyancy and The tension in the crane wire and the magnitude of the buoyancy and first-order wave force on the payload are shown in Figure 7. The buoyancy and wave forces are only activated when the payload is in contact with the ocean surface as shown in the bottom panel of Figure 7. Note that this model does not include slamming loads that is a limiting phenomenon in offshore lifting operations.

Two $1200[s]$ cases were simulated to allow the payload to transition through the wave zone and through the water column. The simulation time also test the wave filter and DP-controller ability to keep the position in response to the wave, current and wind loads on the hull. Two moderate and severe JONSWAP sea states with $Hs = 1[m]$ & $Tp = 7[s]$ and $Hs = 2.5[m]$ & $Tp = 10[s]$ were simulated with $5[m/s]$ wind and $0.5[m/s]$ current. Crane wire tension, buoyancy force and magnitude of the wave force are shown in Figures 8a and Figure 8b, respectively. From the bottom panel of the figures it is evident that the longer waves of the $Hs = 2.5[m]$ sea state influence the payload for a longer time than in the $Hs = 1.0[m]$ case. The time spent in the wave zone is longer as shown in 'noisy' buoyancy force and resulting hull motions are evident as larger variability in the crane wire tension.

Figure 9a and Figure 9b show the magnitude of the wave and drag forces as a function of payload depth. Wave forces decrease with depth as suggested by Eq. 3. The drag forces in Eq. 1 depend on the relative velocity and will be affected by both crane tip motions propagating down to the payload and the current. It should be noted that the wave forces are exiting forces while drag forces act against the relative motion of the payload.

5 Discussion and Conclusion

This paper has demonstrated a practical example of how a synchronized sea-environment model can be included in a co-simulation. The sea-environment model is accompanied by a hierarchical specification of synchronization levels to clearly identify which parts of the environment description are shared between models. The choice of converting the frequency-domain wave spectrum to time-domain before transmitting individual wave components

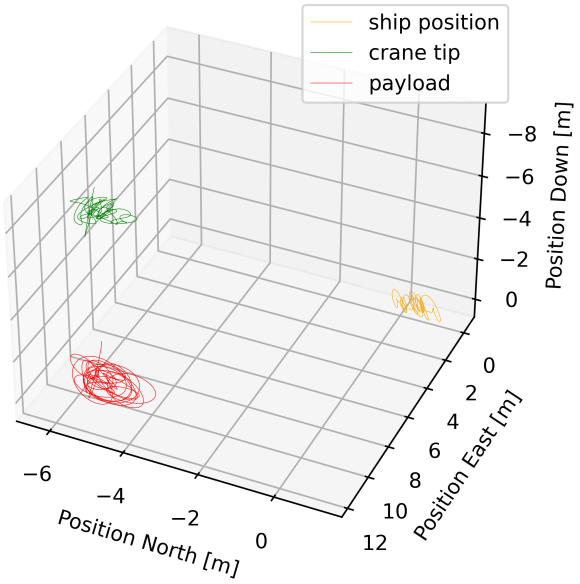


Figure 6. Position of the vessel centre of gravity, crane tip and payload. Cyan dots show the element centres for main wire and slings

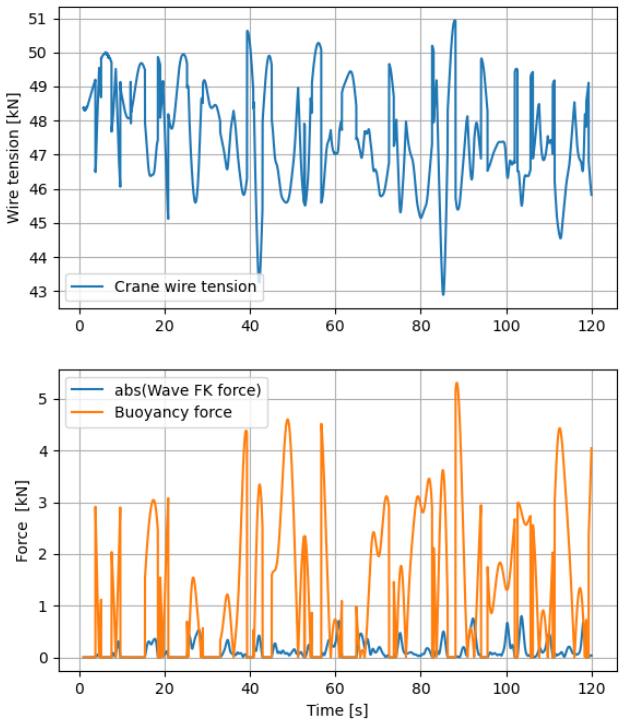
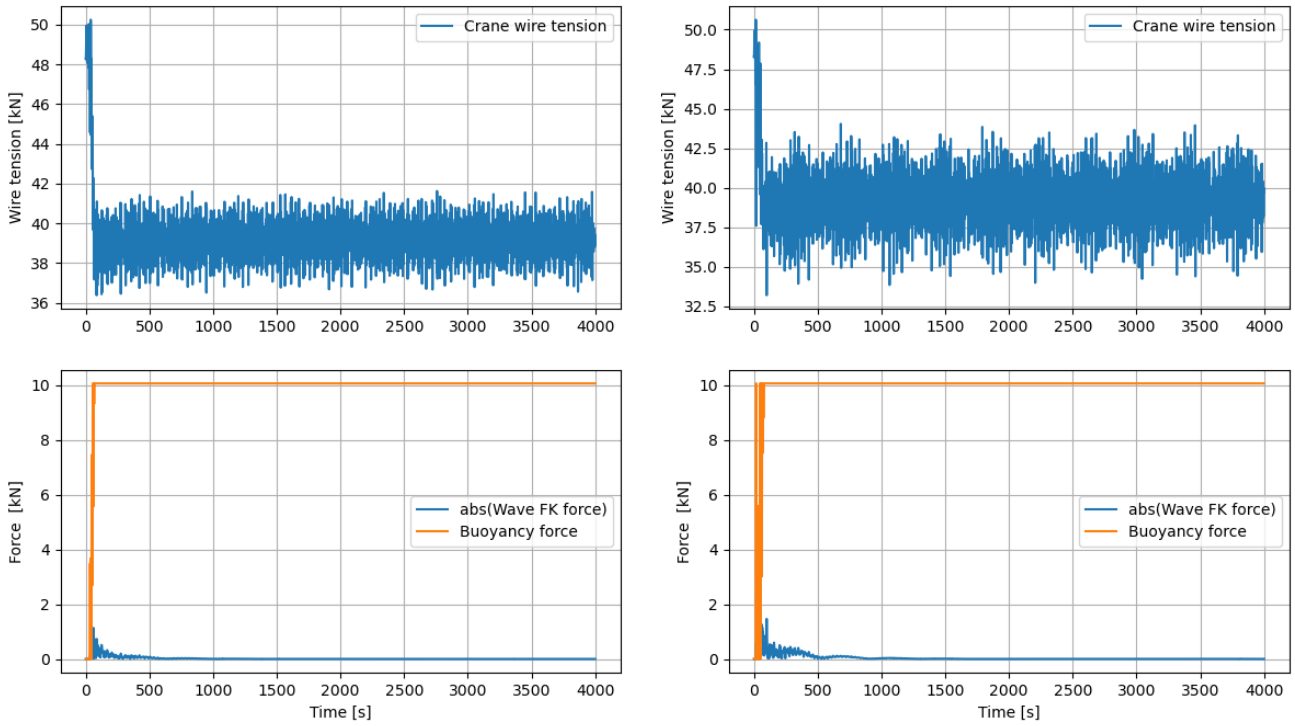


Figure 7. Crane and payload forces over 120 [s] as the payload hangs in the wave zone.

is an engineering approach that limits the model to linear wave theory. Linear wave theory is widely applied in ocean engineering, and this solution is compatible with common methods for evaluating sea-keeping and allows the models to apply Eq. 3 to derive wave properties, but limit the simulation to a description of the wave zone that relies on superposition to describe waves and current. The proposed interface in this paper does not preclude advanced methods for describing current and wind, precomputed velocity fields can be read by the environment model and used to output magnitude and direction at any point. There is neither a necessity that the environment model is a 'pure' model, as a model of a vessel can implement the environment interface and connect to models of propulsors and lifting equipment that interface with an environment a model description that is tightly coupled to the vessel geometry. However, only one model implementing the environment interface would be permissible in a co-simulation. The proposed interface place a large burden on the simulation developer since a great number of ports must be connected. The interface is not well suited for basic FMI v1 and FMI v2 tools due to the large number of interconnections required, it is technically possible but the workload on the simulation developer would be excessive and negate many of the advantages of GUI tools. Software with special features such as VariableGroups in 'cosim' allows easy coupling of models to the environment, and FMI v3 with vector ports would negate the need for extensions in the co-simulation software. Usage of the port system for exchange waves components is not necessarily required as a shared library for computing wave spectrum values, randomly sampling them and computing phase angles can be developed. Control of the random number generator is necessary in such approaches to ensure that random phase angles, frequency and direction random samples are identical between models. It would be difficult to verify that all models actually used the shared library, and that each instance of the library is of the same version as time progresses, and from the authors perspective this would allow for version drift between models and limit the selection of simulation environment where models interfacing with the environment could be developed. An attractive property of transmitting wave components on ports is when components are sorted by amplitude a reduced number of components would mean a reduced fidelity wave field prioritizing the the most energetic waves. Propulsor interaction in co-simulation is a difficult problem as the interaction and loss effects are interdependent on wave conditions (distance from surface), geometries (hull surfaces) and other propulsors (jet and flow effects). One method would be to extend the linear superposition of wave particle velocities to propeller jets superimpose propeller jet velocities on a volume downstream of the propulsor. This would allow propulsor jets to influence other objects and change the inflow conditions of propulsors in the jet, but would not be sufficient to capture the interaction effects between jets influencing the flow patterns and in-

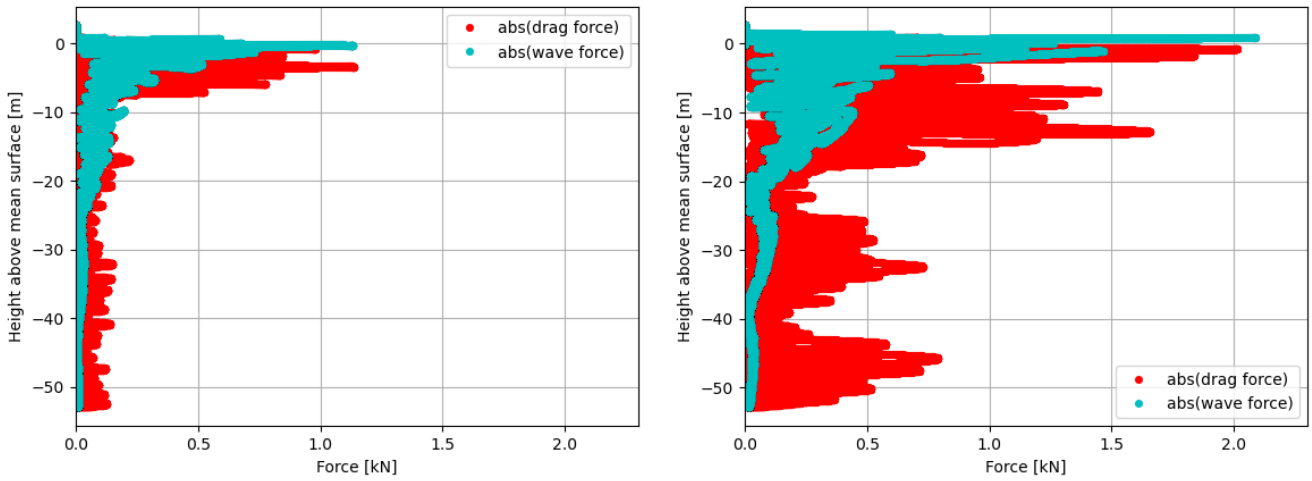
DOI
10.3384/ecp218775



(a) Hs=1$[m]$ Tp=7$[s]$.

(b) $H_s = 2.5[m]$ and $T_p = 10[s]$.

Figure 8. Crane and payload forces over 1200 [s] as the payload is lowered through the wave zone and down into the ocean.



(a) Hs=1$[m]$ Tp=7$[s]$.

(b) $H_s = 2.5[m]$ and $T_p = 10[s]$.

Figure 9. Payload wave and drag forces as a function of water depth over 1200 [s] as the payload is lowered through the wave zone and down into the ocean.

teraction between propulsor jets and hull surfaces. The co-simulation model in this paper is fairly complete, but simplified. In general, the global thrust vector command from the DP controller must be transformed to local thrust forces and distributed to all the available, and operating, propulsors. This is done using a thrust allocation algorithm (which can be a simple thrust allocation matrix in some special cases). Such a setup is shown in (Skjong and Pedersen 2017b) where the thrust allocation is formulated as a non-angular *Model Predictive Control* (MPC) problem to be solved. The payload model include first order wave forces, but not the larger slamming forces experienced during wave impacts in the wave zone. Slamming pressure is larger than the regular dynamic pressure in the wave and must be accounted for in decision support applications. The proposed interface enables software such as

'cosim' to detect and verify the environment specification level for the co simulation. Common physical parameters, wind and current input with or without position feedback would be interpreted as specification level 0, 1 and 2 receptively while input of wave components would place the co-simulation at level 3. This also enables a partition of the models in the co-simulation into models participating in a sea environment, while models representing for instance control systems are identified as not directly influenced. The work on developing a sea-environment specification has highlighted the need for a reference frame specification. Different disciplines and application areas have different traditions for axis systems, and this makes verification of the wind, current and wave forces important as the simulation developer in principle can be without access to the source code of the models in the co-simulation.

Acknowledgements

This work has been supported by The Norwegian Research Council project Safer, Easier and more Accurate Co-Simulations (SEACo) grant number 326710. The authors would also like to thank the Open Simulation Platform consortium for use of the OSP *cosim* software.

References

Baumgarte, J. (1972-06-01). "Stabilization of Constraints and Integrals of Motion in Dynamical Systems". In: *Computer Methods in Applied Mechanics and Engineering* 1.1, pp. 1–16. ISSN: 00457825. DOI: 10.1016/0045-7825(72)90018-7.

Dahmann, Judith S, Richard M Fujimoto, and Richard M Weatherly (1997). "The department of defense high level architecture". In: *Proceedings of the 29th conference on Winter simulation*, pp. 142–149.

Faltinsen, Odd (1993). *Sea loads on ships and offshore structures*. Vol. 1. Cambridge university press.

Fossen, Thor I and Jann Peter Strand (1999-01). "Passive Nonlinear Observer Design for Ships Using Lyapunov Methods: Full-Scale Experiments with a Supply Vessel". In: *Automatica* 35.1, pp. 3–16. ISSN: 00051098. DOI: 10.1016/S0005-1098(98)00121-6.

Fossen, Thor I. (2005-09). "A Nonlinear Unified State-Space Model for Ship Maneuvering and Control in a Seaway". In: *International Journal of Bifurcation and Chaos* 15.09, pp. 2717–2746. ISSN: 0218-1274, 1793-6551. DOI: 10.1142/S0218127405013691.

Hasselmann, Klaus et al. (1973). "Measurements of wind-wave growth and swell decay during the Joint North Sea Wave Project (JONSWAP)." In: *Ergaenzungsheft zur Deutschen Hydrographischen Zeitschrift, Reihe A*.

Kyllingstad, Lars Tandle et al. (2023). "AMADEA-Adaptable Maritime Decision Support Architecture". In.

Liu, Yingyi (2019-03-26). "HAMS: A Frequency-Domain Preprocessor for Wave-Structure Interactions—Theory, Development, and Application". In: *Journal of Marine Science and Engineering* 7.3, p. 81. ISSN: 2077-1312. DOI: 10.3390/jmse7030081.

Liu, Zizheng et al. (2023). "A Co-simulation-Based System Using Vico for Marine Operation". In: *Software Engineering and Formal Methods. SEFM 2022 Collocated Workshops*. Ed. by Paolo Masci et al. Vol. 13765. Cham: Springer International Publishing, pp. 228–241. ISBN: 978-3-031-26235-7 978-3-031-26236-4.

McTaggart, K et al. (2021-12). "International Development and Validation of a Distributed Simulation for Naval Ship Replenishment at Sea". In: *International Journal of Maritime Engineering* 161.A1. ISSN: 1479-8751, 1479-8751.

Modelica Association (2014-07). *Functional Mock-up Interface for Model Exchange and Co-Simulation Version 2.0*. Tech. rep. Linköping: Modelica Association. URL: https://fmi-standard.org.

OSP (2025). *Open Simulation Platform*. URL: https://opensimulationplatform.com/ (visited on 2025-03-27).

Perez, Tristan and Thor I. Fossen (2011-02). "Practical Aspects of Frequency-Domain Identification of Dynamic Models of Marine Structures from Hydrodynamic Data". In: *Ocean Engineering* 38.2–3, pp. 426–435. ISSN: 00298018. DOI: 10.1016/j.oceaneng.2010.11.004.

Reite, Karl-Johan et al. (2014). "Fhsim—time domain simulation of marine systems". In: *International Conference on Offshore Mechanics and Arctic Engineering*. Vol. 45509. American Society of Mechanical Engineers, V08AT06A014.

SINTEF (2021). *Safer, Easier and more Accurate Co-simulations (SEACo)*. URL: https://www.sintef.no/en/projects/2021/seaco/ (visited on 2025-05-08).

SINTEF (2025). *Virtual Prototyping of Maritime Systems and Operations (ViProMa)*. URL: https://www.sintef.no/prosjekter/2013/viproma/ (visited on 2025-04-23).

Skjong, Stian, Lars T Kyllingstad, et al. (2019). "Generic onboard decision support system framework for marine operations". In: *International Conference on Offshore Mechanics and Arctic Engineering*. Vol. 58844. American Society of Mechanical Engineers, V07AT06A032.

Skjong, Stian and Eilif Pedersen (2017a). "Co-simulation of a marine offshore vessel in DP-operations including hardware-in-the-loop (HIL)". In: *International Conference on Offshore Mechanics and Arctic Engineering*. Vol. 57731. American Society of Mechanical Engineers, V07AT06A038.

Skjong, Stian and Eilif Pedersen (2017b). "Non-Angular MPC-based Thrust Allocation Algorithm for Marine Vessels - A Study of Optimal Thruster Commands". In: *Submitted to IEEE Transactions on Transportation Electrification*.

Skjong, Stian, Karl-Johan Reite, and Karl Gunnar Aarsæther (2021-03). "Lumped, Constrained Cable Modeling with Explicit State-Space Formulation Using an Elastic Version of Baumgarte Stabilization". In: *Journal of Offshore Mechanics and Arctic Engineering* 143.December, pp. 1–15. ISSN: 0892-7219. DOI: 10.1115/1.4050422.

Su, Biao et al. (2019). "A multipurpose framework for modelling and simulation of marine aquaculture systems". In: *International conference on offshore mechanics and arctic engineering*. Vol. 58837. American Society of Mechanical Engineers, V006T05A002.

Decreasing Risk in the Design of Large Coupled Systems via Co-Simulation-Based Optimization and Adaptive Stress Testing

Severin Sadjina[1] Lars T. Kyllingstad[2] Stian Skjong[1]

[1]SINTEF Nordvest, Ålesund, Norway {severin.sadjina,stian.skjong}@sintef.no
[2]SINTEF Ocean, Trondheim, Norway, lars.kyllingstad@sintef.no

Abstract

Optimization and stress testing are key aspects of the design and verification process for large, high-risk systems. Optimization is about improving the capabilities and performance of a system; stress testing is about uncovering its weaknesses and faults. Both require a quantitative representation of the system's behavior, and for complex, multi-physical systems, co-simulation can be a very powerful method to create such a representation. However, co-simulation frequently involves the use of black-box subsystem models, which poses challenges to traditional optimization and stress testing methods. Here, we review the state of the art in co-simulation-based optimization and stress testing, focusing especially on *adaptive stress testing* in the latter case, and discuss open research questions and promising research directions. In particular, we make the case that a co-simulation is not an entirely black box even when some or all of its subsystems are; it may be possible to exploit the visible system structure, coupling variable values, and partial subsystem information. We use examples from the maritime industry to motivate and illustrate the discussion, centering on the highly contemporary design case of an autonomous ferry.

Keywords: co-simulation, optimization, stress testing

1 Background

Like most other industries, the maritime sector finds itself in the middle of a "twin transition": a green transition to a low-carbon future, intertwined with a digital transition towards data-driven decision making and autonomous systems. And like any other, the maritime industry has its own, particular set of challenges in that regard.

The main obstacle to decarbonization lies in finding alternatives to the fossil fuel oils that power virtually all of today's ships. At this point in time, there are few readily-available zero- or low-carbon options for ships that spend many days or weeks away from shore, such as long-distance transport ships and large fishing vessels. In the meantime, fuel-saving and transitional solutions like hybrid and dual-fuel power systems significantly increase the complexity of a ship in terms of design, construction, operation, and maintenance.

When it comes to maritime digitization, the frontier is moving towards remotely operated, semi-autonomous, and eventually autonomous shipping. While automating the navigation of a ship through well-regulated seaways is arguably an easier problem than autopiloting a car through chaotic city streets, the consequences of any one accident are likely to be orders of magnitude larger – possibly in terms of human life, depending on the ship type, but also in terms of environmental and financial impacts. Reducing risk and increasing safety are therefore prime concerns.

Hence, the maritime industry is, now more than ever, in need of new methods for design, verification, and assurance of highly complex, energy efficient, and *safe* systems. Such methods must be able to account for the fact that the design, construction, and commissioning of a large ship is a significant undertaking which involves a great number of stakeholders, a wide array of engineering disciplines, and tight regulatory and budgetary constraints.

This is the context within which we have launched the research project *OptiStress: System optimization and stress testing in co-simulations*. Our basic premise is that co-simulation,[1] as a modeling and simulation paradigm, fits the bill in terms of its suitability for multilateral virtual prototyping of multi-physical and cyber-physical systems. This is in large part because it enables one to build simulations of complex systems from loosely coupled black-box subsystems. However, this becomes a *disadvantage* when it comes to design optimization and whole-system assurance, because some of the assumptions underlying existing numerical methods no longer hold.

In *OptiStress*, and in this paper, we will look at two key design and verification processes and explore how co-simulation may be embedded within them in a way that leverages its strengths and mitigates its drawbacks. The first is *system optimization*, in particular the use of numerical optimization algorithms to choose the system configuration and parameters that best fulfill the system design goals. The second is *stress testing*, which is concerned with pushing a system to the limits of its safe operating space to see what breaks, how, and why. Here, we'll look specifically at *adaptive stress testing* (AST), where reinforcement learning is applied to find the most likely paths to a failure event (Lee, Mengshoel, et al. 2020).

[1]The term "co-simulation" is sometimes used loosely in the academic literature to mean different things. Here, we refer specifically to *continuous-time* co-simulation, as defined by Kübler and Schiehlen (2000) and Gomes et al. (2019).

The goal of this paper is to provide a concise summary of prior work and outline the main research needs. To this end, separate discussions on optimization and AST are given in the context of co-simulation in sections 2 and 3, respectively. Section 4 concludes by commenting on how both can be combined in the same design workflow or even the same algorithmic loop.

We will use an autonomous battery-electric car ferry as an illustrative example throughout. Challenging situations such as waves and wind from the side or crossing traffic in conjunction with strict safety and energy-efficiency requirements, make its design, verification, and assurance daunting tasks. With the present paper, we hope to be able to convince the reader that co-simulation-based optimization and adaptive stress testing are fitting tools for decreasing the risk in the design of such large coupled systems.

2 Optimization

There are four main challenges when it comes to optimization of large coupled systems: the number of design parameters to consider is large; there are multiple, possibly contradictory, objectives for the system performance; the objective functions are globally non-convex so gradient-based optimization may yield only local optima; and substantial computational cost can severely restrict the exploration of the solution space. Significant progress has been made on these issues in the field of *multidisciplinary design optimization* (MDO) (Martins and Lambe 2013; Wang, Fan, and Qiang 2023), which has so far found most of its applications in the aerospace and automotive industries, and more recently in the maritime domain (Ojo, Collu, and Coraddu 2022; Serani, Scholcz, and Vanzi 2024).

2.1 Background

Co-simulation-based optimization has recently awoken interest across various domains and applications as a natural extension of the utilization of co-simulation's strengths: the decoupling of the component modeling and system simulation processes, the ability to hide sensitive information within black-box models, the use of software tools and solvers most suited for the individual subsystems, and the ability to distribute the simulation burden across several parallel processes. Additional challenges emerge when the objective function is computed on the basis of co-simulation, however:

- *Complex or unpredictable behavior.* Knowledge of how the physics inside a subsimulator is implemented is often spotty, or may lack entirely. More so, models need not be based on physics representations at all. A wide variety of model types may be used in co-simulations, including discrete event models, stochastic models, agent-based models, or even look-up tables. Hence, assumptions about the functional form of objective functions, such as linearity, convexity, or smoothness, can often not be made.
- *Lack of information.* Access to gradients is typically sparse, or lacking completely for most models. The same goes for access to system states, which are usually not accessible at all with co-simulation.
- *Lack of features.* Important features such as the ability to save and restore the simulation state are often not available.
- *Coupling errors.* Errors stemming solely from the co-simulation coupling lead to significant uncertainty with respect to the validity of the system response. In turn, this can cause issues with traditional algorithms due to numerical instability or invalid model outputs.
- *System reticulation.* How the system is reticulated (split into subsystems) for the purpose of co-simulation and how the coupling between these subsystems is realized can have a significant influence on accuracy and stability (Sadjina, Kyllingstad, Skjong, et al. 2017; Sadjina, Kyllingstad, Rindarøy, et al. 2019). This is especially worth keeping in mind if system reticulation and model interfaces are themselves subject to optimization (discrete component optimization).

In light of this, it is natural to turn towards algorithms that fall under the broad heading of *black-box optimization methods*.

Black-box optimization

Perhaps the simplest, and thus highly popular, method is *ranking and selection*, whereby a small number of preselected scenarios are compared manually via an objective function. It is used, for example, by Rüdenauer, Han, and Geimer (2012) to optimize the design of a tractor's front axle in a collaborative co-simulation setting, and by Wirth et al. (2017) to find (Pareto front) optimal geometry designs of an airfoil in a fluid-structure co-simulation coupling.

Among algorithmic methods, *meta-heuristic algorithms*, especially genetic algorithms and swarm intelligence optimization methods, are popular choices for co-simulation optimization (Vega and Chevrier 2024). The *constrained multi-objective evolutionary algorithm with multiple stages* (CMOEA-MS) is used by Tan et al. (2022) to optimize the design and the thermal–hydraulic performance of nuclear plate-type fuel. Another variant, *covariance matrix adaptation evolution strategy* (CMA-ES), is used by Ahmed, Oekermann, and Kirchner (2014) to minimize forces acting on a steering mechanism. An evolutionary algorithm is also used by Arslan, Suveren, and Moghaddam (2023) to optimize the design of a co-simulated cart-pole test system. Another example is the *non-dominated sorting genetic algorithm II* (NSGA-II) which is used by Hou et al. (2023) to optimize a valve of a ship HVAC system, or the *multi-objective genetic algorithm* (MOGA) used by Anguek and Bounab (2022) to

DOI 10.3384/ecp218785

optimize the design of a gun station bracket with respect to minimal mass and frequency response under stress constraints. The *differential evolution algorithm* (DEA) is used by Anderson et al. (2020) to find optimal dimensions and bandgaps of semiconductor layers to maximize the efficiency of a solar cell. *Swarm intelligence optimization* comprises nature-inspired meta-heuristic algorithms. One popular candidate is *particle swarm optimization* (PSO), an *a posteriori* algorithm which generates a set of alternative solutions (Pareto optimal set) and prunes them with multi-criteria methods to present to the decision maker for a final choice. It is used by Zadeh, Sayadi, and Kosari (2019) to introduce a multi-objective collaborative optimization framework based on surrogate models, albeit not in conjunction with co-simulation.

Surrogate models

It is worth noting that the term *surrogate model* can have two meanings in the context of simulation-based optimization. In the above-mentioned paper by Zadeh, Sayadi, and Kosari (2019), it refers to an approximation of the objective function(s) which is being maintained and updated by the optimization algorithm based on selective evaluation of the actual objective function. The purpose is generally to reduce the number of computationally expensive direct evaluations needed. Other methods in this vein include the *radial basis function optimization* (RBFOpt), which is used by Waibel, Evins, and Carmeliet (2019) to sample an unknown black-box cost function for the study of the coupling between building energy demand and supply, and *constrained optimization by linear approximation* (COBYLA), used by Yamanee-Nolin et al. (2020) to optimize an evaporator process with respect to minimal product stream mass flow oscillations.

The other meaning of "surrogate model" refers to models used *within* the simulation(s) that form the basis for the objective function(s). Here, the point is to replace high-fidelity models with approximated models that are faster to calculate, to replace black-box models with white-box approximations that provide more information (e.g. derivatives), or both. Examples for co-simulation optimization include the use of weighted sums of several individual surrogate models based on either radial basis functions, response surface models, or Kriging surrogate models (Hou et al. 2023), or using feed-forward artificial neural networks to approximate objective functions and calculate gradients (Tuli et al. 2022).

Looking inside the box

Despite the challenges that co-simulation brings to the optimization table, things are not necessarily quite as bad as they may first appear: A co-simulation itself is not entirely a black box, even when its individual subsystems are. We do have access to the connection structure and the signals exchanged between the models, and this can be used to estimate errors and speed up the simulations (Kyllingstad, Sadjina, and Skjong 2024). Moreover, at least some of the subsystems may just as well *not* be black-box models. Instead, they may reveal some of their internal workings to us, granting access to gradients and, though rarer, access to system states. For example, if both the decision variables and the objective function are associated with such model types, it may still be possible to use gradient-based methods.

Gradient-based optimization generally requires derivatives and may only yield local optima because of globally non-convex objective functions. Used in conjunction with co-simulation, examples include *back-propagation of gradients with respect to input* (GOBI) which is proposed by Tuli et al. (2022) to find optimal scheduling decisions for fog computing, or *block coordinate descent* (BCD) used by Sadnan et al. (2021) to solve the decentralized optimal power flow problem in a distribution grid.

Combining optimization and co-simulation

Most examples of co-simulation optimization use an optimization routine *on top* of the co-simulation loop. Ranking and selection currently seems to be the most common such strategy, solely making up almost 45% of all methods investigated by Vega and Chevrier (2024). It consists of simply running a batch of pre-selected co-simulation scenarios, collecting and analyzing the results, adjusting the design, and repeating until satisfied. Ranking and selection can also be supplemented with post-processing techniques such as Pareto front (Wirth et al. 2017), *goal-driven optimization* (GDO) (Zhaoju et al. 2019), or by using a weighted sum of multiple objective functions (He et al. 2021). Verification by co-simulation is a slightly different take used by Assadi et al. (2023) to gauge the thermal-management performance of a high-density electric-vehicle fast charger based on the topology of a heatsink that was first optimized in a (monolithic) thermo-fluid simulation.

A different approach is *optimization-in-the-loop* whereby optimization happens inside of a co-simulation. One such example is *control optimization* which aims to move the system along a desired path as it evolves dynamically. The objective function then depends on a set of actions taken at discrete time instances. This could, for example, be relevant for dynamic positioning or routing tasks. Control optimization is used by Bharati, Chakraborty, and Darrah (2021) to find the optimal control of consumers and generators in a hardware-in-the-loop co-simulation of the interaction of a real-World microgrid facility with a real-time model of a power grid. It is also used by Tuli et al. (2022) for scheduling optimization, or by Sadnan et al. (2021) to minimize communications between agents controlling subgrids in a distribution power system for optimal decentralized power flow.

2.2 Research needs

Some black-box optimization methods seem to remain relatively unexplored in the context of co-simulation still. A

prominent example is the class of methods that go under the heading of *Bayesian optimization* (Garnett 2023), which has seen a resurgence in recent years due to its usefulness in machine learning, in particular for tuning neural networks. In Bayesian optimization, the objective function is treated as a random variable to be inferred in light of prior expectations (i.e., assumptions) and data collected (i.e., function evaluations). In our opinion, this seems like a promising approach when co-simulation is involved, one that should be investigated further.

A fact that does not seem to be discussed much in the current literature on co-simulation-based optimization is that co-simulation coupling gives rise to errors that are not present in a monolithic simulation of the same system. As mentioned, this exacerbates issues with simulation accuracy and stability, both of which are difficult to estimate or control for co-simulations (Kyllingstad, Sadjina, and Skjong 2024). A systematic way needs to be found to make sure that the optimization is robust enough to deal with this situation and to avoid over-optimization. In addition, methods for co-simulation error control can help speed up simulations and thus prove useful in optimization contexts.

Lastly, exactly how optimization and co-simulation can or should be used together is a matter that appears to be far from settled. More generally, a systematic analysis identifying which optimization methods work best for system optimization based on co-simulation, under which conditions, and for which cases seems to be lacking. This is also reflected in a large variety of strategies to be found in the literature, see Section 2.1.

2.3 Example: Optimizing the design of an autonomous battery-electric ferry

A fully battery-electric ferry needs to be fitted with a sufficiently large battery capacity and be as energy-efficient as possible to handle the vast range of environmental conditions it will encounter during operation. Maneuvering, acceleration, transit, and deceleration must all use as little energy as possible, while different environmental and loading conditions typically require different operational modes for optimality. The ferry design is also constrained by safety considerations. For example, it may have to handle (temporarily) reduced or completely unavailable charging capacity and stay maneuverable even under extreme weather conditions. Lastly, it has to be designed in a way that minimizes seasickness-inducing motions.

When setting up a co-simulation of such a system, a typical approach is to split it into subsimulators representing

- the hull (including effects of waves and currents),
- the power system,
- individual propulsors (e.g. propellers),
- individual pieces of heavy machinery (e.g. car ramp hydraulics),
- individual sensors (e.g. GPS and radar), and
- individual control systems (e.g. autopilot and power management system).

Depending on the vendor constellation involved in its construction, it may also make sense to split the power system further into subsystems representing battery packs, frequency converters, switchboards, back-up generators, and so on.

This system is characterized by strong interdependencies, foremost of which is that all on-board energy consumers are powered by the same, strictly limited, energy source. Multiple control systems are at play, controlling individual components, groups of components, or the entire ship, forming a hierarchy with both upwards and downwards dependencies. Hence, even though the system is loosely coupled and quite naturally separable for purposes of co-simulation, it is very tightly coupled in a system optimization perspective.

3 Adaptive Stress Testing

Traditionally, safety has been based on strategies such as introducing redundancy for critical components or by implementing safety functions that are triggered by a controller to prevent unsafe system states (Meulen and Myhrvold 2022). Such strategies are insufficient when dealing with complex and autonomous systems, however, where safety – or lack thereof – emerges as a consequence of the interactions of a large number of interconnected and interdependent controllers (Leveson 2012). This makes simulation-based validation of system safety and compliance a great tool for identifying the effects of these interactions at the system level.

Several approaches have been proposed for trying to find failures[2] in systems in complex interactions with their environment. They can broadly be divided into two categories (Lee, Mengshoel, et al. 2020): *Formal verification* attempts to use mathematical proofs and is able to identify failures as well as absence of failures. Of course, it is often ill-suited for complex engineering systems with stochastic environments. *Simulation-based sampling*, on the other hand, relies on the (manual or automatic) creation of environment instances and initial conditions, and performs searches over the parameter space to find failures in the simulated system–environment interactions. The main challenge is to handle the overwhelming set of possible parameters to test, not least figuring out which are the relevant ones. This calls for methods for automatically running, evaluating, and re-running simulations in a sequence which is *intelligently* guided towards the violation of desired system properties. One such method is *adaptive stress testing* (AST) (Lee, Kochenderfer, et al. 2015) which we believe is a promising candidate for use with co-simulation.

[2]Here, we understand a failure as a violation of a safety property (Corso, Moss, et al. 2021).

DOI 10.3384/ecp218785

3.1 Background

The principal idea behind AST is to try to identify the most likely failures of a complex system in an environment through actively guiding failure path sampling. This is especially useful when a formal mathematical falsification or the exhaustive search over the space of initial conditions, parameters, and over a significant length of time are not feasible. This will typically be the case for complex systems in stochastic environments for which the governing equations are not known. AST is a powerful method with applications in airspace safety (Lee, Kochenderfer, et al. 2015; Durling et al. 2021; Guo, Brittain, and Wei 2023), autonomous road vehicle safety and control (Koren, Alsaif, et al. 2018; Corso, Du, et al. 2019; Koren and Kochenderfer 2019; Corso, Lee, and Kochenderfer 2020; Corso and Kochenderfer 2020), software testing (Corradini et al. 2022; Durling et al. 2021; Hellhake et al. 2022), validation of AI systems (Julian, Lee, and Kochenderfer 2020), and collision avoidance of autonomous ships (Torben et al. 2023; Hjelmeland et al. 2022).

The classic formulation of AST (Lee, Kochenderfer, et al. 2015; Koren, Alsaif, et al. 2018) models the problem as a sequential Markov decision process and uses *deep reinforcement learning* with tree search to find action paths with a high failure likelihood. This approach either requires full access to the system states or, when unavailable, full control over the generation of random numbers in the system and the environment (Lee, Mengshoel, et al. 2020). The former case is rendered irrelevant by most industrial co-simulation use cases, while the latter is at least challenging to implement in practice.

3.2 Research needs

Unlike for optimization, we have been able to find little to no prior work on combining stress testing with co-simulation. Again, the black-box nature of co-simulations poses challenges, in this case when it comes to the interpretability of the results and the inference of causal chains, as well as issues regarding numerical accuracy and stability. At the same time, stress testing based on reinforcement learning (AST) presents itself as a good fit for co-simulation because it relies only on subsimulator inputs and outputs and thus goes together well with black-box models.

Depending on the specific restrictions and requirements of the case at hand, other approaches and formulations could prove useful and should be further investigated for use with co-simulation:

- If the failure paths need to be interpretable, *signal temporal logic* can be used together with *genetic programming* (Corso and Kochenderfer 2020) or *Gaussian processes* (Torben et al. 2023) to optimize action paths which lead to system failure. This approach also allows to encode domain-relevant requirements and compliance (such as minimal safety distances).
- The *backwards algorithm* (Koren, Nassar, and Kochenderfer 2021) can significantly speed up failure discovery with AST by first running it on a low-fidelity simulator before improving on the results using a high-fidelity simulator. Fidelity differences can induce spurious errors and disagreement in the discovered failures, though the algorithm is in principle able to learn to overcome these issues.
- The use of domain-specific heuristics for the reward function can improve AST performance, for example, by encoding improper system behavior or the inclusion of a dissimilarity measure between paths to try to avoid finding the same failure types repeatedly (Corso, Du, et al. 2019).
- If heuristic rewards are not available, AST performance can suffer from impeded learning due to a lack of reward information before a failure is found. The *go explore* (GE) algorithm can be used to improve AST performance under such conditions (Koren and Kochenderfer 2020).
- The use of a *recurrent neural network* with *long short-term memory* layers can yield a more robust and efficient detection of failures across the entire space of initial conditions (Koren and Kochenderfer 2019) by being able to learn relationships between different paths.
- AST also allows for differential stress testing (Lee, Mengshoel, et al. 2020) to compare failure paths against a baseline. This is useful, for example, when changes to an existing design need to be validated.

Other interesting approaches include the direct estimation of the distribution of failures (Corso, Lee, and Kochenderfer 2020), trying to find the boundary between compliant and non-compliant behavior based on parametric scenarios (Petrov et al. 2022), or extending AST to complex failures involving several autonomous agents and improper environment conditions together (Guo, Brittain, and Wei 2023).

3.3 Example: Stress testing the design of an autonomous battery-electric ferry

Our prior discussion of a battery-electric autonomous ferry in section 2.3 came short in terms of making sure that the design solutions are not only optimal but also *safe*. The ferry will have to handle all kinds of potentially dangerous situations, such as: vital sensor signals dropping out, becoming unreliable, or even actively being manipulated; collisions with crossing traffic; propulsors becoming unavailable; battery errors and other power electronics errors; or communications issues with a remote operation center. Worse, it will have to handle all sorts of *combinations* of such situations because the most severe system failures in complex systems stem from the interdependencies between subsystems.

A typical AST approach here would be to have a reinforcement learning agent define a policy that determines how a simulated scenario unfolds, for example how a crossing ship behaves or how the weather changes. The agent would then learn from the scenario outcomes, in particular how close the system came to failure, and in what sense. A proper evaluation of the latter could quite naturally depend on information which would normally be hidden in a co-simulation. One example might be the onboard control systems' internal representation of the state of the system and its environment; another might be the internal causal structure of a complex subsystem (e.g. the power system).

4 Outlook

Optimization and stress testing are not completely separable aspects of a complex system design workflow. Safety is an important design objective, and therefore something to be optimized for. Conversely, optimization of other factors could easily affect safety. Keeping to the maritime domain, one example would be a ship power plant which is tightly optimized for energy efficiency in ordinary operating conditions, but which does not have sufficient reserve power and therefore causes a critical black-out in extreme weather conditions. Having found good methods to implement optimization and AST with co-simulation-in-the-loop, the next obvious step is therefore to integrate the two processes in one design loop.

At least two strategies seem to be worthy of further investigation here:

- *nested*: including the results of stress testing when computing the objective function(s) in an optimization loop
- *sequential*: optimize the system, perform stress testing, evaluate, adjust optimization constraints/goals, repeat

We suspect that the first would, at least in principle, be capable of yielding the best results, but be very computationally expensive. The second might be more tractable in terms of computation time, but determining how to constrain or target the optimization procedure to reduce the likelihood of identified failure paths seems a highly nontrivial task, especially when dealing with black-box systems. Lastly, some synergies could potentially be exploited when it comes to surrogate models (optimization) or the use of low-fidelity simulators (AST with the backwards algorithm) to speed up implementations of co-simulation optimization and stress testing. But for now, this all remains conjecture and a highly interesting direction for future research.

This being a contribution to the *International Modelica & FMI Conference*, we would be amiss if we did not comment on FMI's role. Luckily, the standard already provides the complete feature set needed for co-simulation optimization and stress testing. The abilities to read and set derivatives and to store and restore model states, for example, are both already part of the FMI specification (Blochwitz et al. 2011). Therefore, it is up to the model developers to make use of the already implemented features to allow for their models to be used in optimization and stress testing settings.

Acknowledgements

This work was supported by the Research Council of Norway under the project *System optimisation and stress testing in co-simulations (OptiStress)* (grant no. 344238). We are also thankful for the supplementary funding from DNV, HAV Design, Norwegian Electric Systems, Corvus Energy, and HD Hyundai Europe Research & Development Center.

References

Ahmed, Mohammed, Christian Oekermann, and Frank Kirchner (2014-01). "Cosimulation environment for mechanical design optimization with evolutionary algorithms". In: *2014 World Symposium on Computer Applications & Research (WSCAR)*. Sousse, Tunisia: IEEE, pp. 1–6. ISBN: 978-1-4799-2806-4. DOI: 10.1109/WSCAR.2014.6916781.

Anderson, Tom H. et al. (2020-04). "Coupled optoelectronic simulation and optimization of thin-film photovoltaic solar cells". In: *Journal of Computational Physics* 407, p. 109242. ISSN: 00219991. DOI: 10.1016/j.jcp.2020.109242.

Anguek, Oussama and Belkacem Bounab (2022-12). "Multi-objective design optimization of a Turret's U-bracket mounted on moving platform". In: *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 236.24, pp. 11371–11388. ISSN: 0954-4062, 2041-2983. DOI: 10.1177/09544062221115106.

Arslan, Erdem, Memduh Suveren, and Seyed Tofigh Hosseini Moghaddam (2023-11). "ROS Gazebo and MATLAB/Simulink Co-simulation for Cart-Pole System: A Framework for Design Optimization". In: *2023 7th International Symposium on Innovative Approaches in Smart Technologies (ISAS)*. Istanbul, Turkiye: IEEE, pp. 1–7. ISBN: 979-8-3503-8306-5. DOI: 10.1109/ISAS60782.2023.10391735.

Assadi, Seyed Amir et al. (2023-03). "Electro-Thermal Co-design of a High-Density Power-Stage for a Reconfigurable-Battery-Assisted Electric-Vehicle Fast-Charger using Multi-Physics Co-simulation and Topology Optimization". In: *2023 IEEE Applied Power Electronics Conference and Exposition (APEC)*. Orlando, FL, USA: IEEE, pp. 1808–1815. ISBN: 978-1-6654-7539-6. DOI: 10 . 1109 / APEC43580 . 2023 . 10131627.

Bharati, Guna R., Sudipta Chakraborty, and Julien Darrah (2021-02). "A Co-simulation-based Hardware-in-the-loop Architecture for Validating Power Systems Optimization with Large-scale Grid Models". In: *2021 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*. Washington, DC, USA: IEEE, pp. 1–5. ISBN: 978-1-7281-8897-3. DOI: 10.1109/ISGT49243.2021.9372282.

Blochwitz, T. et al. (2011-06). "The Functional Mockup Interface for Tool independent Exchange of Simulation Models". In: *Proceedings of the 8th International Modelica Conference*. Linköping Electronic Conference Proceedings 63, pp. 105–114. DOI: 10.3384/ecp11063105.

Corradini, Davide et al. (2022-08). "Automated black-box testing of nominal and error scenarios in RESTful APIs". In: *Software Testing, Verification and Reliability* 32.5, e1808. ISSN: 0960-0833, 1099-1689. DOI: 10.1002/stvr.1808.

Corso, Anthony, Peter Du, et al. (2019-10). "Adaptive Stress Testing with Reward Augmentation for Autonomous Vehicle Validatio". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. Auckland, New Zealand: IEEE, pp. 163–168. ISBN: 978-1-5386-7024-8. DOI: 10.1109/ITSC.2019.8917242.

Corso, Anthony and Mykel J. Kochenderfer (2020-09). "Interpretable Safety Validation for Autonomous Vehicles". In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. Rhodes, Greece: IEEE, pp. 1–6. ISBN: 978-1-7281-4149-7. DOI: 10.1109/ITSC45102.2020.9294490.

Corso, Anthony, Ritchie Lee, and Mykel J. Kochenderfer (2020-09). "Scalable Autonomous Vehicle Safety Validation through Dynamic Programming and Scene Decomposition". In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. Rhodes, Greece: IEEE, pp. 1–6. ISBN: 978-1-7281-4149-7. DOI: 10.1109/ITSC45102.2020.9294636.

Corso, Anthony, Robert Moss, et al. (2021-10). "A Survey of Algorithms for Black-Box Safety Validation of Cyber-Physical Systems". In: *Journal of Artificial Intelligence Research* 72. ISSN: 1076-9757. DOI: 10.1613/jair.1.12716.

Durling, Michael et al. (2021-10). "Certification Considerations for Adaptive Stress Testing of Airborne Software". In: *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*. San Antonio, TX, USA: IEEE, pp. 1–9. ISBN: 978-1-6654-3420-1. DOI: 10.1109/DASC52595.2021.9594395.

Garnett, Roman (2023). *Bayesian optimization*. Cambridge: Cambridge university press. ISBN: 978-1-108-42578-0.

Gomes, Cláudio et al. (2019-05). "Co-Simulation: A Survey". In: *ACM Computing Surveys* 51.3, pp. 1–33. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3179993.

Guo, Wei, Marc Brittain, and Peng Wei (2023-10). "Safety Validation for Deep Reinforcement Learning Based Aircraft Separation Assurance with Adaptive Stress Testing". In: *2023 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC)*. Barcelona, Spain: IEEE, pp. 1–10. ISBN: 979-8-3503-3357-2. DOI: 10.1109/DASC58513.2023.10311238.

He, Haibin et al. (2021-05). "Structure Optimization of Natural Gas Engine's Intake Pipe Based on One-Dimensional and Three-Dimensional Coupled Simulation". In: *IOP Conference Series: Earth and Environmental Science* 781.4, p. 042029. ISSN: 1755-1307, 1755-1315. DOI: 10.1088/1755-1315/781/4/042029.

Hellhake, Dominik et al. (2022-06). "Towards using coupling measures to guide black-box integration testing in component-based systems". In: *Software Testing, Verification and Reliability* 32.4, e1811. ISSN: 0960-0833, 1099-1689. DOI: 10.1002/stvr.1811.

Hjelmeland, Hanna W. et al. (2022). "Identification of Failure Modes in the Collision Avoidance System of an Autonomous Ferry using Adaptive Stress Testing". In: *IFAC-PapersOnLine* 55.31, pp. 470–477. ISSN: 24058963. DOI: 10.1016/j.ifacol.2022.10.472.

Hou, Jianjun et al. (2023-03). "Co-Simulation Modeling and Multi-Objective Optimization of Dynamic Characteristics of Flow Balancing Valve". In: *Machines* 11.3, p. 337. ISSN: 2075-1702. DOI: 10.3390/machines11030337.

Julian, Kyle D., Ritchie Lee, and Mykel J. Kochenderfer (2020-09). "Validation of Image-Based Neural Network Controllers through Adaptive Stress Testing". In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. Rhodes, Greece: IEEE, pp. 1–7. ISBN: 978-1-7281-4149-7. DOI: 10.1109/ITSC45102.2020.9294549.

Koren, Mark, Saud Alsaif, et al. (2018-06). "Adaptive Stress Testing for Autonomous Vehicles". In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. Changshu: IEEE, pp. 1–7. ISBN: 978-1-5386-4452-2. DOI: 10.1109/IVS.2018.8500400.

Koren, Mark and Mykel J. Kochenderfer (2019-10). "Efficient Autonomy Validation in Simulation with Adaptive Stress Testing". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. Auckland, New Zealand: IEEE, pp. 4178–4183. ISBN: 978-1-5386-7024-8. DOI: 10.1109/ITSC.2019.8917403.

Koren, Mark and Mykel J. Kochenderfer (2020-09). "Adaptive Stress Testing without Domain Heuristics using Go-Explore". In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. Rhodes, Greece: IEEE, pp. 1–6. ISBN: 978-1-7281-4149-7. DOI: 10.1109/ITSC45102.2020.9294729.

Koren, Mark, Ahmed Nassar, and Mykel J. Kochenderfer (2021-09). "Finding Failures in High-Fidelity Simulation using Adaptive Stress Testing and the Backward Algorithm". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Prague, Czech Republic: IEEE, pp. 5944–5949. ISBN: 978-1-6654-1714-3. DOI: 10.1109/IROS51168.2021.9636072.

Kübler, R. and W. Schiehlen (2000-06). "Two Methods of Simulator Coupling". In: *Mathematical and Computer Modelling of Dynamical Systems* 6.2, pp. 93–113. ISSN: 1387-3954. DOI: 10.1076/1387-3954(200006)6:2;1-M;FT093.

Kyllingstad, Lars T., Severin Sadjina, and Stian Skjong (2024). *Error estimation and step size control with minimal subsystem interfaces*. arXiv:2406.17353v1. DOI: 10.48550/ARXIV.2406.17353.

Lee, Ritchie, Mykel J. Kochenderfer, et al. (2015-09). "Adaptive stress testing of airborne collision avoidance systems". In: *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*. Prague, Czech Republic: IEEE, pp. 6C2–1–6C2–13. ISBN: 978-1-4799-8940-9. DOI: 10.1109/DASC.2015.7311450.

Lee, Ritchie, Ole J. Mengshoel, et al. (2020-12). "Adaptive Stress Testing: Finding Likely Failure Events with Reinforcement Learning". In: *Journal of Artificial Intelligence Research* 69, pp. 1165–1201. ISSN: 1076-9757. DOI: 10.1613/jair.1.12190.

Leveson, Nancy G. (2012-01). *Engineering a Safer World: Systems Thinking Applied to Safety*. The MIT Press. ISBN: 978-0-262-29824-7. DOI: 10.7551/mitpress/8179.001.0001.

Martins, Joaquim R. R. A. and Andrew B. Lambe (2013-09). "Multidisciplinary Design Optimization: A Survey of Architectures". In: *AIAA Journal* 51.9, pp. 2049–2075. ISSN: 0001-1452, 1533-385X. DOI: 10.2514/1.J051895.

Meulen, Meine van der and Tore Myhrvold (2022). *Demonstrating safety of software-dependent systems*. DNV AS. ISBN: 978-82-515-0324-2.

Ojo, Adebayo, Maurizio Collu, and Andrea Coraddu (2022-12). "Multidisciplinary design analysis and optimization of floating offshore wind turbine substructures: A review". In: *Ocean Engineering* 266, p. 112727. ISSN: 00298018. DOI: 10.1016/j.oceaneng.2022.112727.

Petrov, Aleksandar et al. (2022-10). "HiddenGems: Efficient safety boundary detection with active learning". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Kyoto, Japan: IEEE, pp. 5147–5154. ISBN: 978-1-6654-7927-1. DOI: 10.1109/IROS47612.2022.9982243.

Rüdenauer, Andreas, Song Han, and Marcus Geimer (2012). "Optimization of the development process of mobile machines using a standardized co-simulation". In: *Landtechnik* 67.2, pp. 122–126.

Sadjina, Severin, Lars T. Kyllingstad, Martin Rindarøy, et al. (2019-02). "Distributed Co-simulation of Maritime Systems and Operations". In: *Journal of Offshore Mechanics and Arctic Engineering* 141.1, p. 011302. ISSN: 0892-7219, 1528-896X. DOI: 10.1115/1.4040473.

Sadjina, Severin, Lars T. Kyllingstad, Stian Skjong, et al. (2017-07). "Energy conservation and power bonds in co-simulations: non-iterative adaptive step size control and error estimation". In: *Engineering with Computers* 33.3, pp. 607–620. ISSN: 0177-0667, 1435-5663. DOI: 10.1007/s00366-016-0492-8.

Sadnan, Rabayet et al. (2021-07). "Distributed Optimization for Power Distribution Systems with Cyber-Physical Co-Simulation". In: *2021 IEEE Power & Energy Society General Meeting (PESGM)*. Washington, DC, USA: IEEE, pp. 1–5. ISBN: 978-1-6654-0507-2. DOI: 10.1109/PESGM46819.2021.9638251.

Serani, Andrea, Thomas P. Scholcz, and Valentina Vanzi (2024-12). "A Scoping Review on Simulation-Based Design Optimization in Marine Engineering: Trends, Best Practices, and Gaps". In: *Archives of Computational Methods in Engineering* 31.8, pp. 4709–4737. ISSN: 1134-3060, 1886-1784. DOI: 10.1007/s11831-024-10127-1.

Tan, Jiatong et al. (2022-05). "Multi-Objective Optimization Design of Plate-type Fuel via Co-simulation Method". In: *Annals of Nuclear Energy* 169, p. 108914. ISSN: 03064549. DOI: 10.1016/j.anucene.2021.108914.

Torben, Tobias Rye et al. (2023-04). "Automatic simulation-based testing of autonomous ships using Gaussian processes and temporal logic". In: *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability* 237.2, pp. 293–313. ISSN: 1748-006X, 1748-0078. DOI: 10.1177/1748006X211069277.

Tuli, Shreshth et al. (2022-01). "COSCO: Container Orchestration Using Co-Simulation and Gradient Based Optimization for Fog Computing Environments". In: *IEEE Transactions on Parallel and Distributed Systems* 33.1, pp. 101–116. ISSN: 1045-9219, 1558-2183, 2161-9883. DOI: 10.1109/TPDS.2021.3087349.

Vega, Diego Alejandro and Vincent Chevrier (2024-08). *Co-simulation-based optimization: state of the art*. hal-04668193, version 1.

Waibel, Christoph, Ralph Evins, and Jan Carmeliet (2019-05). "Co-simulation and optimization of building geometry and multi-energy systems: Interdependencies in energy supply, energy demand and solar potentials". In: *Applied Energy* 242, pp. 1661–1682. ISSN: 03062619. DOI: 10.1016/j.apenergy.2019.03.177.

Wang, Chong, Haoran Fan, and Xin Qiang (2023-10). "A Review of Uncertainty-Based Multidisciplinary Design Optimization Methods Based on Intelligent Strategies". In: *Symmetry* 15.10, p. 1875. ISSN: 2073-8994. DOI: 10.3390/sym15101875.

Wirth, Nadja et al. (2017). "Analysis and Optimization of Flow Around Flexible Wings and Blades Using the Standard Co-simulation Interface MpCCI". In: *Recent Progress in Flow Control for Practical Flows*. Ed. by Piotr Doerffer, George N. Barakos, and Marcin M. Luczak. Cham: Springer International Publishing, pp. 283–321. ISBN: 978-3-319-50568-8. DOI: 10.1007/978-3-319-50568-8_15.

Yamanee-Nolin, Mikael et al. (2020-03). "Trajectory optimization of an oscillating industrial two-stage evaporator utilizing a Python-Aspen Plus Dynamics toolchain". In: *Chemical Engineering Research and Design* 155, pp. 12–17. ISSN: 02638762. DOI: 10.1016/j.cherd.2019.12.015.

Zadeh, Parviz Mohammad, Mohsen Sayadi, and Amirreza Kosari (2019-01). "An efficient metamodel-based multi-objective multidisciplinary design optimization framework". In: *Applied Soft Computing* 74, pp. 760–782. ISSN: 15684946. DOI: 10.1016/j.asoc.2018.09.014.

Zhaoju, Qin et al. (2019-11). "Diesel engine piston thermo-mechanical coupling simulation and multidisciplinary design optimization". In: *Case Studies in Thermal Engineering* 15, p. 100527. ISSN: 2214157X. DOI: 10.1016/j.csite.2019.100527.

ShipSIM: A Modelica Library for Ship Maneuverability Modeling and Simulation

Basilio Puente Varela [1] M. Dolores Fernández Ballesteros [1] M. Isabel Lamas Galdo [2] Luis Carral [3]

[1]PhD undergraduate, Universidade da Coruña, Spain, {basilio.puente,dolores.fernandez}@udc.es
[1]Ekium - Techno Pro Hispania S.L., Spain, {bpuente,mdfernandez}@tphispania.com
[2]Departamento de Ciencias de la Navegación e Ingeniería Marina, Universidade da Coruña, Spain, {isabel.lamas.galdo}@udc.es
[3]Departamento de Ingeniería Naval e Industrial, Universidade da Coruña, Spain, {l.carral}@udc.es

Abstract

This paper introduces the ShipSIM, a novel free (standard conforming) Modelica library for modeling and simulation of ship maneuverability. This first Modelica library in the field of ship propulsion and maneuverability provides the components and flexibility to model the ship propulsion, hydrostatics and hydrodynamics to develop ship maneuvering simulations fully compatible with the Modelica Standard Library components.

In this paper is presented the library's key features and structure and introduce the underlying physical and mathematical foundations and modeling approaches. In addition, current implementation status, applicability limits and future development is discussed.

The library development is coordinated by the authors and it is used in several MSc thesis. ShipSIM library is available on https://github.com/BasilioPV/ShipSIM

Keywords: Modelica Library, ship maneuverability, marine engineering, ShipSIM

1 Introduction

Navigation safety is essential for all ships and specially for those that carry dangerous goods. On this sense, maneuverability is recognised as an ability to avoid dangerous situations specially near ports, where the coast, traffic and harbour facilities produces significative hazards.

For this reason, the International Maritime Organization (IMO) was increasing the maneuvering requirements to ships by means of A.601(15) (Organization 1987) [1] and later by MSC.137(76) (Organization 2002a) [2] and their associated guidelines (Organization 2002b) [3] where mathematical models can be used to demonstrate manoeuvring compliance on design stages. In addition, the same mathematical models can be used to correct sea trials performed with environmental conditions that deviate to those indicated on the regulations. On this sense, the integration of 1D simulation on Model Based Systems Engineering (MBSE) is clearly an advantage on shipbuilding.

Computational Fluid Dynamics (CFD) has great relevance on marine industry and nowadays is essential to estimate ship resistance in conceptual and basic designs, prior to towing tank tests and, in some cases, replacing them. But CFD has the disadvantage of large computation time, more man hours to produce a model, and low reusability when compared with 1D simulation. On the other hand, CFD can capture small design details that 1D simulation parameter cannot and the results was in general more precise than 1D simulations, especially when flow shedding phenomena occurs.

Previous approaches on ship specific simulation libraries using Modelica was SEECAT[4] from Bureau Veritas (Marty et al. 2012). This library provides a full set of ship specific components to simulate a power plant from the energy and fuel consumption perspective on a determined route. The main difference with ShipSIM is that on SEECAT only 1DoF is modelled, so manoeuvrability simulations are not possible.

DNV R&D in Greece has developed COSSMOS[5] library in gPROMS language (Dimopoulos and Kakalis 2010)(Dimopoulos, Georgopoulou, et al. 2014) focused on process simulation of ship energy systems. This tool is used in several ship types, such as LNG carriers (Dimopoulos, Stefanatos, and Kakalis 2016) to analyse from process point of view the ship systems. COSSMOS does not provide movements or navigation simulation, so manoeuvring or alternative means of propulsion, such as wingsails, are not possible to simulate.

Other applications can be found in MANSIM software (Aksu and Türkay 2017), that uses the same methodology of ShipSIM, but it lacks connectivity with other components or software. This inconvenience is found in other commercial software, such as MARIN SHIPMA 7 or ProteusDS ShipMo3D toolset.

[1]A.601(15) Provisions and display of manoeuvring information on board ships, 1987

[2]MSC.137(76) Standards for ship manoeuvrability, 2002

[3]MSC/Circ. 1053 Explanatory notes to the standards for ship manoeuvrability, 2002

[4]Ship Energy Efficiency Calculation and Analysis Tool

[5]COmplex Ship Systems MOdelling and Simulation

The rise of the digital twin technology on shipbuilding industry creates the needs of develop simulation models that can run at a speed several times less than real time to aid the operators into their decisions (predictive) or check if the system behaviour is the expected (diagnostic). Ship systems are composed by several components from many manufacturers, so the simulation model needs to integrate component from different sources. For this reason, the use of Modelica Association technologies, such as FMI (Modelica Association 2014) Standard[6] enables reusability of components and efficient system simulation model development.

2 ShipSIM library

ShipSIM Modelica library start this development on early 2021 as an technology demonstrator of the use of Modelica(Modelica Association 2021) in maritime industry, where the digital twin technology was started to gain attention. On July 2021, a webpage[7] was created to spread the use of Modelica in this sector. On April 2023, the authors and other significant Modelica users found *Modelica User's Group of Spain*[8] to spread the use of Modelica languaje and associated technologies in Spanish.

This library is licensed under 3-clause BSD license and the latest ShipSIM library version is available on `https://github.com/BasilioPV/ShipSIM`

2.1 Overview

ShipSIM library is built on top of MSL multibody library (Otter, Elmqvist, and Mattsson 2003) to provide an easier and seamless integration with other components of the Modelica Standard Library. The library is extensively documented for each component, in both documentation section and source code, where description of variables and parameters and main equations comments aids to understand the inner working of the library. Additionally, some parameters default value expression was taken from literature to aid in modelling on conceptual stages or training, where only few data is available.

2.2 Modelica library structure

Library structure adheres to usual MSL structure, where the library is presented to the user with a users guide, license, main authors, release notes, references and contact points. Next on the library are the examples, which provide sample systems that uses the main ShipSIM components. Validation and debugging sections are provided to check the proper working of some models when validated against the literature or test data.

Components section provides the main models developed for this library: ship hydrostatics, inertia, hydrodynamics, propellers and rudders. These components are the core of the library that enables simulation of ship manoeuvring.

3 Methods

The authors of this library considers two main features that deviates from the usual mathematical models found on the literature.

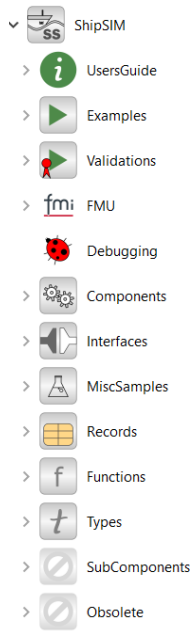


Figure 1. Main ShipSIM library structure.

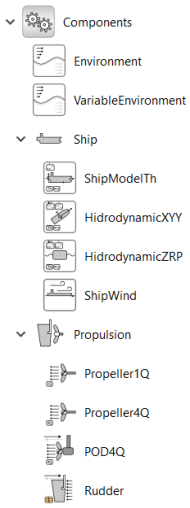


Figure 2. Hydrodynamics and propulsion components.

Usually ship movements are divided in two pair of 3DoF systems, to complete the 6DoF ship movement axis shown in Figure 3. The first one is the surge-sway-yaw, that represents the movement on the floatation plane, which is used for *manoeuvring* calculations. This 3DoF system has no restoring forces (*spring*) and only present *mass* and *damper* forces. The other 3DoF system is the roll-pitch-heave motion, which defines the *seakeeping* ability of the ship. These movements can be modeled as an *spring-mass-damper* system that oscillates around the equilibrium position.

The first feature is to provide a 6DoF enabled model from beginning, where the *manoeuvring* and *seakeeping* forces are modeled separately in order to enable only one 3DoF ability or both. This implementation requires to split the equivalent *mass-spring-damper* system into three main components:

1. Mass, inertia and hydrostatic restoring forces (corresponding to heave, pitch and roll movements)

[6] Functional Mock-up Interface

[7] *www.ShipTwin.com*, now offline and moved to *modelica-spain.org*

[8] *modelica-spain.org*

DOI 10.3384/ecp218793

2. Hydrodynamic damping forces relative to maneuvering (corresponding to surge, sway and yaw movements)

3. Hydrodynamic damping forces relative to seakeeping (corresponding to roll, pitch and heave movements)

This decomposition enables that different methods can be implemented into the library in order to have components with several fidelity levels or adjust the behavioural equations in other situations, such as shallow waters.

Other main feature is the segregation of the rudder from the manoeuvring model. Usually, all papers treat both components together, but for a sake of clarity the best is to be splitted in order to model non-conventional propulsion systems. In addition, the ship connection is placed at the common ship axis origin (rudder stock end at baseline) instead of the middle of the ship or the centre of gravity. With this position is easier to add other ship systems (rudder, propeller, cranes, etc...) because their position onboard is commonly described using these axis reference.

3.1 Ship hydrostatics and inertia

To calculate the new position of the centre of buoyancy, small rotations around equilibrium flotation theory was used. From this approach, the centre of buoyancy rotates around the longitudinal and transverse metacentre of the ship. This is the usual way to calculate hydrostatics forces and moments (Lewis et al. 1988).

Inertia matrix of the ship is usually simplified to only the main three axis inertia, discarding any cross inertia.

Hydrostatic data depends of the middle draft of the ship. Values of displacement, centre of bouyancy and metacentre position was provided using a custom expandable polynomial fit, with real custom numbers for the exponents. Hydrostatic data needs to be fitted previously to this kind of equation to be used.

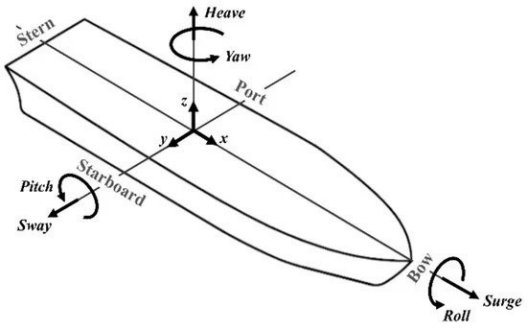


Figure 3. Ship movement axis.

3.2 Ship maneuvering hydrodynamics

Since the 1970's various mathematical models have been developed and validated to reproduce the ship motions during maneuvering (Norrbin 1970). Modern methods are based on the Maneuvering Modeling Group (MMG) standard methodology (Yoshimura 2005; Yasukawa 2015; Taimuri et al. 2020). This method accounts coupled effects from movements in different axis by polynomial functions where the coefficients can be obtained by standard tests.

This method is based on the following assumptions:

1. Hull in-plane forces are represented by quasi-steady flow reactions and described by non-dimensional coefficients referring to motion velocities and accelerations.

2. No shallow water effects are considered

3. Heave, pitch and roll motions are assumed to be small and represented by the linear relations of forces and motion components.

4. Memory effects are not taken under consideration.

Since mass and inertia effects are provided by other component, this component only accounts for the hydrodynamic force due to ship motion. According to MMG method (Yoshimura 2005; Yasukawa 2015), these in plane hull forces was defined by the following empirical equations:

$$X'_{Hull} = X'_{vv}v'^2 + X'_{vr}v'r' + X'_{rr}r'^2 + X'_{vvvv}v'^4 \tag{1}$$

$$Y'_{Hull} = Y'_{v}v' + Y'_{r}r' + Y'_{vvv}v'^3 + Y'_{vvr}v'^2r' + Y'_{vrr}v'r'^2 + Y'_{rrr}r'^3 \tag{2}$$

$$N'_{Hull} = N'_{v}v' + N'_{vvr}v'^2r' + N'_{vvv}v'^3 + N'_{r}r' + N'_{vrr}v'r'^2 + N'_{rrr}r'^3 \tag{3}$$

These equations provide the hydrodynamic forces as a function of several ship specific hydrodynamic parameters for surge ($X'_{vv}, X'_{vr}, X'_{rr}, X'_{vvvv}$), sway ($Y'_{v}, Y'_{r}, Y'_{vvv}, Y'_{vvr}, Y'_{vrr}, Y'_{rrr}$) and yaw ($N'_{v}, N'_{vvr}, N'_{vvv}, N'_{r}, N'_{vrr}, N'_{rrr}$), when multiplied to non-dimensional sway speed (v') and yaw (r'), raised to the power up to three. This methodology accounts for coupled response of the sway, surge and yaw movements. On MMG method, only the above mentioned coefficients were found relevant to define the behaviour of the ship when manoeuvring.

Ship resistance is modelled into the MMG method as a drag factor that multiplies the square of the vessel speed, to be added to the X'_{Hull} of Equation 1. Ship resistance is usually obtained from CFD studies and towing tank test. The results of ship resistance versus speed does not fit exactly with a square function. For this reason, on this library a custom polynomial function is introduced to fit the resistance results and enhance the precision. The end user shall perform a curve regression fit on the resistance curve data obtained by CFD or towing tank test, by using a polynomial function where the number of terms and power factors are not restricted to get a best fit of the data.

Hydrodynamics coefficients can be obtained from PMM (Conference 2017) [9] tests for a given ship. By default, this library provides an estimation based on several literature sources (Taimuri et al. 2020) that can be used for large ships with one rudder-propeller configuration in advance speed.

3.3 Ship propulsion and steering

Over years, Wageningen B-series propellers was used as default propellers on conceptual designs. The availability of propeller data and geometry makes easy to implement into simulation software (Oosterveld and Oossanen 1975). On this library, Wageningen B-series was implemented in both one-quadrant (Oosterveld and Oossanen 1975) and four-quadrant (Roddy, Hess, and Faller 2006) behaviour. One-quadrant represent the usual situation where the propeller has positive advance ratio and the ship is moving ahead. On the other hand, four-quadrant depends on the blade attack angle, and can be used for all the propeller situations, such as reversed thrust in a crash stop.

[9] Planar Motion Mechanism tests, see *ITTC Procedures 7.5-02-06-02 Rev05*

On this component, main propeller characteristics are defined, such as diameter, pitch ratio, number of blades and blade area ratio to select the Wageningen B-series propeller from the catalogue. Propeller inertia and added mass inertia can be also defined. By default, these parameters are calculated using built in correlation formulas for this kind of propellers.

To correct the behaviour of the propeller from open water test to propeller behind hull arrangement, coefficients of wake fraction, thrust deduction and rotative relative are used. Wake fraction represents the percentage of the ship advance speed that reach the propeller disc due to turbulence produced by ship hull. When the propeller is producing thrust, the water is accelerated by suction just ahead of the propeller disc. On this situation, the hull drag on the small part of the hull near the propeller is increased and treated as a propeller thrust deduction. Finally, the ship's hull is surrounding the propeller by above, which usually benefits the propeller working, produces a change in the propeller efficiency accounted with the rotative relative factor.

By default, these three factors are calculated using empirical formulaes (Harvald 1983; Holtrop 1984; Brix 1993) with main ship dimensions, but letting to the final user substitute for real values obtained from self-propulsion tests or CFD.

Finally, the water stream accelerated by the propeller directly impacts on the rudder, modifying the water speed on a determined zone. This rudder zone affected by the propeller is calculated by using flow straightening theory (*vena contracta*) (Taimuri et al. 2020).

Figure 4 shows the representation of rudder forces and flow velocity at the rudder plane. On this figure $V_R, V_{X,R}$ and $V_{Y,R}$ are the flow velocity of the water to the rudder, and their components in X and Y reference axis. The flow drift angle at rudder, β_R, is represented among with the rudder angle δ. Then, the hydrodynamic profile of the rudder will produce a lift force represented by $Y_r ud$.

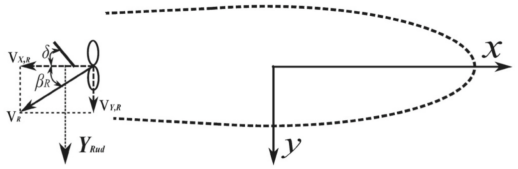


Figure 4. Representation of the rudder forces and flow velocity at the rudder plane. (Taimuri et al. 2020)

Ship steering model is composed by an hydrodynamic profile that rotates about a vertical axis, producing both lift and drag (Liu, Quadvlieg, and Hekkenberg 2016). This library contains several NACA 00xx symmetrical profiles lift and drag data, obtained for a range of Reynolds numbers and for attack angles from zero to 180 degrees (Sheldahl and Klimas 1981). In addition, a quick check of cavitation warning was implemented (Burrill 1955).

The water speed and direction is corrected with the propeller thrust affected zone to account for propeller-rudder interaction effect. In the relevant literature (Taimuri et al. 2020), lift reduction factor in way of the slip-stream radius is included. In a same manner, the influence of the hull ahead of the rudder can be included in the rudder force as a factor that represents the increase of the lateral force due to the presence of the hull above the rudder root. Also, the flow passing through the hull of the ship reduces the angle of attack on the rudder. This effect is usually accounted using flow straightening coefficient. This library tends to implement an interchangeable rudder profile data, that instead use a unique coefficient for lift and drag, uses a results table for full 180 degree angle of attack and several Reynolds number. For this reason, direct implementation of the formulae provided in the literature is not possible and adaptations are needed. At the time of this paper was written, neither lift reduction factor, increase of lateral force and flow straightening coefficient was included into the models.

3.4 Environment forces

Current speed and direction is used to calculate the relative water speed and direction and, therefore, include the current into the simulation. Apparent wind forces, composed by external wind speed and direction and ship's speed, can be estimated by the method presented in (Fujiwara, Ueno, and Nimura 1998). On this method, windage areas of the ship over the floatation and their respective centers of gravity was used. At the time of this paper was written, no sea waves forces was included into the library.

4 Application

4.1 Case study description

Turning circle is a classical maneuvering test that consist of turn the rudder up to maximum angle when the ship is navigating at full speed. The ships starts to navigate in circles where the following conditions shall be met (Organization 2002a):

1. The test speed (V) used in the Standards is a speed of at least 90% of the ship's speed corresponding to 85% of the maximum engine output.

2. Turning circle manoeuvre is the manoeuvre to be performed to both starboard and port with 35° rudder angle or the maximum rudder angle permissible at the test speed, following a steady approach with zero yaw rate.

3. The test shall be performed at deep, unrestricted water; calm environment; full load (summer load line draught), even keel condition and steady approach at the test speed.

Under the above conditions, the following criteria shall be complied:

1. Advance is the distance travelled in the direction of the original course by the midship point of a ship from the position at which the rudder order is given to the position at which the heading has changed 90° from the original course. The advance should not exceed 4.5 ship lengths (L).

2. Tactical diameter is the distance travelled by the midship point of a ship from the position at which the rudder order is given to the position at which the heading has changed 180° from the original course. It is measured in a direction perpendicular to the original heading of the ship. The tactical diameter should not exceed 5 ship lengths in the turning circle manoeuvre.

Figure 5 shows a graphical description of the definitions used above.

The ship selected for this example is a single shaft and rudder propulsion configuration. Main ship dimensions are described in

DOI 10.3384/ecp218793

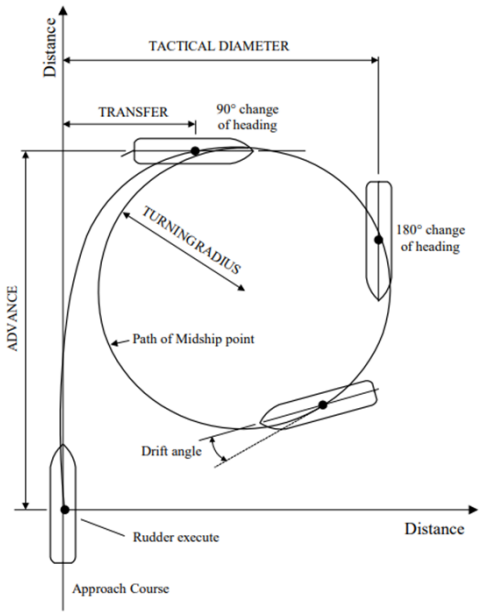


Figure 5. Turning circle manoeuvre. (Organization 2002a)

the Table 1. Hydrostatics was calculated by designing a custom hull lines with the main data shown in the table and fitting the values of displacement, metacentre radious and bouyance centre with polymonial fit. Windage areas was also included in the simulation model.

Propeller characteristics are defined in the Table 2, where a *Wageningen-B* propeller type was selected. Similarly, the rudder selected is a *NACA 0015* profile with a chord of 2 metres and a span of 3.5 meters. The rate of turn of the rudder comply with IMO regulations and was set to 2.5º/s, with a maximum rudder angle of 35º. Table 3 shows rudder main data.

Table 1. Main ship characteristics

Parameter	*Value*
Length	100 m
Beam	20 m
Draft	4 m
Block coefficient	0.693
Displacement	5,681ton
Num. of propellers	1
Num. of rudders	1
Trans. wind area	316 m^2
Lateral wind area	1,337 m^2

Figure 6 shows the simulation model diagram developed with ShipSIM library. The ship is modeled using three components. The first component is the mass, inertia and floatation forces that define the static floatation forces and inertia. The second one is the hydrodynamic forces, that define the forces induced by the movement of the ship in the water. In addition, a wind resistance force is added to the simulation. This wind force is lesser when comparing to the hull hydrodynamics forces.

The propulsion system is modeled using a four-quadrant propeller coupled with a rudder. The propeller speed is controlled by a constant signal and the rudder angle is controlled also by a constant signal that actuates from the beginning of the simulation.

Table 2. Propeller characteristics

Parameter	*Value*
Diameter	4 m
Type	*Wageningen-B*
Num. blades	4
Blade area ratio	1.0
Pitch diameter ratio	1.0

Table 3. Rudder characteristics

Parameter	*Value*
Chord	2 m
Span	3.5 m
Profile	*NACA 0015*
Rate of turn	2.5º/s
Max. rudder angle	35º

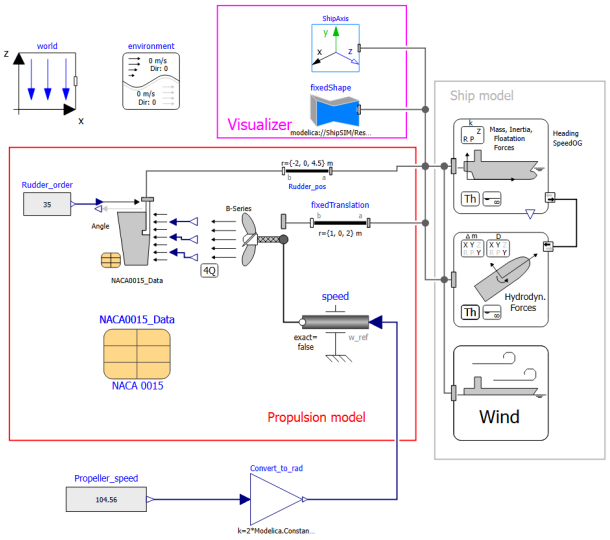


Figure 6. Simulation model diagram.

The ship is navigating at steady state with constant propeller r.p.m. where the rudder order is set to maximum rudder angle. In this condition, the rudder starts to rotate from the initial zero angle to maximum. Then, the ship start to turn and after a transient movement the ship start to navigate in circles, with a constant diameter.

4.2 Results and discussion

The simulation setup is set to a total time of 800 seconds with a time step of 0.2 seconds, using DASSL integration method with variable step size and order on OpenModelica v1.25.0. After compilation, the simulation time lasts less than 2 seconds, thus 400x faster than real time. Figure 7 shows the turning circle

Table 4. Initial conditions

Parameter	*Value*
Ship speed	13 *knot*
Propeller speed	104.56 *r.p.m.*
Rudder angle	0º
Initial rudder angle order	35º

result from this simulation.

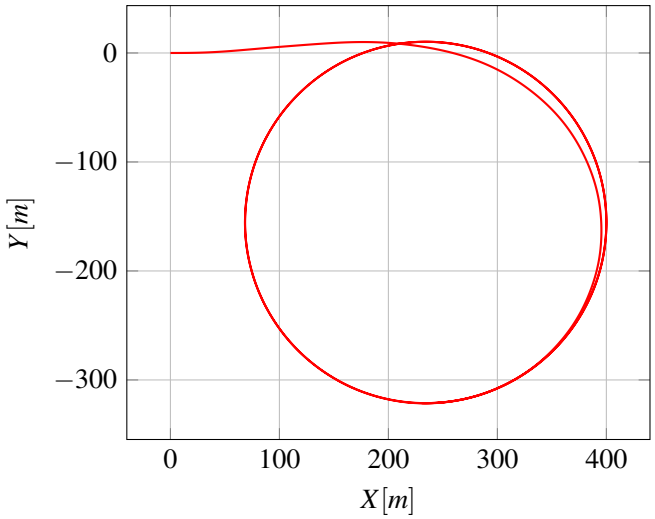


Figure 7. Turning circle test.

To check the compliance with the IMO regulations, Figure 8 shows the heading angle (Yaw) in red, surge position (X) in blue and sway position (Y) in green. As explained before, when the heading angle is equal to 90º, at 65.6 sec, the advance is 361.2 m (less than 4.5 times the ship length). In a similar way, when the heading angle is equal to 180º, at 138.8 sec, the tactical diameter is 291.3 m (less than 5 times the ship length).

This example demonstrate the benefits of perform quick simulation on conceptual design phase of ships to comply with IMO regulations regarding manoeuvrability.

5 Conclusion and outlook

This paper shows a proof of concept for the use of Modelica language into marine industry simulations. After describing the main maneuverability methods an application case is developed and their results presented. Some components need to have further development and validation to achieve the quality required to be used in real scenarios. This paper proves that physics 1D simulation is a reliable and fast method to provide simulation models to digital twin technologies, because the fast computation enables predictive usage.

The development of these physical models in Modelica has the big advantage of the acausal model definition, which enables a higher reusability and is easier to describe the equations that governs the behaviour of each model. On the other hand, collaboration between professionals is a key factor to develop high quality simulation models.

Acknowledgements

The diffusion of this work has been supported by Ekium - Techno Pro Hispania S.L.

The authors would also like to thank to the MSc students of Universidade da Coruña and Universidad Politécnica de Madrid for their support developing some components and using this library on their MSc thesis.

References

Aksu, Sinan and Semih Türkay (2017). *Theoretical Background and Application of MANSIM for Ship Maneuvering Simulations*. Accessed: 2025-07-12. URL: https://mansim.org/articles/Theoretical%20background%20and%20application%20of%20MANSIM%20for%20ship%20maneuvering%20simulations.pdf.

Brix, J. (1993). *Manoeuvring Technical Manual*. Hamburg: Seehafen Verlag.

Burrill, L.C. (1955). "The Phenomenon of Cavitation". In: *International Shipbuilding Progress* 2.15.

Conference, International Towing Tank (2017). *7.5-02-06-02 Rev.05 Procedure Captive Model Test*.

Dimopoulos, George G., Chariklia A. Georgopoulou, et al. (2014). "A general-purpose process modelling framework for marine energy systems". In: *Energy Conversion and Management* 86, pp. 325–339. DOI: 10.1016/j.enconman.2014.04.046.

Dimopoulos, George G. and Nikolaos M. P. Kakalis (2010). "An integrated modelling framework for the design, operation and control of marine energy systems". In: *26th CIMAC World Congress*. Bergen, Norway.

Dimopoulos, George G., Iason V. Stefanatos, and Nikolaos M. P. Kakalis (2016-09). "A process modelling approach to the evaluation of ship machinery configuration alternatives of LNG carriers". In: *International Journal of Thermodynamics* 19.3, pp. 146–156. DOI: 10.5541/ijot.5000156256.

Fujiwara, T., M. Ueno, and T. Nimura (1998-01). "An estimation method of wind forces and moments acting on ships". In: *Journal of the Society of Naval Architects of Japan*. DOI: 10.2534/jjasnaoe1968.1998.77.

Harvald, S.A. (1983). *Resistance and Propulsion of Ships*. Wiley. ISBN: 0-89464-754-7.

Holtrop, J. (1984). "A Statistical Re-analysis of Resistance and Propulsion Data". In: *International Shipbuilding Progress* 31.

Lewis, Edward V. et al. (1988). *Principles of Naval Architecture*. Second Revision. SNAME.

Liu, Jialun, Frans Quadvlieg, and Robert Hekkenberg (2016). "Impacts of the rudder profile on manoeuvring performance of ships". In: *Ocean Engineering*. DOI: 10.1016/j.oceaneng.2016.07.064.

Marty, Pierre et al. (2012-01). "Modelling of energy flows and fuel consumption on board ships: Application to a large modern cruise vessel and comparison with sea monitoring data". In: *Proceedings of the 11th International Marine Design Conference*. Bureau Veritas / STX France. Glasgow, UK.

Modelica Association (2014-07). *Functional Mock-up Interface for Model Exchange and Co-Simulation Version 2.0*. Tech. rep. Linköping: Modelica Association. URL: https://fmi-standard.org.

Modelica Association (2021-02). *Modelica – A Unified Object-Oriented Language for Systems Modeling. Language Specification Version 3.5*. Tech. rep. Linköping: Modelica Associa-

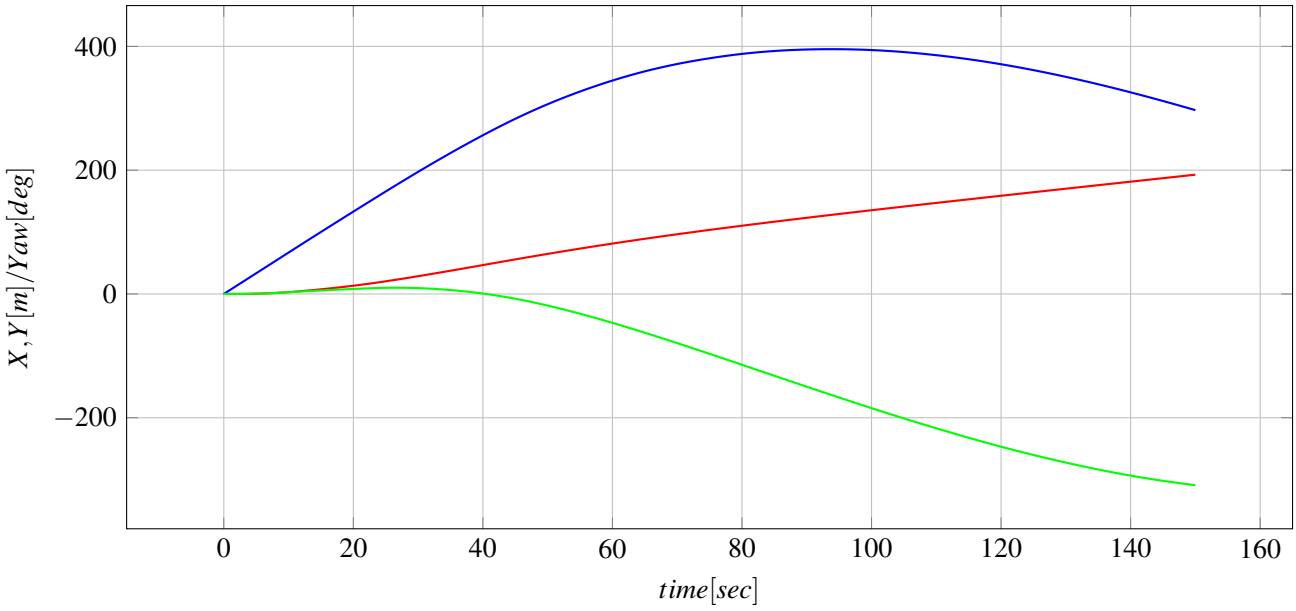


Figure 8. Turning circle test results in yaw, advance and tactical diameter.

tion. URL: https://specification.modelica.org/maint/3.5/MLS.html.

Norrbin, N.H. (1970). "Theory and observations on the use of a mathematical model for ship manoeuvring in deep and confined waters". In: *Proceedings of the 8th Symposium on Naval Hydrodynamics, Pasadena*.

Oosterveld, M.W.C. and P. Van Oossanen (1975-07). "Further computer-analyzed data of the Wageningen B-screw series". In: *International Shipbuilding Progress* 22.251.

Organization, International Maritime (1987). *Res. A.601(15) Provision and display of manoeuvring information on board ships*.

Organization, International Maritime (2002a). *MSC.137(76) Standards for ship manoeuvrability*.

Organization, International Maritime (2002b). *MSC/Circ. 1053 Explanatory notes to the standards for ship manoeuvrability*.

Otter, M., H. Elmqvist, and S.E. Mattsson (2003). "Proceedings of the 3rd International Modelica Conference". In.

Roddy, Robert F., David E. Hess, and Will Faller (2006-04). *Neural network predictions of the 4-quadrant Wageningen propeller series*. Tech. rep. NSWCCD-50-TR-2006/004. David Taylor Model Basin.

Sheldahl, Robert E. and Paul C. Klimas (1981-03). *Aerodynamic Characteristics of Seven Symmetrical Airfoil Sections Through 180-Degree Angle of Attack for Use in Aerodynamic Analysis of Vertical Axis Wind Turbines*. Tech. rep. SAND8O-2144. Sandia National Laboratories. DOI: 10.2172/6548367.

Taimuri, Ghalib et al. (2020). "A 6-DoF maneuvering model for the rapid estimation of hydrodynamic actions in deep and shallow waters". In: *Ocean Engineering*. DOI: 10.1016/j.oceaneng.2020.108103.

Yasukawa, Y. Yoshimura h. (2015). "Introduction of MMG standard method for ship maneuvering predictions". In: *Journal of Marine Science and Technology* 20, pp. 37–52. DOI: 10.1007/s00773-014-0293-y. URL: https://link.springer.com/article/10.1007/s00773-014-0293-y.

Yoshimura, Yasuo (2005). "Mathematical Model for Manoeuvring Ship Motion (MMG Model)". In: *Workshop on Mathematical Models for Operations involving Ship-Ship Interaction, Tokyo*.

DOI
10.3384/ecp218793

Modelica driven development of the thermal management control system for a zero emission yacht

A. van Dam[1] B. van Groos[1] C. von Ohlen[2] F. Theel[2] J. Brunnemann[2] J. Eiden[2]

[1]Foundation ZERO, Amsterdam, Netherlands, {alje.vandam, boudewijn.vangroos}@foundationzero.org
[2]XRG Simulation GmbH, Hamburg, Germany {vonohlen, theel, brunnemann, eiden}@xrg-simulation.de

Abstract

This paper describes the use of a MODELICA system model to support the development of a heat management control system for a fossil-free sailing yacht. Due to tight project timelines, the control system was developed and tested virtually, avoiding delays associated with waiting for the physical system to become available. The system model covers key functions such as heat recovery and heat dumping, enabling automated testing of various operational scenarios. This approach not only accelerates development but also reveals early insights into interactions between the control logic and system dynamics. The model is designed to be seamlessly replaced by the real system once it is built. Future comparisons between simulated and real-world performance will guide refinements to improve model accuracy and support model-based tuning of the control system.

Keywords: HVAC, control design, FMU, thermal management, zero emission yacht, ship

1 Introduction

In the year 2022 about 2% of the worldwide greenhouse gas was emitted by the shipping industry (International Energy Agency 2023). In absolute values this means an emission of 858 billion tons of greenhouse gas (Clarke et al. 2023). A transition to cleaner technologies is necessary to reduce those values. FOUNDATION ZERO is a relatively new organization which aims to share technical innovations to promote sustainable technologies in various sectors. It is active in the maritime industry, having previously provided its PowerHub to provide the sail team BCN in the 2024 America's Cup with fossil free energy (Foundation Zero 2024b). The source code of which has since been open sourced (Foundation Zero 2025b). One of the bigger items being covered by FOUNDATION ZERO is PROJECT ZERO, which is attacking the problem of shipping emissions with traditional propulsion with sails and the comfort of today. It is developing a 69-meter sailing yacht that aims to be the first of its kind to operate entirely free of fossil fuels. To achieve this, the project relies on the development of advanced technologies that balance energy generation, storage, and demand on board. The project also aims to share these innovations on an open-source basis, with the goal of accelerating the maritime industry's energy transition (Leslie-Miller and Someren 2022). Key systems enabling the yacht's fossil-free operation include a propulsion system that is capable of hydrogeneration by reversing the propellers, generating up to 750 kW of power that can be stored in a 5 MW h battery bank. Top-of-the-line insulation (Foundation Zero 2024c) and a heating, ventilation and air conditioning system (HVAC) that includes adiabatic cooling and natural ventilation through the mast (Foundation Zero 2025a) reduce the energy needed to control the climate on board, which typically accounts for up to 50 % of a yacht's energy use. Additional energy savings are achieved through the "Thermal Harvesting and Recovery System" (THRS), which is the focus of this paper. A simplified diagram is shown in Figure 1.

The THRS reduces electrical energy consumption by leveraging thermal energy (heat) alongside electrical energy. It harvests heat using custom-developed PVT panels capable of generating both electrical (PV) and thermal (T) energy, optimizing the limited surface area available on board (Foundation Zero 2024a). Additionally, it recovers waste heat from the propulsion system and electrical components, providing an extra source of thermal energy while reducing cooling demand, as this heat would otherwise accumulate inside the yacht. Thermal energy is stored using a phase change material (PCM) that acts as a thermal battery, providing heat to the domestic hot water supply and an adsorption chiller. The adsorption chiller supplies cool water to wall and ceiling panels throughout the yacht, reducing the load on the electrically driven HVAC system.

Recovering heat that would otherwise be discarded via cooling water and reusing it in place of electrical energy, for example for domestic hot water use, can offer significant energy savings. However, it also introduces the challenge of efficiently managing thermal harvesting, storage, and consumption while maintaining safe and stable operation of critical subsystems. This requires an intelligent control system with the ability to respond other systems on board, integrating inputs such as trip itineraries, weather forecasts, and consumption profiles,

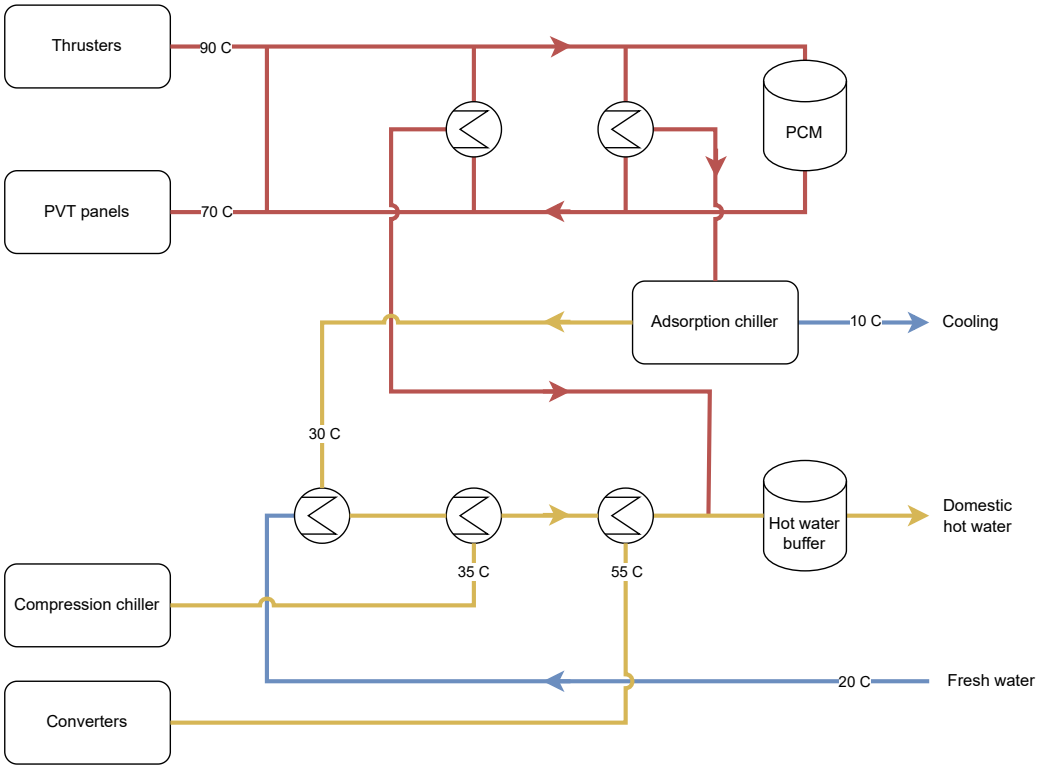


Figure 1. Simplified diagram of the thermal harvesting and recovery system

while also providing feedback to crew—for example, recommending optimal times for energy-intensive activities like laundry.

To manage the complexity of the system, it is divided into functional submodules, each with its own control logic to deliver basic functionality. The THRS system submodules are the high temperature circuit, the hot water circulation circuit, the buffer tank boost circuit, the buffer tank fill circuit, the chiller circuit, the booster circuit and two low-medium temperature circuits. These submodules operate under distinct control modes and setpoints, determining, for example, whether heat from the propulsion system should be harvested or rejected via seawater cooling—and at what temperatures. The high-over control system can then coordinate the modes and setpoints across submodules based on current or forecasted onboard conditions. The control modes of the submodules each have a flow configuration. That is achieved through valve and pump settings and includes basic control loops to maintain temperature and flow targets.

A more practical challenge is the tight timeline of the construction and commissioning of the yacht. In addition to these reasons, there are issues of availability and high costs, meaning that Hardware-in-the-Loop (HiL) or even proper testing of the controller in the real system are out of the question. This has motivated the creation of a dynamic model of the physical system, which facilitates co-development of the control logic alongside the physical system, effective troubleshooting before the system is installed, and isolation of individual subsystems for testing. MODELICA is used for the Software-in-the-Loop (SiL) approach to model the plant model. The controller is implemented in Python, as the goal is to use the implemented Python code in the real system as well. In addition, the entire project focuses on open access, so Python is preferred over conventional tools, which are mostly commercial. Further details of how the implementation of the control logic in Python is pointed out in Section 3. Once validated, the control logic can seamlessly transition from simulation model to real-world operation by replacing the dynamic model with the yacht's actual input/output interfaces during commissioning (see Figure 2). When replacing the plant model with the real system, latency effects are expected to occur, but these can be neglected due to the high capacities and inertia of the THRS. This paper demonstrates this approach for a submodules component of the THRS that recovers heat from the yacht's propulsion system.

The paper is structured as follows: In Section 2 a functional description of the thermal harvesting and recovery system is given. Section 3 describes how MODELICA is integrated in the project architecture and explains the interface between the developed MODELICA model and the rest of the tool chain. To demonstrate the viability of this approach, the focus lies on a single component of a submodule that recovers heat from

DOI 10.3384/ecp218801

the thrusters. A detailed description of the thrusters component and its modelling in MODELICA is given in Section 4. Section 5 presents simulation results of the MODELICA model coupled with the developed Python control. Finally, a summary and outlook are given in Section 6.

2 Description of the Thermal Harvesting and Recovery System

The THRS (see Figure 1) comprises components that produce, consume, or store thermal energy. When sailing—either in propulsion or hydrogeneration mode—the thrusters generate heat and require cooling, providing a potential heat source of up to 13.3 kW at 90 °C that can be recovered by the system. PVT panels contribute additional thermal input that is also available when at anchor, typically harvesting at around 70 °C during peak sunlight hours. Other onboard sources include the waste heat from the compression chiller and electrical converters, which offer a more constant source of energy, but at lower temperatures.

Thermal energy is consumed by domestic hot water demands at 48 °C (e.g. showers, laundry, and dishwashing) and by an adsorption chiller, which requires hot water between 50 °C and 95 °C to produce chilled water for onboard cooling. To balance fluctuating availability and demand throughout the day, heat can be stored in both a phase change material (PCM) storage unit and a hot water buffer tank. To manage thermal energy efficiently across different temperature levels, the system is divided into a high-temperature and a medium-temperature circuit. This split enables harvesting energy at both high and low temperatures - for example allowing use of the waste heat from the adsorption chiller to preheat water entering the medium-temperature loop. The high-temperature circuit primarily supplies the adsorption chiller and can supplement the medium-temperature circuit as needed. Figure 1 presents a simplified schematic of the system. For clarity, elements such as valves, pumps, and seawater coolers are omitted. In practice, each critical component includes a seawater-based backup cooler to ensure safe operation when thermal energy cannot be stored or utilized.

As an illustrative example, the thrusters component is highlighted, which is responsible for cooling the thrusters during operation and, when possible, feeding the recovered heat into the rest of the system. This component is described in detail in Section 4, which also presents its implementation in MODELICA.

3 Using MODELICA to design a control system

The main objective of the project underlying this paper is to develop a dynamic model of the THRS that can be used as a development environment for the control logic. A scheme of the according software architecture is shown in Figure 2: The box at the top shows the Python control module. The Module is implemented with pytransitions which is a library for finite state machines (Alexander Neumann 2025). It takes as input sensor values and computes the desired control outputs based on the control logic as output. Sensor values include temperature and flow sensor readings, but also states of other systems on board, such as the propulsion system. On the receiving

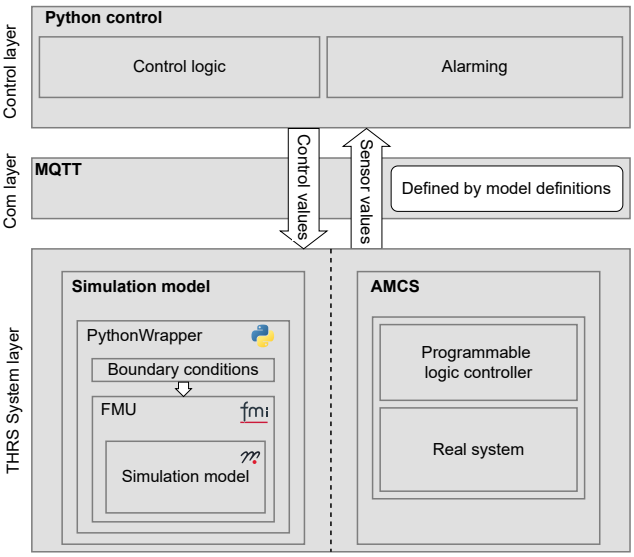


Figure 2. Architecture - Dividing into control layer, communication layer and THRS system layer. Simulation model and AMCS are interchangeable.

end of the control, shown at the bottom box, one finds either the simulation model or the ship's real alarming, monitoring and control system (AMCS). The AMCS provides an interface for critical systems on board, such as power management and propulsion control. It handles most of the ship's sensors data and control signals and is connected to the physical components of the THRS through a programmable logic controller (PLC). The interface between the control logic and the simulation or the AMCS is realized with the message queuing telemetry transport (MQTT) protocol (Banks et al. 2019).

In this setup, the THRS IO and control are completely separated, and simulation and AMCS are interchangeable. This means that the control logic can be developed and tested with the simulation model, and then seamlessly transitioned to the AMCS without changes to the control code. Other than changing the MQTT broker to which the

control code connects there is no meaningful difference between simulation and the actual system. Therefor potential difficulties which might ordinarily occur when going from model to compiled code don't apply. One can expect latency effects between the computer running the control code and the AMCS. It is assumed that those effects are small compared to thermal capacities and inertia of the THRS.

3.1 Setup of the THRS system model

The THRS components are modeled with the MODELICA library HVAC (XRG Simulation GmbH 2025) in combination with the MODELICA STANDARD LIBRARY (Modelica Association 2023). The HVAC library is designed for fast simulations of thermo-hydraulic and air distribution systems. It provides models for cooling towers, heaters, turbo machines, heat exchangers, heat pumps, chillers, humidifiers, water separators, storages, sensors, photovoltaic panels, thermal collectors, fittings, pipes, valves, sinks, sources and other HVAC relevant components (see example icons in Figure 3). To model the THRS a selection of the above-mentioned models are used. In addition to that, a model for PCM based on Brunnemann (2020) is integrated to the system. A detailed description of modeling the entire THRS would be beyond the scope.

The model of the thruster component (see Figure 4) is embedded into an interface model in MODELICA. Real inputs, representing control values or boundary conditions, and outputs, representing sensor values, are defined in a way to match the real system. To prevent sudden changes or unexpected jumps of the input signals, a first order block is placed behind the inputs to smooth them before they are passed on to the thrusters component. To use the MODELICA model as the replacement of the

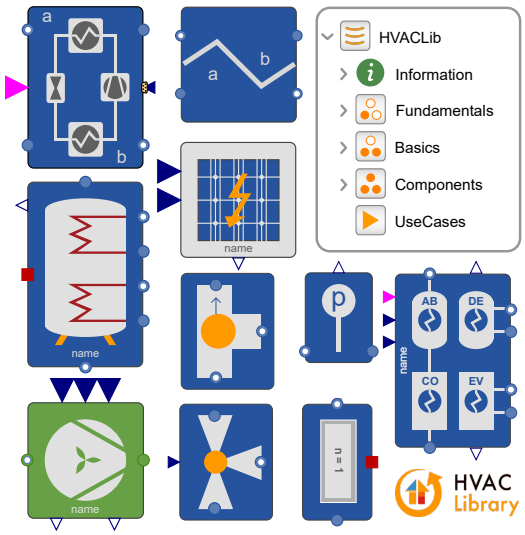


Figure 3. HVAC LIBRARY with example models for heat pump, buffer tank, fan, heat exchanger, photovoltaic panel, fitting, sensors and adsorption chiller

AMCS, it is compiled into a FUNCTIONAL MOCK-UP UNIT (FMU). A Python wrapper enables the FMU to communicate via MQTT with the control module (see Subsection 3.2). The development and compiling of the model is done in DYMOLA 2025X. Co-Simulation with the solver Cvode is selected. The FMI version chosen is 2.0. To enable recompilation of the FMU on another operating system, the source code export is included. It is recommended to deactivate the tools flag "evaluate". Another way to enable the model to communicate with the control module via MQTT is to use the MODELICA DEVICE DRIVERS LIBRARY (Thiele et al. 2017). Since there are more options to integrate a FMU into any workflow the compilation in a FMU is the preferred way. This opens greater flexibility in the event of changes to the process.

3.2 Python integration

The FMU is executed using FMPy (Dassault Systèmes 2025), enabling compilation and execution of the FMU in a Python environment. From here, a Python wrapper is constructed that provides the MQTT interface to the FMU. The wrapper is responsible for reading the control values from the MQTT broker and writing them to the FMU inputs. It also reads the sensor values from the FMU outputs and writes them to the MQTT broker. The data schema for all these values is defined using the Pydantic Python package (Colvin et al. 2025). Pydantic is a data validation and serialization library for Python, which allows for the definition of data models with type annotations. This enables automatic validation of the data being passed to and from the FMU and AMCS.

One of the key aspects of the setup is the ability to exchange the real system with the simulation model. This means that the control code is fully unaware of that difference. Specifically, the interface of the simulation (the sensor values it sends, and the control values it receives) should be identical to those of the real-world system (i.e. the AMCS). This is done as follows:

- Model definitions: Pydantic models are defined that describing sensor values and control values. The models follow the structure of the MQTT messages as defined by the AMCS.
- FMU mapping: The Pydantic models are translated to the FMU's input and output variables. This is done by maintaining a strict naming convention for the FMU's variables.
- FMU boundary condition input: The system model requires some additional inputs that are not part of the control values (e.g. seawater temperature), which are injected in the simulation as boundary conditions. These are also defined by Pydantic models as either constants or time series.

DOI 10.3384/ecp218801

- Unsimulated sensor values: any sensor values that are not simulated in the FMU are added to the simulation output (and must also be defined as simulation inputs). An example is the signals from the ship's propulsion control system indicating which thrusters are active.

With this setup there is a complete stand-in for the real-world system, abstracted behind a Python class which receives control values and output sensor values. This class can read and write MQTT messages and translate them to FMU inputs and outputs. The control loop reads the sensor values, enters those into the control logic and feeds the control values back into the simulation.

4 Thrusters Component in Modelica

This section describes in more detail the simulation model of the thrusters component within MODELICA (see Figure 4). Its main subcomponents can be identified as a circulating pump (1), the two thrusters, a forward thruster (4) and an aft thruster (8) and a seawater cooler (14). It also includes several control valves, fittings and different types of physical sensors ((3), (6), (7), (10), (16), (18), (19), (27), (28)). The medium, a 20 % glycol-water mixture, is assumed with constant fluid properties.

The module has three main operating modes, an idle mode, the heat recovery mode and the direct cooling mode, which are described in Section 5.

4.1 Circulation Pump

The flow in the thrusters component is provided by the circulation pump (1) (highlighted in light purple in Figure 4). For reasons of redundancy there are two parallel pumps which are represented by one circulation pump. The two identical pumps do not operate at the same time, so it is sufficient to use only one pump model with a built-in pump changeover sequence. The pump model is parametrised with one characteristic pump curve for a specific speed. The pump gets a relative speed as control parameter and operating points deviating from nominal values are approximated from characteristic curve with affinity laws.

4.2 Forward and Aft Thruster

As mentioned before, the thrusters of the yacht are generating heat during operation and must be cooled. Therefor the thruster model ((4), (8)) is simplified to transfer a certain heat load to the coolant and to have a linear pressure loss model, which is defined by nominal parameters. The heat load is given through an input and can be varied during the simulation. Forward and aft thruster are shown with a red frame in Figure 4.

4.3 Valves and fittings

There are three types of valves used in the thruster module, the switching valves ((11), (20)), the mixing valves ((13), (21), (22)) and the flow control valves ((5), (9)). The switching and mixing valves are three-port valves and are based on the same model. The flow control valves are two-port valves. Each valve is parametrised with a KV nominal value, provided by their data sheets. The opening for the two-port valves and the flow direction for the three-port valves are set as relative ratios and can be varied during the simulation. There is a leakage of 0.01 % to make the overall model more robust against numerical issues. The opening is implemented with a time delay behaviour.

The fittings can be divided into splits ((2), (25)) and joins ((12), (15), (17), (23), (24), (26) (29)). The joins are designed with a control volume-based approach and flows are ideal mixed. Splits can be used either with predefined split ratio or with a computed ratio provided by outer components which are generating mass flows.

4.4 Seawater cooler

The seawater cooler (14), highlighted in green in Figure 4, is implemented to guarantee the cooling of the two thrusters. The heat transfer is modelled as an eps-NTU approach (Hesselgreaves 2001). Nominal data from datasheets parameterize the model. A seawater medium with constant data at 30 °C is created for seawater side of the heat exchanger. A simple quadratic nominal pressure loss is implemented to model the relation between flow and pressure. The volume flow and temperature of the seawater are variables and can be set as inputs during the simulation.

5 Application Scenario

5.1 Thrusters control

Based on the sensor values coming from the simulation model, the control module determines control values to reach the desired setpoints. The control logic for the thrusters module covers several key functions:

- **Flow control:** The total flow is calculated as the sum of the setpoints for each individual thruster and is regulated by a PID controller that adjusts the pump speed accordingly. The flow is balanced between the thrusters using the flow control valves ((5), (9)). Each valve is controlled by a PID controller, with control values offset to ensure that one valve always remains fully open to avoid unnecessary counterpressure. The system also adapts to thruster activity—closing the flow control valve and reducing pump speed when only one of the two thrusters is active.
- **Mode switching:** As mentioned before, there are three operational modes : idle (no flow), heat recovery, and direct cooling. The control must support all modes. Each mode has distinct flow and temperature setpoints. Mode transitions are triggered by sys-

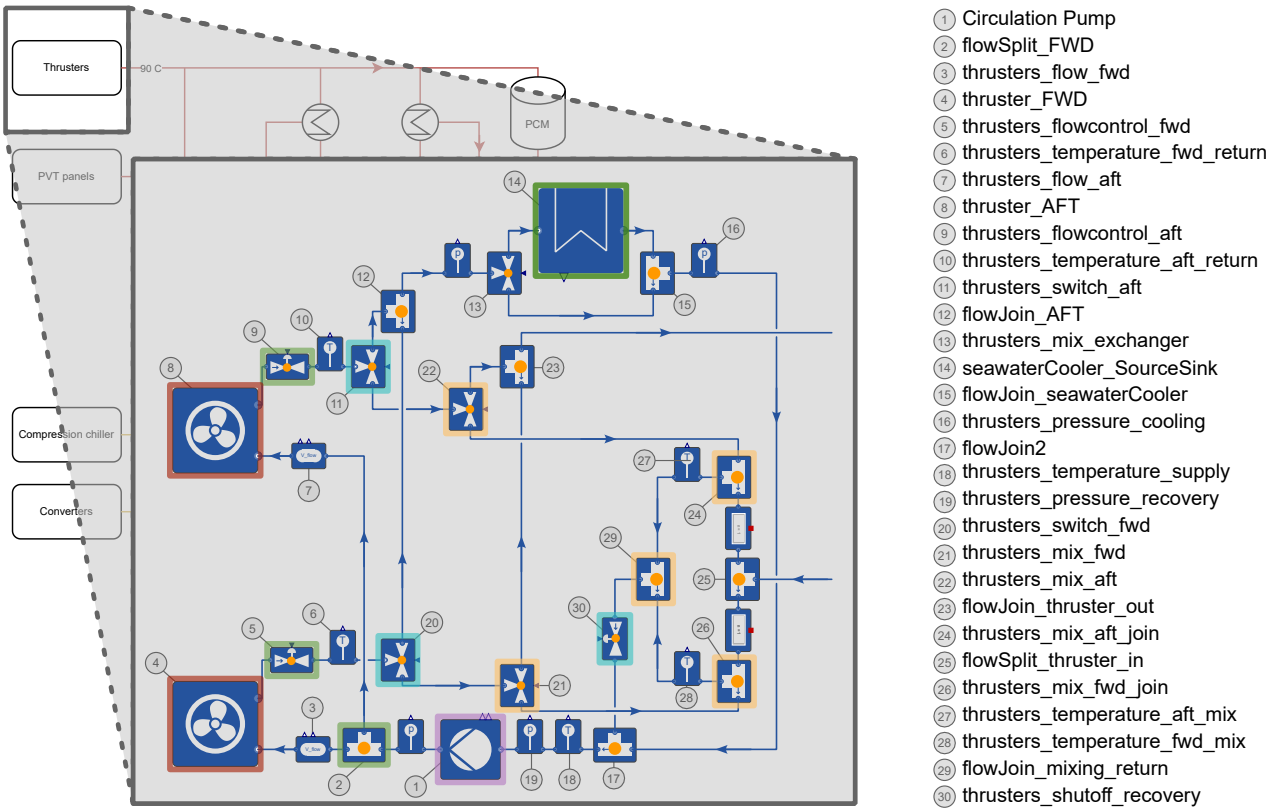


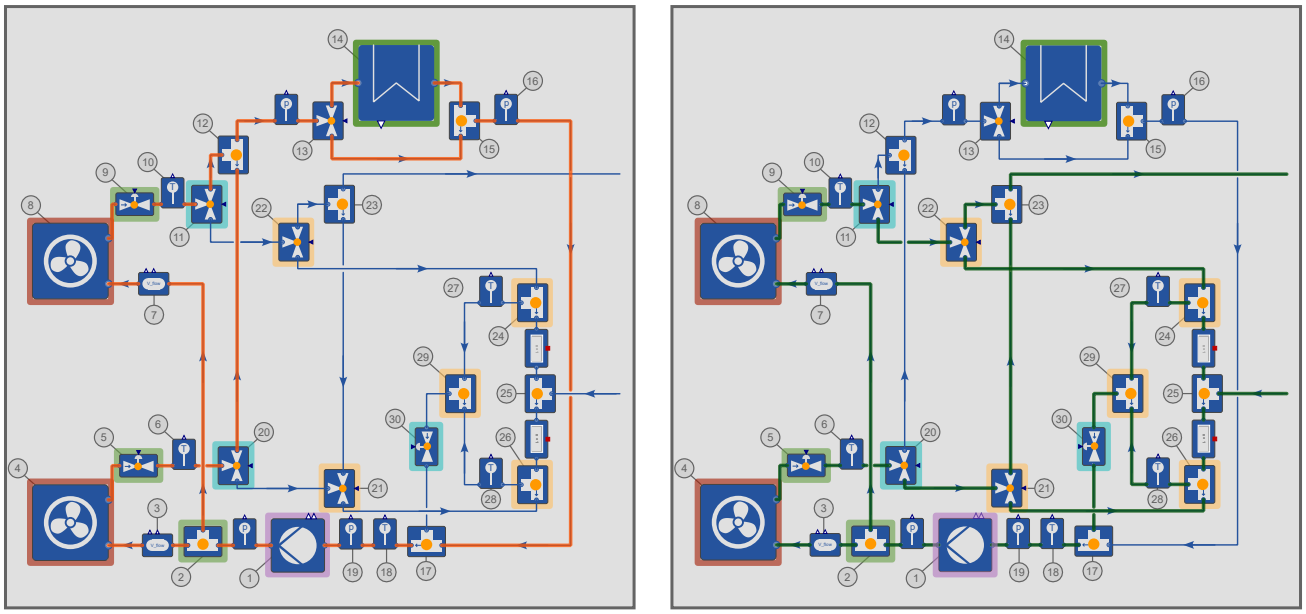
Figure 4. MODELICA model of the thrusters component with legend on the right side.

tem conditions — for example, engaging a thruster switches from idle to recovery, while an overheating alarm may switch from recovery to cooling. Overheating can occur when there is no thermal demand in the rest of the system, causing the inlet temperature to rise too high for safe thruster cooling. In such cases, excess heat must be dumped via the seawater cooler (14).

- **Cooling temperature control:** In direct cooling mode (see Figure 5a), the primary goal is to ensure safe thruster operation by rejecting heat directly, without recovering it for onboard use. It is triggered when the supply temperature of the thrusters passes a set threshold, to prevent overheating. The flow path in this mode is marked in orange. From the circulation pump (1) the flow starts its way through the forward (4) and aft thruster (8). The shutoff recovery control valve (30) is closed and the three-way valves after the forward thruster (22) and the aft thruster (11) are controlled to direct the flow to the flow join (12) where both streams are mixed. After passing the flow join the coolant reaches the three-way valve (13) in front of the seawater cooler (14). This valve is implemented to control the ratio between the flow through the seawater cooler (14) and its bypass flow. This is done by a PID controller to reach the desired setpoint temperature of return flow. From the seawater cooler (14) the coolant flows via a flow join (15) back to the circulation pump (1).

- **Recovery temperature control:** The heat recovery mode (see Figure 5b) is engaged when there is a demand for thermal energy (or available storage) in the rest of the system. The flow path in this mode is shown in green. From circulation pump to the three-port valves ((11), (20)) the flow takes the same pass as in direct cooling mode. Instead of the coolant flowing in the direction of the flow join (12) upstream of the seawater cooler, it heads into the two mixing three-port valves ((22), (21)) which are highlighted with a yellow frame. One part of the medium leaves the thrusters component via the flow join (23) into the high temperature circuit of the yacht. The other part of the medium is directed to two mixing flow joins ((24), (26)). Both are connected via the flow split (25) with supply flow from high temperature circuit into the thrusters component. After mixing both flows with the supply flow the water-glycol mixture is headed back to the circulation pump with the shutoff recovery control valve (30) which is opened in this mode. Regarding the control, the temperature of the return flow is controlled by a PID controller that adjusts the three-port mixing valves ((21), (22)) regulating the amount of coolant that recirculates past the thrusters to reach the desired return

DOI
10.3384/ecp218801



(a) Direct Cooling Model

(b) Heat Recovery Model

Figure 5. Operating modes of the thrusters component

Cooling mix setpoint	40	°C
Recovery thruster flow	10	l/min
Cooling thruster flow	22	l/min
Maximum temperature	70	°C
Recovery temperature setpoint	60	°C

Table 1. Control setpoints

Aft thruster heat	4.3	kW
Fwd thruster heat	9	kW
Seawater supply temperature	32	°C
Seawater supply flow	20	l/min
Module supply temperature	50	°C

Table 2. Simulation parameters

temperature. This way, a minimum return temperature of the module is guaranteed (flow which is leaving the join (23)).

The control logic is demonstrated through a simulated operation of the thrusters module governed by a Python-based controller. Figures 6 and 7 present consecutive segments of a two-hour simulation run. Figure 6 illustrates the first hour, during which the system transitions from idle into heat recovery mode with both thrusters active. The simulation parameters and control setpoints are provided in Tables 1 and 2, respectively.

Each figure includes a top plot showing control signals sent by the module, consisting of valve setpoints and the pump duty point, which regulates flow depending on the current control mode (cooling or recovery). The bottom three plots display system variables—primarily sensor readings—except for the module supply temperature, which is predetermined as a simulation parameter.

While the top plots of both figures appear similar, they represent different data: the uppermost shows valve setpoints (desired positions) given by the control, whereas the second plot reflects actual valve positions that are read as sensor values. These diverge because physical valve actuation takes time (approximately 1 degree per second), resulting in about 90 seconds for valve to open or close.

5.2 Heat recovery

In the first five minutes of the simulation, the thrusters are off, the system is in idle mode, and there is not flow. The seawater return temperature starts at the default 30 °C but stabilizes to 32 °C, as specified in the parameters.

After five minutes, the thrusters are activated and trigger the control to go into recovery mode. The pump starts running and the flow control valves of each thruster are opened. The aft and forward thruster return temperatures rise due to flow past their respective heat sources (9 kW and 4.3 kW). Once the aft return temperature reaches the 60 °C recovery setpoint, the *aft mix* valve opens, allowing heated return flow to exit the module (*module return*) and supply the THRS. Simultaneously, the aft inflow begins drawing from the *module supply*, lowering the *mix aft* temperature and ultimately returning it to the supply temperature of 50 °C as the mix valves fully opens.

The forward thruster, generating roughly half the heat of the aft thruster, warms more slowly. When its return temperature eventually reaches the 60 °C setpoint, the *fwd mix* valve begins opening. Due to insufficient heating power, the PID controller causes the valve position to oscillate before settling near 0.25. This behavior

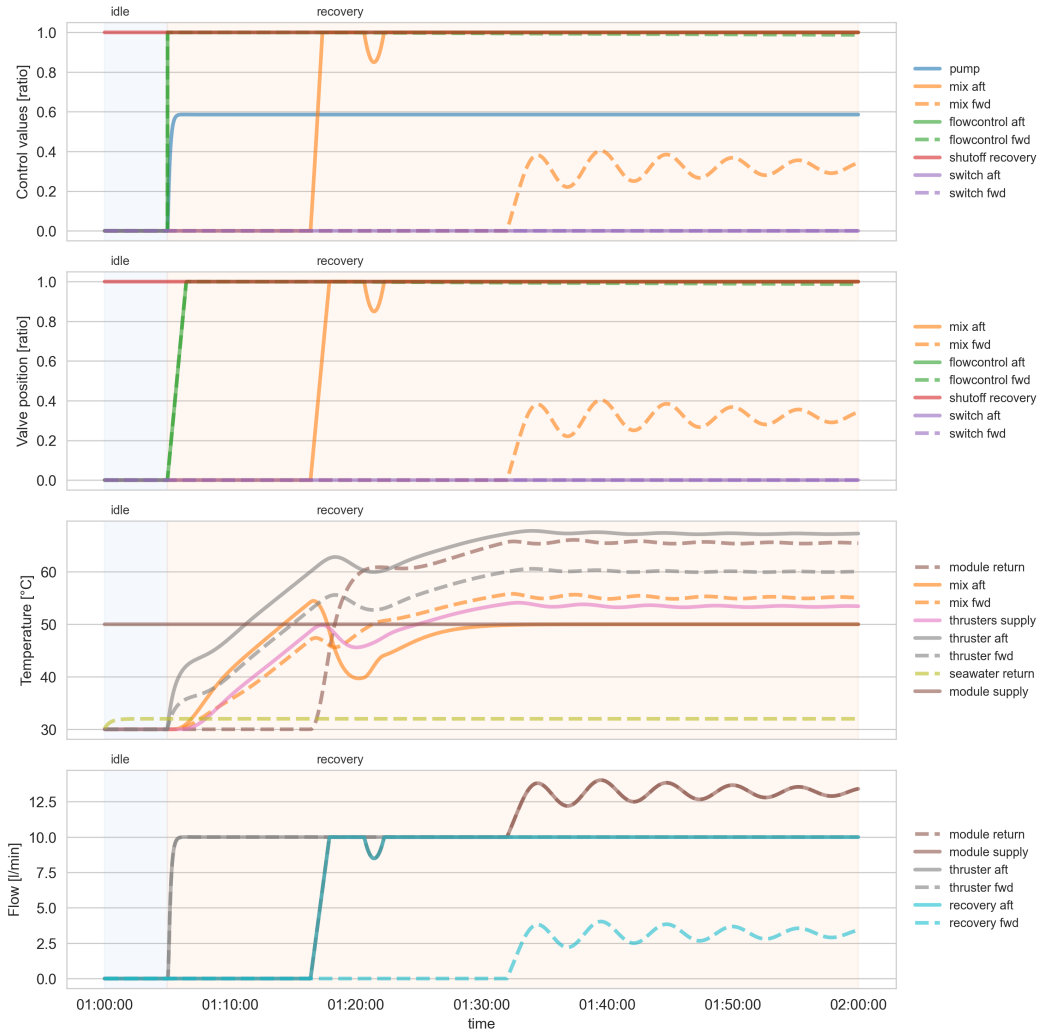


Figure 6. First hour of simulation in recovery mode, showing initialization and mixing control of the aft thruster.

indicates that some recirculation is required to maintain the desired temperature, and results in the *thrusters supply* temperature stabilizing slightly above the *module supply* temperature of 50 °C.

5.3 Mode Switching

Figure 7 depicts the second half of the simulation. At 02:00, the forward thruster is deactivated. In response, the controller shuts off flow to the forward thruster by closing *flow control aft*, and adjusts the *pump* duty to maintain the aft flow setpoint.

At 02:30, the *module supply* temperature is raised from 50 °C to 80 °C. This leads to the *thrusters supply* temperature to pass the 70 °C limit, triggering the cooling mode. This switches the valves *switch aft*, *switch fwd* and *shutoff recovery*. Note the difference between the control signal and the actual movement of the valves. In cooling mode, the *seawater return* temperature increases as heat is expelled through the heat exchanger, while the *thruster aft* temperature decreases as the control will maintain the cooling setpoint.

6 Summary and Outlook

This paper introduces a workflow to use MODELICA in the control system design and development process of a zero emission yacht's thermal system. It is shown that a MODELICA model of the thermal system can be compiled into a FMU, which is then embedded to a python wrapper to enable communication via MQTT to the control layer.

This enables the design of a control system even though the real system has not yet been finalized. It is also possible to assess scenarios that are quite difficult to evaluate in the real world. When the project has a tight schedule, the time taken to implement the control parameters of the real system can be reduced. An assessment of the time and cost savings by using this method may also be of interest.

For this project, the creation of the remaining submodules and their assembly into the whole system is still pending. Once the yacht has been constructed, it will be possible to evaluate the quality of the parameters

DOI
10.3384/ecp218801

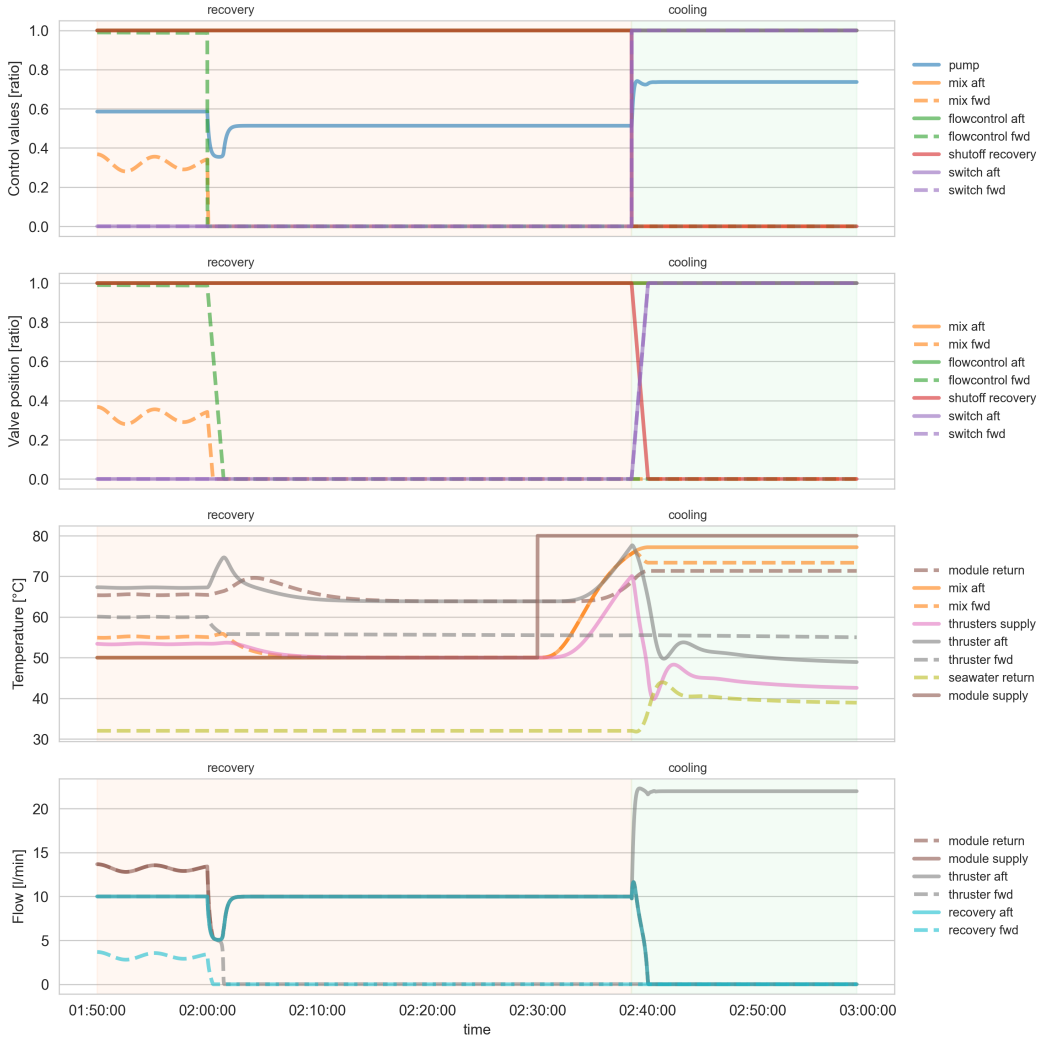


Figure 7. Continuation of the simulation shown in Figure 6, including shutdown of the forward thruster at 02:00 and supply temperature increase at 02:30.

determined by the approach this paper has shown.

Acknowledgements

The authors acknowledge FOUNDATION ZERO for the possibility of sharing the results in this paper.

References

Alexander Neumann (2025). *pytransitions Version Release 0.9.3*. URL: https://github.com/pytransitions/transitions.

Banks, Andrew et al. (2019-03). *MQTT Version 5.0*. Tech. rep. OASIS. URL: https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html.

Brunnemann, Johannes (2020). *NAKULEK - Naturumlaufkühlung für Leistungselektronik : Schlussbericht XRG Simulation GmbH : Laufzeit: 01.07.2016-30.06.2020*. Hamburg. DOI: 10.2314/KXP:1755577478. URL: https://www.tib.eu/de/suchen/id/TIBKAT%3A1755577478.

Clarke, Daniel et al. (2023). "CO2 emissions from global shipping: A new experimental database". In: *OECD Statistics Working Papers* 2023/04. DOI: 10.1787/bc2f7599-en.

Colvin, Samuel et al. (2025-04). *Pydantic*. Version v2.11.4. URL: https://github.com/pydantic/pydantic.

Dassault Systèmes (2025). *FMPy Version 0.3.23*. URL: https://github.com/CATIA-Systems/FMPy.

Foundation Zero (2024a). *An exploration into making PVT panels suitable for use at sea*. URL: https://www.foundationzero.org/insights/pvt-panels.

Foundation Zero (2024b). *The Foundation^0 Power Hub: A sustainable energy source for the future*. URL: https://www.foundationzero.org/insights/power-hub.

Foundation Zero (2024c). *Uncovering the results of lightweight insulation*. URL: https://www.foundationzero.org/insights/insulation.

Foundation Zero (2025a). *How mast ventilation can reduce energy expenditure in the maritime industry*. URL: https://www.foundationzero.org/insights/mast-ventilation.

Foundation Zero (2025b). *Power Hub*. URL: https://github.com/foundation-zero/power-hub.

Hesselgreaves, John E. (2001). *Compact Heat Exchangers*. Oxford: Pergamon. ISBN: 978-0-08-042839-0. DOI: 10.1016/B978-0-08-042839-0.X5000-9. URL: https://www.

sciencedirect . com / book / 9780080428390 / compact - heat - exchangers.

International Energy Agency (2023). “Tracking Clean Energy Progress 2023”. In: URL: https : / / www . iea . org / reports / tracking-clean-energy-progress-2023.

Leslie-Miller, Mark and Bob van Someren (2022-11). “Hydro generation, solar, electric and heat generation”. In: 27th International HISWA symposium. URL: https : / / www . foundationzero.org/insights/the-boat.

Modelica Association (2023-03). *Modelica – A Unified Object-Oriented Language for Systems Modeling. Language Specification Version 3.6*. Tech. rep. Linköping: Modelica Association. URL: https://specification.modelica.org/maint/3.6/MLS.html.

Thiele, Bernhard et al. (2017-05). “Towards a Standard-Conform, Platform-Generic and Feature-Rich Modelica Device Drivers Library”. In: *Proceedings of the 12th International Modelica Conference*. Ed. by Jiří Kofránek and Francesco Casella. Prague, Czech Republic, pp. 713–723. DOI: 10.3384/ecp17132713.

XRG Simulation GmbH (2025-03). *HVAC Library v.3.4.0*. Tech. rep. Hamburg: XRG Simulation GmbH. URL: https : / / xrg - simulation.de/seiten/hvac-library.

Race Car Braking System Thermal Model for Real Time Use in a Driving Simulator

Massimo Stellato[1] Alberto Momesso[2] Theodor Ensbury[3]
Alessandro Picarelli[4]

[1]Dallara Automobili, Italy, m.stellato@dallara.it
[2]Dallara Automobili, Italy, a.momesso@dallara.it
[3]Claytex USA, Canada, theodor.ensbury@claytex.com
[4]Claytex, United Kingdom, alessandro.picarelli@claytex.com

Abstract

Vehicle performance optimisation is the primary target in the development of race cars and road hypercars. Considering the complexity of modern vehicles, a holistic approach to analyse the interaction of vehicle dynamics, powertrain cooling systems, brake cooling, and human drivers in the same simulation can be crucial to maximise the overall performance (*Bouvy et al., 2012*). This paper is a follow up of a previous article written by Dallara and Claytex *("Race Car Cooling System Model for Real Time use in a Driving Simulator", Stellato et al., 2023*), which describes a one-dimensional (1D) cooling system model integrated with a vehicle multibody model, used in the Dallara dynamic driving simulator (Figure 1). This collaboration has led to a further step, where Dallara has developed a model to optimise the brake cooling of its vehicles, with Claytex's VeSyMA suite used for the auxiliary vehicle systems and code compilation. The model was validated through the comparison with real-vehicle data, demonstrating good accuracy in i.) sizing a race car braking system; and ii.) providing a refined assessment of the global vehicle performance on the driving simulator.

Keywords: brakes cooling, cooling systems, real-time, driving simulator

1 Introduction

Optimising vehicle performance is a primary objective in the development of high-performance vehicles. A holistic simulation approach that integrates key component models, such as powertrain and brake cooling systems, with vehicle dynamics and driver behaviour, is essential for maximising overall performance. With this regard, in a previous paper *(Stellato et al., 2023),* a car cooling system model was integrated inside the vehicle multi-body model. This integration allowed the definition of the best cooling system architecture for every project, targeting the trade-off to maximise the vehicle performance, especially in a professional driving simulator with a human driver. As a further step, the previous model was extended with the braking system thermal model, to predict the braking system's main components' temperatures (disc and caliper) and neighbouring components (disc bells and rims). As for the cooling system, this integration allows the evaluation of several layouts for air intakes, brake ducts, disc, and caliper size in different boundary conditions, finding the best configuration for aerodynamic, handling, and cooling targets, based on the vehicle mission and the driving style.

In the first design phase, the simulation model may run offline, using vehicle speed and brake pressure profiles coming from:

- Track data acquisition, if the activity involves an existing vehicle update, and thus, there is already data available.
- Quasi-static lap time simulations, in the case of designing a new vehicle.

Then, in a more advanced phase, it can be integrated directly into the Dallara dynamic driving simulator to obtain more accurate and precise feedback from the driver.

In the remainder, this article is organised as follows: Section II presents the Claytex VeSyMA suite utilised in the project; Section III delves into the proposed models; Section IV discusses the main results; and Section V summarises the main conclusions.



Figure 1: Dallara dynamic driving simulator

2 Claytex VeSyMA suite

The VeSyMA (**Ve**hicle **Sy**stems **M**odelling and **A**nalysis) simulation suite was used to form the backbone of the Dallara vehicle model. Comprising various subject-specific Modelica libraries built on top of a common base, the VeSyMA suite was originally developed by Claytex using Dymola and Modelica. More than 25 years of experience in the motorsport world supporting customers from Formula 1, IndyCar and NASCAR with Dymola and Modelica simulations was drawn upon in the creation of VeSyMA (*Hammond-Scott and Dempsey, 2018*).

Conceived and built in a hierarchical structure, all subject-specific libraries (such as Suspensions for Vehicle Dynamics analysis, Motorsport for motorsport-specific requirements, PTDynamics for the simulation of drive trains and gearboxes, etc.) inherit from the base VeSyMA library. This fully leverages the concepts of model reuse and inheritance present in the object-oriented Modelica language, helping users to efficiently manage their own simulation libraries.

The cornerstone of model reuse in the VeSyMA suite is the utilisation of the open-source Vehicle Interfaces library. All vehicle models found within VeSyMA are built from templates located in the VeSyMA library, which are built up of component templates from Vehicle Interfaces. Integrating third-party libraries built on the same open-source Vehicle Interfaces library into VeSyMA base vehicle models is thus a simplified task. Such a feature is of use in a study such as this, as the implementation of customer-developed libraries into VeSyMA vehicle models is straightforward.

Beyond templates, the VeSyMA suite contains packages of commonly found vehicle components, with simplified versions found in the VeSyMA library, and detailed versions found within the subject-specific libraries. Scalability of the detail in a model is therefore possible by selecting which component is used, which is of benefit as it enables vehicle models to be configured to focus on the detail where required. This can be done both for intellectual clarity and efficient use of computational resources. All component and vehicle models found in the VeSyMA suite have been built from the outset with performance in mind, with optimisations made in key areas such as state selection.

3 The Model

Figure 2 shows the cooling and braking system models connected to the vehicle model and the human driver. The interface with the multi-body vehicle model allows further dynamic analysis to improve vehicle performance.

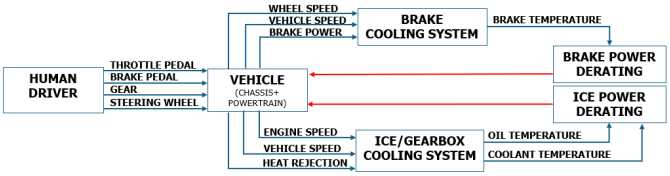


Figure 2: Cooling and braking system interactions with vehicle model and human driver

Regarding the cooling system model, the coolant and oil temperatures are defined as model outputs. Therefore, they affect the internal combustion engine (ICE) power curve according to the derating strategies agreed with the engine supplier. Instead, for the braking system model, the outputs are the disc and caliper temperatures, which will affect the friction coefficient inside the vehicle model. A temperature increase will decrease the coefficient, forcing the driver to increase the braking distance.

3.1 Architecture and Physics

Figure 3 shows the components of the braking system model and describes how the heat is propagated among them.

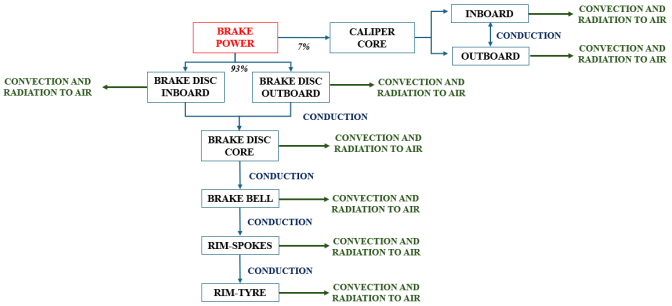


Figure 3: Heat fluxes in the simulation model

The first block is the "Brake Power", where the brake power, and thus the overall system heat rejection, is calculated by the following equations at each time step:

$$P_B = \mu F_n R \omega \quad (1)$$

$$R = \frac{R_e + R_i}{2} \quad (2)$$

$$F_n = p A \quad (3)$$

P_B = brake power [W]
μ = friction coefficient [-]
F_n = disc normal force [N]
R = disc mean radius [m]
ω = wheel angular speed [rad/s]
R_e =disc external radius [m]
R_i =disc internal radius [m]
p = brake pressure [Pa]
A = brake pistons area [m^2]

DOI 10.3384/ecp218811

A 7% fraction of the heat rejection produced is assigned to the caliper core. This fraction is exchanged to the inboard and outboard areas through conduction and finally dissipated by convection into the environment. This percentage was estimated after several validation processes on different types of Dallara vehicles (race cars, hypercars, road cars). To align the track data with the simulation model, the values fell in the range of 6.5% to 7.5%. This matches well with both the standard estimation of 5-10% and the values reported in recent studies (*Vdovin and Sebben, 2018*). As a further step in the model, the implementation of a disc rotational model in contact with the pads could be considered, where this percentage will be given as output of the boundary conditions.

The remaining 93% of heat rejection is assigned to the inboard and outboard brake disc area, equally distributed. The heat is then conducted to the disc core, and thereafter, it passes through the brake bell, rim-spokes and rim-tyre. All these components dissipate the heat by convection and radiation into the environment.

To evaluate the temperature of each component, the first principle of thermodynamics is applied. First, the heat transferred to the component is calculated. This is the instantaneous difference between the heat rejection (HR), produced by the braking system or exchanged from the neighbouring component by conduction, and the heat dissipated by convection to air (HD):

$$HF = HR - HD \tag{4}$$

HF = overall heat flux [W]
HR = heat rejection [W]
HD = heat dissipation [W]

At this point, the component temperature can be calculated at each time step by:

$$T_{t1} = T_{t0} + \frac{HF\ (t_1 - t_0)}{C} \tag{5}$$

T_{t1} = temperature at time t_1 [K]
T_{t0} = temperature at time t_0 [K]
C = thermal capacity [J/K]
t = time [s]

The thermal conduction is evaluated by the following equation:

$$\dot{Q}_{cond} = K\ \frac{A}{d}\ \Delta T \tag{6}$$

$\dot{Q}_{cond}$ = thermal conduction [W]
K = conductivity coefficient [W/mK]
A = component reference area [m^2]
d = component reference length [m]
ΔT = delta temperature [K]

While the thermal convection is given by:

$$\dot{Q}_{conv} = HTC\ (T - T_{air}) \tag{7}$$

$\dot{Q}_{conv}$ = thermal convection [W]
HTC = heat transfer coefficient [W/K]
T = component temperature [K]
T_{air} = air temperature [K]

3.2 Parameters

The model input parameters are obtained from computational fluid dynamics simulations (CFD), experimental tests, and CAD data (computer-aided design). In detail:

- The heat transfer coefficient (HTC) for each component (disc, caliper, brake bell, rim-spokes and rim-tyre) is the output of CFD simulations that consider both convection and radiation. The HTC is a function of vehicle speed and component heat rejection: for each component, 9 different HTCs are evaluated at 3 different speeds (100, 180, and 300 km/h) and 3 heat rejection rates (minimum, average, and maximum). As an example, Table 1 illustrates an HTC map for one component. The instantaneous value in the 1D simulation is then extrapolated or interpolated from these tables. Figure 4 shows the outputs of the CFD simulations, where the variability of HTCs can be appreciated: for the inboard disc surface, close to the air duct, the values are practically doubled with respect to those of the outboard.

HTC [W/K]		Heat Rejection [kW]		
		2.4	7.2	12
Vehicle speed [km/h]	100	7.52	6.96	7.2
	180	11.6	10.56	10.16
	300	16.72	15.52	14.56

Table 1: HTC map for 3 different heat rejections and vehicle speeds

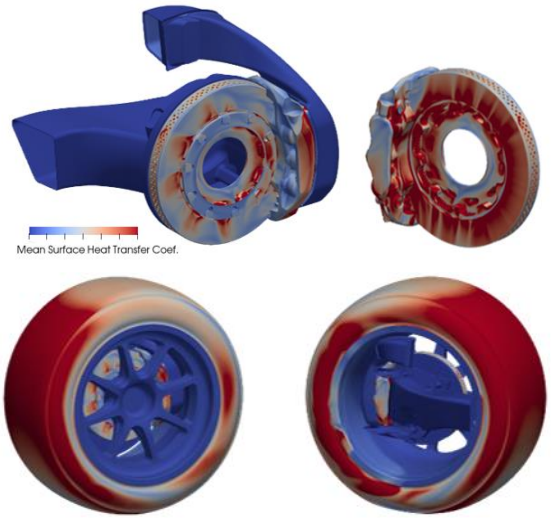


Figure 4: HTCs values from a CFD simulation

- The brake friction coefficient is provided by the manufacturer and is described in a table as a function of surface temperature and brake pressure. Figure 5 reports an example: it has to be reported that the unit of measurement of the x-axis is [bar·rad/s], as the map takes into account also the disc angular speed.

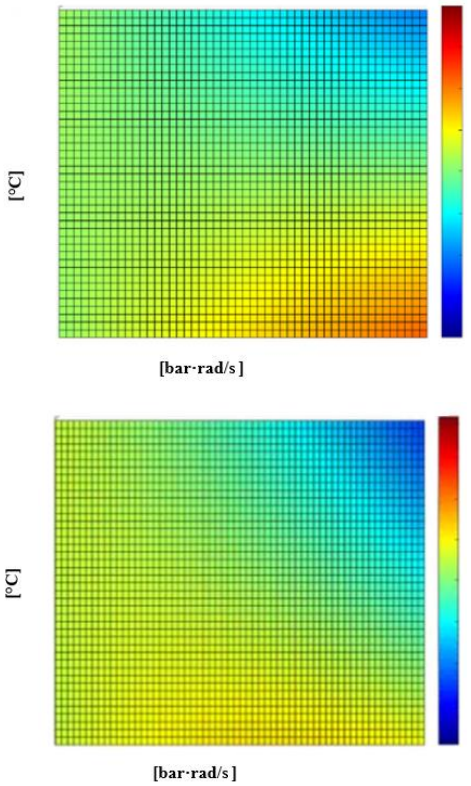


Figure 5: Brake friction coefficient maps for front brakes (top) and rear ones (bottom)

- The component's physical properties, such as density, emissivity, conductivity, and specific heat, are obtained from the supplier's datasheets. The evaluation of the reference area and length for the calculation of the heat exchanged by thermal conduction is done by direct analysis of CAD data. The main parameters for each component are summarised in Table 2. An interesting note concerns the rim. To have a better discretisation, it has been divided into two parts: rim-spokes and rim-tyre. The first part considers the spokes and the area close to the brake bell. The other one is the part close to the tyre (Figure 6). Each part has its own HTCs.



Figure 6: Rim-tyre part

CALIPER		BRAKE BELL	
Core Mass	[kg]	Mass	[kg]
Inboard/Outboard Mass	[kg]	Heat Coefficient	[J/kgK]
Heat Coefficient	[J/kgK]	Initial Temperature	[K]
Initial Temperature	[K]	Material	
Material		Conductivity	[W/(mK)]
Conductivity	[W/(mK)]	Density	[kg/m³]
Density	[kg/m³]	Emissivity	[-]
Emissivity	[-]	Reference Area	[m²]
Reference Area	[m²]	Reference Length	[m]
Reference Length	[m]	RIM-SPOKES	
Piston Area	[m²]	Mass	[kg]
DISC		Heat Coefficient	[J/kgK]
Core Mass	[kg]	Initial Temperature	[K]
Inboard/Outboard Mass	[kg]	Material	
Disc Heat Coefficient	[J/kgK]	Conductivity	[W/(mK)]
Surface Initial Temperature	[K]	Density	[kg/m³]
Core Initial Temperature	[K]	Emissivity	[-]
Material		Reference Area	[m²]
Density	[kg/m³]	Reference Length	[m]
Emissivity	[-]	RIM-TYRE	
Conductivity	[W/(mK)]	Mass	[kg]
Reference Area	[m²]	Heat Coefficient	[J/kgK]
Reference Length	[m]	Initial Temperature	[K]
External Radius	[m]	Material	
Internal Radius	[m]	Conductivity	[W/(mK)]
Thickness	[m]	Density	[kg/m³]
		Emissivity	[-]
		Reference Area	[m²]
		Reference Length	[m]

Table 2: Components geometrical and physical properties

Overall, if the model is executed offline, e.g., outside the Dallara simulator, it requires input data for the brake pressure at each caliper, the vehicle speed, and the angular velocity profiles of each wheel during a representative track lap. Otherwise, if deployed online in the Dallara dynamic driving simulator, the human driver and the vehicle model will directly provide these inputs.

3.3 Real-time Features

In terms of real-time setup, the proposed model follows the same methodology as the previous paper on this subject (*Stellato et al., 2023*). Dymola's fixed-step Euler solver was employed, running at a sampling time of 1 ms. Preliminary testing of the model was undertaken on a 2.8 GHz I7 processor SSD-equipped desktop workstation; successful real-time simulation on this machine ensured the model would be suitable when deployed on the more powerful hardware associated with Dallara's DiL simulator.

The VeSyMA – Driver-in-the-Loop library was utilised for the compilation of the vehicle and cooling models into a real-time app capable of running on the Dallara simulator. A 2nd order Runge-Kutta explicit solver method was used for the real-time application, with advanced inline settings (*Dymola manual*) contributing to a robust and high-performance real-time model (*Hammond-Scott and Dempsey, 2018*).

DOI 10.3384/ecp218811

Regarding the brake thermal model in real-time use, some first-order filters (with a 5 ms time constant) were added when solving non-linear equations.

4 Virtual Validation

4.1 Offline Simulation

For any simulation activity, the validation phase represents the crucial moment to assess the correctness of the model. For this work, it consisted of the following steps:

- Implementation of the braking system architecture of an existing vehicle.
- Offline simulation using measured vehicle speed, brake pressure and ambient temperature as input data.
- Comparison between the simulated temperature profiles and the measured ones on the track.

Figure 7 shows the comparison between the simulated disc temperature profile and the real logged data of a hypercar with more than 1000 horsepower at Vallelunga track. The average difference between the measured temperature and the simulated one is ~12°C, with a root mean square error (RMSE) of ~15 °C, and a mean relative error of ~2%. This level of accuracy, given the high temperature range profile (from 400 to 780 °C), is considered acceptable for sizing the braking system and analysing its performance. Similarly, Figure 8 shows the comparison for the caliper temperature profile. In this case, the average temperature difference between the measured and the simulated one is ~3.3°C, the RMSE is ~3.9 °C and the mean relative error is ~2.7%.

Overall, the accuracy of the model is considered acceptable for sizing a race car braking system and doing a refined assessment of the global vehicle performance on the driving simulator.

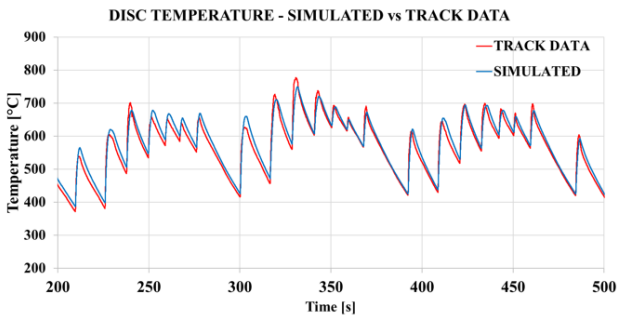


Figure 7: Simulated disc temperature vs real logged data

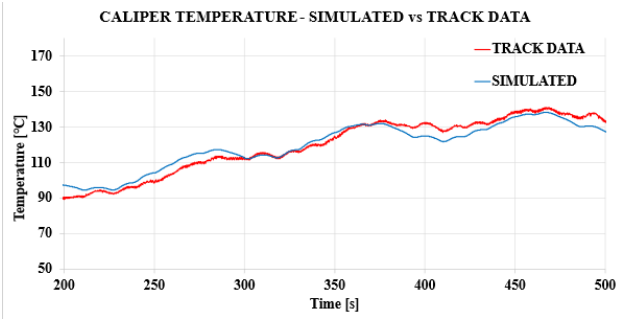


Figure 8: Simulated caliper temperature vs real logged data

4.2 Real-time Simulation in the Dallara Dynamic Driving Simulator

The same model was also deployed directly on the Dallara Driving Simulator, where the human driver performed a series of laps at maximum vehicle performance on the Zandvoort track. From this experimental campaign, the main target is the evaluation of:

- Disc temperature before the braking point. This is a value assessed by the supplier to preserve the disc and avoid excessive wear of the component.
- Friction coefficient reduction (due to the increased disc temperature), and thus the brake power.

The following figures show the results of a simulator session, where the driver performed a run (a total of 8 laps) at maximum performance to test the braking consistency and effectiveness. In Figure 9, the front left (FL) and front right (FR) brake disc outboard temperatures are plotted: after 400 seconds, the peak temperature stabilises between 900 and 1000 °C.

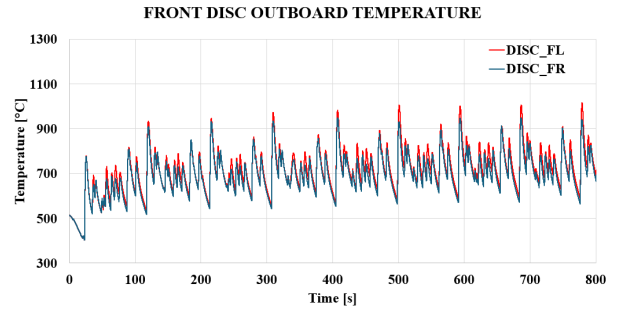


Figure 9: Front discs outboard temperature in a driving simulator session

It can be noted that the left disc is warmer than the right one in the heaviest braking zone. This is attributable to the different brake pressure values for these turns (Figure 10), caused by the ABS (anti-lock braking system). However,

from the driver's point of view, this temperature imbalance did not affect the performance.

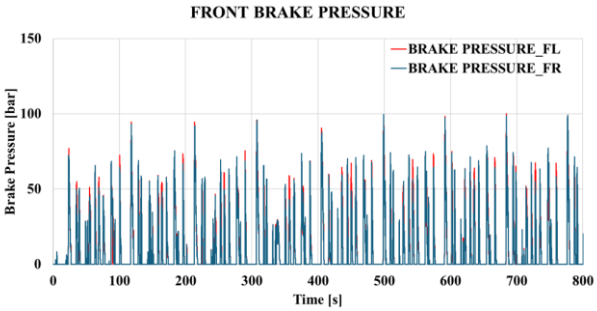


Figure 10: Front brakes pressure in a driving simulator session

The most interesting part turns out to be the friction coefficient trend. As mentioned in the previous paragraph, inside the model, it is discretised as a look-up table, function of brake pressure and disc temperature. Figure 11 shows the actual friction coefficient during the whole run. As expected, increased temperature leads to overall friction degradation, therefore, a reduced braking performance. In the track's hardest braking zone, the disc temperature reaches 1000°C, and the coefficient is reduced by more than 20% from its optimum value.

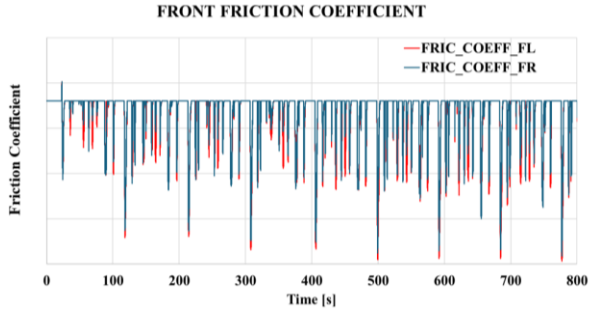


Figure 11: Front brakes friction coefficient in a driving simulator session

The same analysis can also be done for the rear brakes, where the disc outboard peak temperature is approximately 350 °C lower than the front ones (Figure 12). Also, in this case, the left side is slightly warmer than the right one, but with a smaller gap than before.

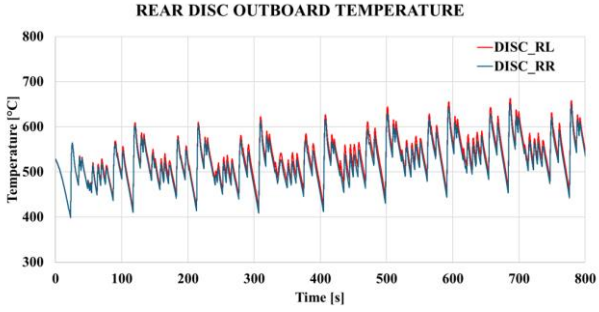


Figure 12: Rear discs outboard temperature in a driving simulator session

However, the mean lower temperature results in a different friction coefficient trend. If for the front brakes, its value varies over the lap course, then for the rear ones it is practically constant, with only two turns leading to a reduction of about 10 % (Figure 13).

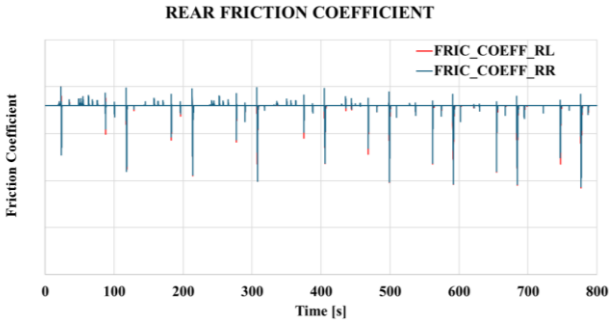


Figure 13: Rear brakes friction coefficient in a driving simulator session

The driving simulator and this simulation model can also be useful in preparing for race weekend by evaluating the ideal air duct size, if necessary. Several blanking percentages for both front and rear brakes can be tested, accounting for different ambient temperatures and race conditions (qualifying, traffic, free air, etc.). So, based on the situations they face, track engineers have a precise assessment of the necessary blanking, which ensures the best trade-off between performance and reliability.

5 Conclusions and Further Developments

The proposed simulation model, as developed and validated, is currently used at Dallara to size the braking system and analyse the vehicle performance on the driving simulator for different tracks, driving styles and boundary conditions (brake size, materials, aero configurations, ambient temperature). As all these parameters affect the overall results, the simulation model is useful to evaluate each effect to develop the best braking system architecture for every project, targeting the maximisation of vehicle performance (*Stellato et al., 2023).* The duty cycle analysed for the case study is a track lap at maximum performance, but additional critical driving cycles can be studied on the simulator, such as the brake warm-up during the out lap. Certainly, a key aspect turns out to be the friction coefficient sensitivity. The temperature calculation allows the thermal effect to be considered, defining the ideal design for the cooling ducts in each track. However, the variable friction coefficient makes it possible to evaluate the impact on lap time and vehicle performance. For example, in some tracks, the air duct size could be decreased to reduce the aerodynamic drag. Consequently, the temperature will increase, and the friction coefficient will decrease, resulting in worse

DOI 10.3384/ecp218811

braking performance. However, thanks to the improved aerodynamics, the lap time will improve. This scenario represents the typical trade-off that is analysed by the simulation model.

Further refinement and improvements are planned for the simulation model. Currently, the thermal discretisation stops at the rim. The next step will be to predict the thermal conduction from the rim to the tyre and integrate the tyre thermal model. This latter feature will further expand the holistic approach and enable a greater optimisation of vehicle performance by considering the temperature effect on the tyre's lateral and longitudinal grip and its correlation to the brake thermal performance.

References

C. Bouvy, P. Jeck, J. Gissing, T. Lichius, L. Ecksterin. "Holistic Vehicle Simulation using Modelica – An Application on Thermal Management and Operation Strategy for Electrified Vehicles". *Proceedings of 9th International Modelica Conference, pp. 263-270, 2012.*

H. Hammond-Scot, M. Dempsey. Vehicle Systems Modelling and Analysis (VeSyMA) Platform. *Proceedings of 2nd Japanese Modelica Conference, pp. 61-65, 2018.*

M. Stellato, L. Bergianti, A. Picarelli. "Race Car Cooling System Model for Real Time use in a Driving Simulator". *Proceedings of 15th International Modelica Conference, 2023.*

M. Stellato, L. Bergianti, J. Batteh. "Powertrain and thermal system simulation model of a high-performance electric road vehicle". *Proceeding of 12th International Modelica Conference, pp. 171-180, 2017.*

A. Vdovin, S. Sebben. "A coupled approach for vehicle brake cooling performance simulations". *International Journal of Thermal Sciences, June 2018.*

Dassault Systèmes. Dymola Manual.

DOI
10.3384/ecp218811

Development of a Multi-Physical Simulation Platform for Durability Prediction for Hyundai & Kia Electric Vehicles

Jaewung Jung[1] Alessandro Picarelli[2] David Briant[2]
Kadir Sahin[2] Garron Fish[2] LEBRUN Victor-Marie[3]
Christopher STROMBERGER[3] COLLEONI Arnaud[3]
PRIETO Quentin[3]

[1]Hyundai Motor Company, South Korea, JAEWUNGJUNG@hyundai.com
[2]Claytex Services Ltd., United Kingdom,
{alessandro.picarelli, david.briant, kadir.sahin,
garron.fish}claytex.com
[3]Dassault Systems, France / Germany, {Victor-Marie.LEBRUN,
christopher.stromberger, arnaud.colleoni,
quentin.prieto}3ds.com

Abstract

One of the most popular ways of measuring signals for various components in vehicle on durability tracks will be Road Load Data Acquisition (RLDA) which gains load data through accelerated customer usages tests on either proving ground or field, using an instrumented vehicle. However, under the current circumstances, not only does a vehicle development cycle get shorter but also new mobility concepts keep being developed, a methodology is needed to cope with overall durability assessment in advance of a physical prototype being made for testing. In this study, this virtual testing methodology will be called Virtual RLDA.

Keywords: Durability, Multi-Physics, Vehicle Simulation

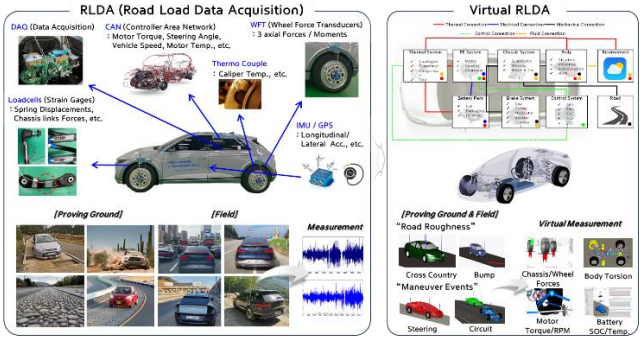


Figure 1. Idea of Virtual RLDA from Conventional Measurement Campaign

In this study, one of the most well-known electric vehicles in market is from Hyundai, with 2 in-wheel motors in rear which was chosen as the target vehicle. First, 6 major sub-systems such as vehicle controller, suspension, hydraulic brake, motor, battery and thermal management are modelled and validated separately, then they are integrated into one large vehicle model as a part of establishing the Virtual RLDA Platform which will be used for overall vehicle level durability development in early stage in Hyundai Kia Motor Company (HKMC).

1 Introduction

1.1 Background of the project

From the overall durability point of view, an electric vehicle still requires a number of studies to figure out unknown failures and degradation mechanisms especially on the motor, inverter and battery which has different failure mechanisms from conventional ICE vehicle. If the durability failures are divided into two large aspects, one aspect will be structural failure and the other will be powertrain related failure. The objective of this study is to develop the vehicle model that can cover both structural and powertrain related failures depending on the choice of the level of fidelity.

First, when it comes to structural durability of an EV, a high voltage battery system weighs about 300kg more than a fuel tank in ICE vehicle. Also, with regards to in-wheel motor systems, the motor system is integrated in each wheel assembly which have much heavier unsprung mass than conventional wheel. This weight increase has a significant influence on the amount of damage to the suspension system and each body mountings due to the increased reaction forces.

Moreover, powertrain coupled components such as subframes, motor mountings and driveshafts are also expected to have more damage due to the reaction torque from a motor as an electric motor can theoretically generate maximum torque at low speed. For this reason, a multi-physics model which can efficiently simulate a motor's movement aligned with powertrain coupled

suspension is essential to replicate the coupled reaction using both table-based electrical motor model and powertrain dynamics-based suspension in a single simulation environment.

As for powertrain durability, the heat generated from motor during driving is the key factor for the degradation of powertrain systems. Also, since the hydraulic brake assembly is integrated within the in-wheel motor system, most of the brake heat during braking is directly radiated to the motor system. For this reason, there should be a realistic method to simulate the thermal interface between hydraulic brake and motor system. In order to simulate the above phenomenon in the model, the first thing we need is thermal management system, modelled with TiL Library, which controls temperatures for both coolant and refrigerant to be maintained in a proper level for most driving scenarios. The second thing we need is the vehicle controller model which simulates pure hydraulic brake pressure that is used as an input signal from thermal brake model by subtracting regenerative motor torque from the total required brake torque in a vehicle. This is one of the reasons why the vehicle controller is required in the vehicle model.

The virtual RLDA platform which is developed in this study to simulate various perspectives of durability influences in vehicle level consists of the following systems below:

- Vehicle Controller.
- Hydraulic Brake (Mechanical / Thermal)
- Vehicle Dynamics (Ideal & Bushed Suspension)
- Motor (Electrical / Thermal for Standard & IWM)
- Battery (ECM / Thermal Cell & Pack)
- Thermal Management (Entire Vehicle Circuit)
- Integrated Vehicle Model (Assembly)
- Durability Scenarios (Tire / Driver / Road Models)

1.2 Motivation of Using Modelica

The vehicle level durability simulation environment based on Modelica libraries has distinctive advantages as follows.

First, each system in a vehicle behaves based on their own physics. It would take numerous efforts to model each different domain systems from scratch as it requires each domain-specific expertise for the modelling. However, the eco-system of, Modelica provides each system domain-specific libraries which gives a lot of potential and flexibility to modelling engineers.

Also, the whole vehicle level simulation which exchanges multi-physical signals between different systems in one single simulation environment is possible in Modelica. In this study, 3 different libraries from Dassault Systèmes (battery, motor), VeSyMA (vehicle template, suspension, brake, driver, road, etc.), TiL (thermal management) are integrated to simulate one large vehicle which is the most efficient way for multi-physical simulation.

Lastly, it provides very efficient way of replacing one model to another from component to large-scale level. From the perspective of managing models in different fidelity and types, this makes the modelling process even more scalable and efficient.

The figure below is the model management strategy of the developed models in this study depending on purpose of durability scenarios.

Fidelity Level: Basic Fidelity / High Fidelity — Simulation Output: Category 1 + 2 / Category 2 Only

No.	System Models		Category 1 Proving Ground (Suspension) Road Roughness	Category 1 Proving Ground (Suspension) Maneuvering	Category 2 Proving Ground (Powertrain)	Category 2 Field	Category 2 Circuit	Simulation Outputs
1	Suspension		Bushed or Bushed Subframe		Table-based or Ideal			[Spring/Links/Bush] Force / Displacement
2	Brake		Mechanical		Thermal			[Brake] Torque, Disc Temp.
3	Driveline		Base or Joint type (Drive Shaft & Differential)					[Drive Shaft] Torque
4	Powertrain (Motor)		Electric		Electric + Thermal			[Motor] Torque / RPM [Coil & Magnet] Temp., Heat Loss
5	Battery	Cell	None or Electric		Electric + Thermal			[Cell] Voltage, SOC, Temp.
		Pack			Scaled			[Cell] Temp. in the Pack [Coolant (PE/Battery)] Temperature
6	Cooling				PE + Battery + Heat Pump Loop			
7	Tire		FTire		Pacejka			[Wheel Center] Forces / Moments
8	Driver		Longitudinal	Longitudinal + Lateral	Longitudinal	Longitudinal + Lateral		-
9	Vehicle Controller		2WD or AWD					Motor Torques, Brake Pressures
10	Simulation		Rig (Half / Full-car) or Closed-Loop		Closed-Loop			-
Representative Events			Pothole	Steering, Go & Stop	UDDS, ColdFTP	Ex) City road in Seoul	Nurbergring	-

Figure 2. Overall Vehicle Model Fidelities for Different Durability Scenarios

2 System Modelling

2.1 Vehicle Controller

The main function of the vehicle controller in the model is to decide driving torque, regenerative torque, and Hydraulic brake torque relative to the driving conditions. It also decides the clutch engagement and disengagement timing between front and rear motor of all-wheel drive (AWD) vehicle models. Figure 3 shows the vehicle controller model built with Dymola for HMC's AWD electric vehicle.

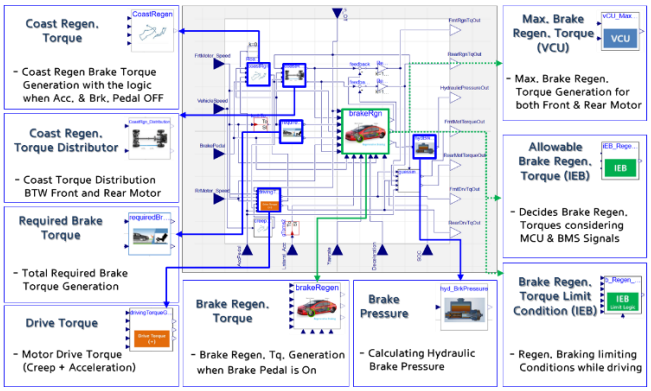


Figure 3. Vehicle Controller Model built for Electric AWD Durability Scenarios

DOI 10.3384/ecp218819

One of the main roles of the vehicle controller is to generate drive torque using both maximum torque map relative to motor speed and tuned torque map in the development stage which has different torque levels in different driving modes and regenerative brake levels dependant on accelerator pedal stroke. For this reason, vehicles can generate different levels of driving torque even with the same vehicle speed and accelerator pedal stroke dependent on different driving modes. The model also generates creep torque while driving using a tuned map when the accelerator pedal is not pressed.

The total brake torque can be divided into regenerative brake torque and hydraulic brake torque. The total brake torque is decided by the table-based tuned map relative to vehicle speed and brake pedal stroke. When regenerative brake torque is calculated considering the status of clutch engagement between front and rear motor based on the logic, the pure hydraulic brake torque can be lastly calculated.

Regenerative brake torque of a motor can be also divided into brake regenerative torque, which occurs when brake pedal is pressed, and coast regenerative torque, which is generated when normally brake pedal is off. Especially for brake regenerative torque, it is mainly decided by the coordinated control between vehicle controller unit (VCU) and integrated electronic brake (IEB). Therefore, the model includes the logics of interaction between different Electronic Control Units (ECUs) to decide the brake regenerative torque in sequence as it happens in real vehicle.

Hydraulic brake torque is obtained from the remaining torque between total brake torque and regenerative brake torque. The hydraulic brake model that will be discussed in the section 2.2 takes hydraulic master cylinder pressure as an input to calculate brake torque which generates disc temperatures. For this reason, additional converter which converts brake torque to pressure is also added to the model.

To validate the developed vehicle controller model, the data measured in one of the highest mountains in Korea, considered as the most severe driving conditions with frequent braking and steep steering, is used for the controller validation in Figure 4.

As it can be seen in the data, the vehicle controller generates driving torques, regenerative torques and hydraulic pressure quite accurately compared to the measurement even with the severe driving condition. There is a bit of inaccuracy in brake pressure around 850s, it is due to in accurate total brake torque from the tuned map as the measurement kept changing its driving and regenerative mode.

Comparing several values with measurement, this vehicle controller model still needs some improvements, primarily the clutch logic between front and rear motor and also for the regenerative brake logic. For this reason, the vehicle controller model will be updated. However, it still shows good prediction if it is not a very special driving case.

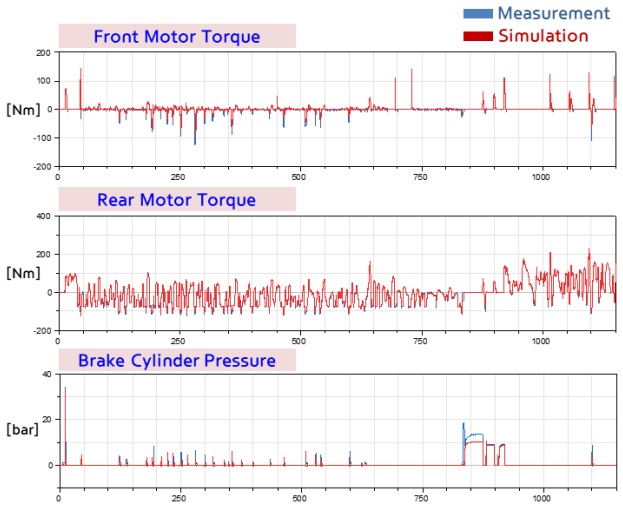


Figure 4. Validation of Motor Torque and Brake Pressure with measurement

2.2 Hydraulic Brake

The Hydraulic brake system has been modelled using Dymola and the VeSyMA library.

During the braking, the vehicle's kinetic energy, which comes from its motion, is converted into brake power through the braking system. This brake power is then transformed into thermal energy (heat) due to friction between the brake pads and the disc. As a result, the brake disc heats up, and the temperature of the disc rises, which is a key factor in brake performance and safety. The brake system modelling is divided into two key sections: hydraulic and mechanical system and the thermal system.

The mechanical brake model generates braking force relative to the driver's pedal stroke to meet the required vehicle deceleration. In high ambient temperatures or during frequent braking, heated brake discs can lead to reduced brake torque, which poses a significant safety issue. Accurate prediction of brake disc temperature is essential for both durability and vehicle safety.

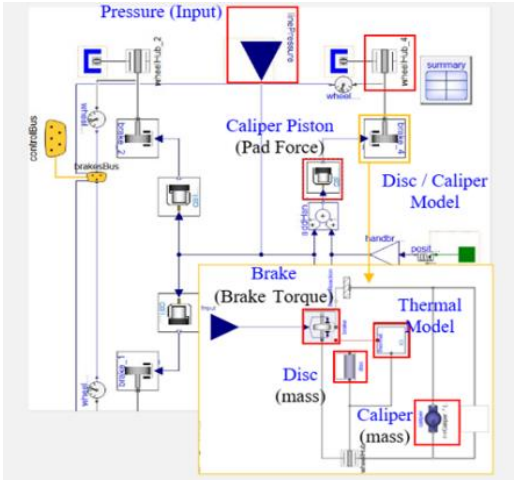


Figure 5. Mechanical brake model

Figure 5 shows the mechanical brake model, with hydraulic pressure coming from controller model, which is equally distributed to the front and rear calliper pistons. Brake Torque in the model is calculated based on pressure, piston area, disc radius, and pad mu.

$$Brake\ Torque = 2 \times \mu_pad \times A_{Piston} + R_{Eff.} \qquad (1)$$

The thermal model, shown in Figure 6, captures key heat transfer mechanisms conduction, convection, and radiation across components like the brake pad, inner/outer disc plates, vane, and disc hub thermal mass, allowing accurate prediction of brake disc temperature during operation.

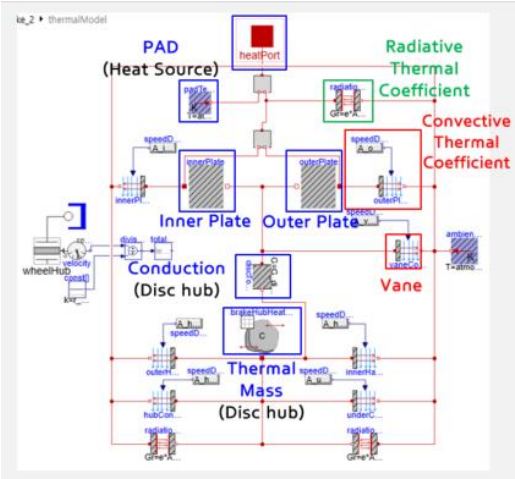


Figure 6. Thermal Brake Model

Figure 7 shows a comparison between the measured and simulated brake disc temperatures for the front left and front right wheels. The close match between the two lines means that the simulation is working well and can predict how the brake disc temperature changes during braking. This confirms that the thermal model is reliable and can be used to study how heat builds up and spreads in the brake system.

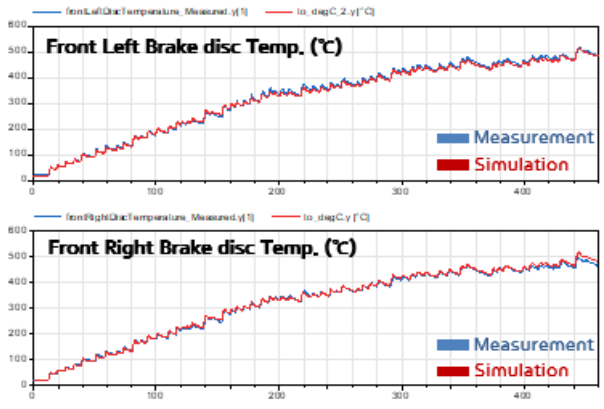


Figure 7. Comparison of the brake thermal model with experimental results

2.3 Suspension and Vehicle Dynamics

The multibody suspension systems are extended and modified from VeSyMA – Suspensions templates and models. The front and rear suspensions are variants of the MacPherson strut and Multilink linkages respectively. With examples of these existing in the VeSyMA - Suspensions library, the principal work was to parameterise them to create variants specific to the Ioniq 5 and validate them against test data.

While the initial aim was to create and validate the Ioniq 5 suspension, the long-term goal was to ensure easy future development, and ability to introduce future linkages and variants for other HKMC models.

There were three levels of fidelity used within the suspension models:

- No Bushed: Joints have no flexibility.
- Bushed: Bushes applied to appropriate linkage joints to match reality.
- Bushed Subframe: Adding bushes to the subframe, in addition to bushed linkage joints.

In all the fidelities, there was no flexibility of any of the components, just the joints, this was to both simplify, optimise but principally because adding component flexibility would add little benefit in terms of accuracy for the additional data and time requirement.

The data required to create each linkage matches the fidelity of the system, but in all cases the following data was required:

- Hardpoint locations (stated in and relative to HMC chosen orientation and origin).
- Component mass and inertia properties.
- Spring stiffness.
- Damper force curve.
- Bump stop force curve and clearance.
- Anti-Roll bar (ARB) stiffness.

In addition to all the above data, when adding bushes to either the linkage or the subframe, each bush needs the following data:

- Force, torque curve and damping constant in each axis.
- Bush orientation relative to the suspension.

When adding bushes to the subframe mass and inertia data is required for the subframe and all attached components as a lumped mass model. One key method to ensure that any future linkage can be easily created, developed and tested, mechanisms were implemented that

DOI
10.3384/ecp218819

reads all required data is read from external files using the open source ExternData Modelica Library[1].

Within ExternData there is a model that reads .xlxs files. Force suspension linkages there is a fixed format file for each linkage, created and used primarily by the HKMC CAE team.

Using this file as a source of all data means that both there was little chance of human error when copying data, but also that any further MacPherson or Multilink suspension from any future model can be created instantly by just changing the file that is being read.

To set this up, there was a parameter that stated the location of each item of data, both in terms of sheet (one parameter used for each group of data) and cell location (specific to each parameter). For the tables of data, such as damper and force curves, sizes of tables also needed stating. Doing this rather than hard coding locations allow any future changes to the .xlsx file format can be easily implemented without requiring version specific variants to be maintained.

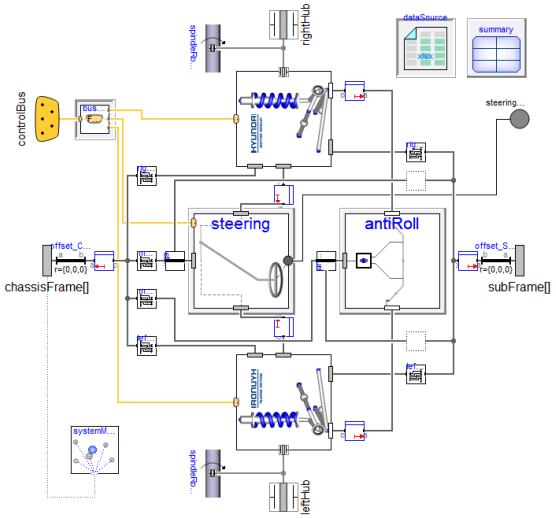


Figure 8. Front Suspension Model built based on HKMC's Standard excel template

One tedious, and repeating task of using any linkage in a vehicle is to ensure that the vehicle and suspension initialises correctly. This is important for 2 key benefits, because the hardpoints stated are for when the vehicle is resting on flat ground at curb weight; therefore, to ensure that when the vehicle settles the bushes and springs have the appropriate preload to ensure that the linkages, and by extension the geometry is in the correct positions. Secondly it ensures that simulations are more efficient because the initialisation of each vehicle is steady-state requiring no settling time at the start of each simulation and improving robustness of simulations by eliminating unstable initial conditions.

Whenever a new linkage, mass distribution, vehicle or bush parameterisation new initial settings need to be evaluated and implemented. But with each vehicle there are several load conditions that each need unique initial settings. Therefore, a series of initial functions were created to automate these tasks, performing the following functions:

- Curb weight Mass: Set the appropriate mass and inertia value in the body to ensure that the mass and inertia of the full vehicle is correct regardless of what components are used.
- Spring preload and initial attitude: find the preload in each spring and vehicle height, roll and pitch appropriate to keep the suspension in the correct position in curb weight condition.
- Bush preloads: Tune and store the preloads in each bush to ensure that the initial offset of each bush is zero when at curb weight.
- Load condition initialisation: Evaluate the body attitude, suspension positions and bush offsets (if included) for each load condition.

For each of the functions above, each value was calculated then the setting applied in the vehicle, half car and/or quarter car models, ensuring that any test at any level is using the linkages with the correct initial settings.

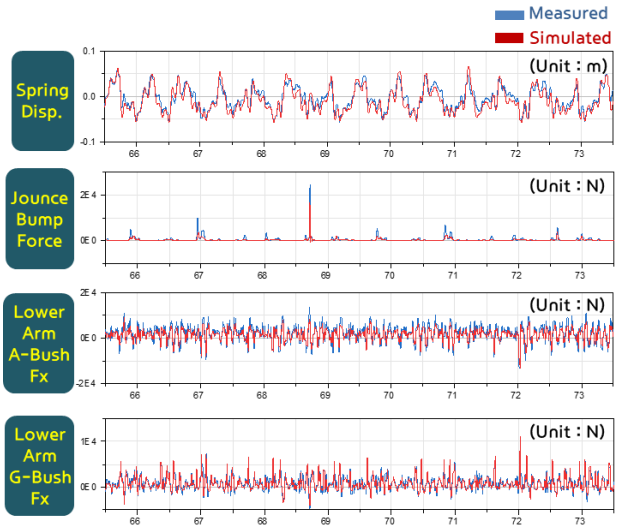


Figure 9. Validation of front left suspension from Half-car rig simulation on Belgian Pave

The first test the suspension models was to validate the dynamic spring length against measured data when a measured set of wheel forces and torques was applied to each wheel hub, comparing to a test that was driving over cobblestones in a 2up load condition. The test consisted of either the half car or full vehicle tests where the body was fixed in place and forces and torques applied to the wheel hubs. The forces and torques time traces were applied with the measured data being read in from the .mat file format they were stored in. The correlation, shown in **Figure 9**.

2.4 Electrical Motor

The investigated electric vehicle architecture combines conventional and in-wheel electric motor designs. At the front axle a conventional synchronous motor with permanent magnet (PSM). In addition, two In-Wheel Motors (IWMs) are installed at the left and right rear wheels. The durability analysis pays particular attention to the temperature development within the individual electric motors.

For building models of front motor and IWMs, the Electrified Powertrains Library developed by Dassault Systèmes is used [2]. Both types of motors are implemented based on table-based machine models from the library. The model computes the required electrical power according to the requested mechanical power with considering machine losses. The different types of machine losses are specified in form of multi-dimensional characteristic maps that typically depend on speed and torque. **Figure 10** shows the top-level structure of the table-based machine model.

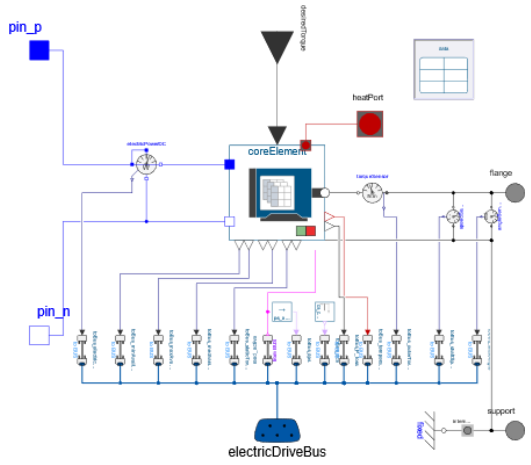


Figure 10. Top-Level structure of electric machine models

The thermal behaviours of the electric machines are modelled by lumped parameter thermal networks (LPTN). Specifically, for the IWMs that are equipped with an inner cooling channel, custom-developed thermal models are created. Heat transfer throughout the different layers of the IWM (housing to shaft and vice versa) is mainly considered in radial direction. The layers of the IWM are represented by thermal capacities in the core. During simulation, the individually computed losses by the electric machine models are injected into the respective thermal capacities. Heat exchange between the different layers is implemented via lumped thermal elements for modelling heat transfer mechanisms of conduction, convection and radiation. The cooling channel is modelled by means of components from the TIL-Library developed by TLK-thermo [3]. Behaviours of the brakes within the IWMs are modelled separately (see section 2.2).

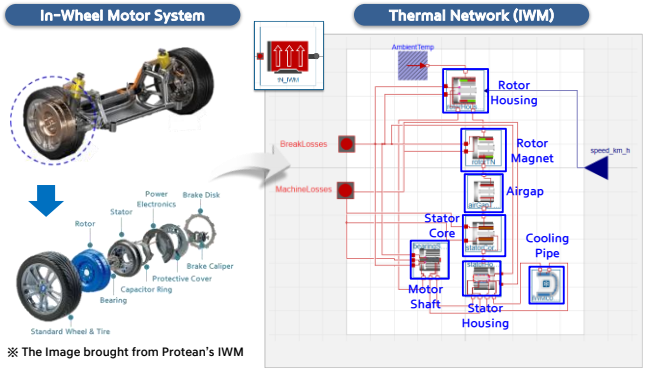


Figure 11. IWM Thermal Network Model

The electrical machine models are parameterised by provided characteristic power loss-maps extracted from test rig measurements. The maps are split into winding, core, magnet and friction losses. The thermal networks are parameterised by motor geometries and material specific properties like thermal conductivities, material densities and specific heat values.

The developed electrical and thermal motor models are validated by means of reference signals extracted from real-world driving scenarios. These reference data include, among other things, winding temperatures, speed and torque profiles at the machine flange, as well as the vehicle's velocity. **Figure 11** shows the validation results of two different compared driving scenarios focusing on the temperature development within the motor windings. The maximum deviation in temperature comparing both results is around 8.4 °C. In the current implementation a simplified setup and control strategy for the cooling channel is used. As a next step the cooling system is adapted to better fit with the real-world implementation.

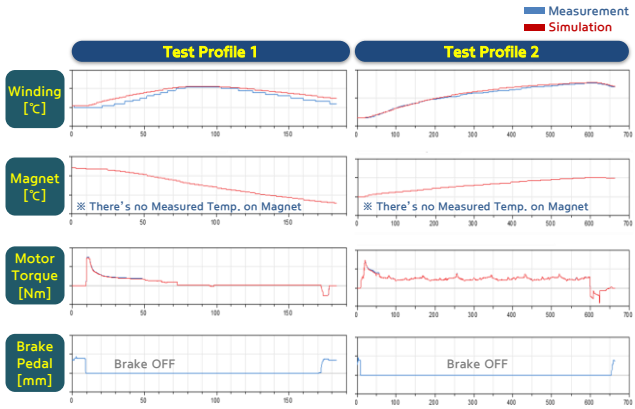


Figure 12. Validation results of the In-Wheel motor models

2.5 Battery Cell / Pack

The battery system implemented in this project is aligned to the battery pack of the Ioniq5 using an 800-Volt technology with a total capacity of up to 84 kWh. The battery pack comprises a variety of interconnected NMC-type lithium-ion pouch cells at the basis. The durability analysis pays particular attention to the temperature development within the individual battery cells of the

DOI 10.3384/ecp218819

pack when applying varying operational and environmental conditions.

Battery cell and pack models are developed based on equivalent circuit models (ECM) from the Dassault Systèmes Battery Library [4]. At the current phase of the project the focus is mainly set on the electro-thermal behaviour of cells (including aging characteristics is planned in a next step). **Figure 12** shows the structure of the used ECM cell model on the left-hand side ("Cell-Level"). The ECM cell is split into three sub-components to model electrical, thermal and aging behaviours separately. All three sub-components are continuously exchanging information via a bus connector during simulation. This approach offers a high degree of flexibility to replace individual sub-components. The selected thermal cell model can be discretised along the height-direction of the cell with additional thermal connectors at the surface sides for modelling heat exchange with the environment.

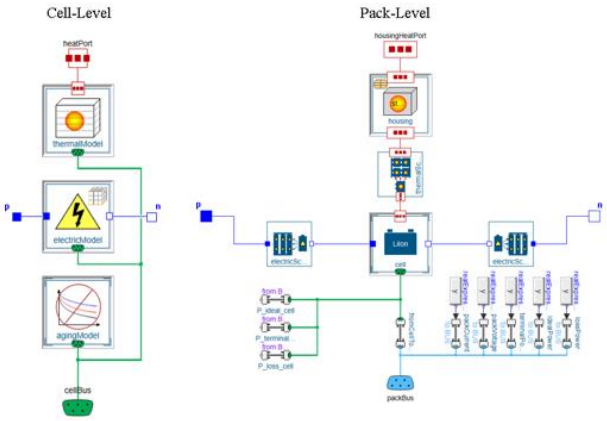


Figure 13. ECM Cell and Pack model concepts

The battery pack is modelled by means of scaled pack templates from the Battery library. Its structure is shown on the right-hand side of **Figure 13** ("Pack-Level"). Using this type of model, the behaviour of one cell instance is computed and scaled up according to the overall selected number of cells in series and parallel. The scaled model implementation is comparable to a pack setup in which all cells behave identically without deviations. This type of pack model is particularly suitable for analyses that focus on the battery pack as a complete system, such as long-term energy or aging studies. In contrast to the scaled packs, the Battery library also supports discretised pack models that calculate the behaviour of each cell individually. Thus, cell-to-cell interactions are also considered during simulation.

The ECM cell model is parameterised based on RC-data tables tested and 3D simulation. RC-data tables are used to compute the inner resistance and open-circuit-voltage of the battery cell depending on its states during simulation with respect to the cell temperature, State-of-Charge (SoC) and applied current loads at the terminal pins. The provided 3D simulation results are reused as reference temperatures within an automated data-correlation workflow for parameterising the thermal cell model.

The developed battery cell and pack models are validated by means of reference signals extracted from real-world test scenarios which consists of 7 laps in the circuit. This reference data includes, among others, captured temperatures, voltage responds and applied load profiles at cell and pack level. In **Figure 14**, the simulation results with focus on the computed temperatures within battery cells for an applied dynamic current load profile are plotted. Input current load profile as well as reference signals for the validation are provided by sensor data that are captured during running vehicle driving scenarios.

In the validation model, a simplified cooling system is implemented for pack cooling at the bottom side. The shown signals *Tmin_Pack* and *Tmax_Pack* in the diagram are the measured min./max. cell temperatures that appear inside the battery pack of the electric vehicle during operation. Signal *Tmean_Cell* in the diagram shows the computed mean core cell temperature of the scaled battery pack model (assuming that all cells behave identically).

The validation results show that computed cell temperatures are kept inside the min./max. reference data boundaries with a slight trend towards *Tmax_Pack*. By means of the scaled model simulations first initial insights into rough pack designs at early development phases are given and worst-case scenarios can be further analysed. As next steps, the use of discretised pack in combination of modified cooling system models are planned.

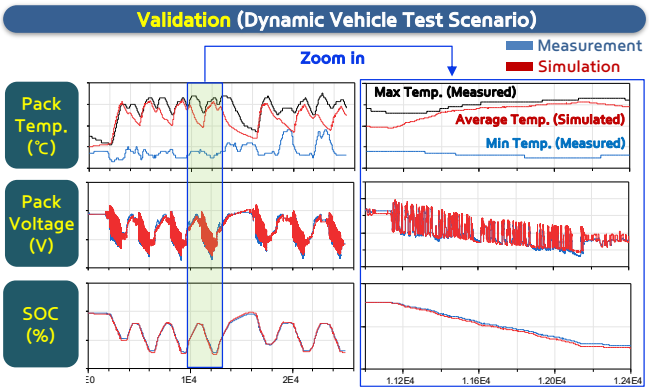


Figure 14. Pack-Level Temperature Validation

2.6 Thermal Management System

Effective thermal management is crucial for maintaining the efficiency, reliability, and longevity of EVs: the main parts (battery, inverts, electrical drive) need to work in specific temperature range for maximum efficiency and the comfort that has to be maintained in the passenger compartment. To be highly efficient and minimise its impact on the vehicle range, thermal management of EVs tries to dispatch and re-use dissipated heat as much as possible, making the circuitry and control logic associated very complex.

To evaluate properly the durability of the vehicle, the thermal management of the car needs to be modelled completely. It consists of:

- Battery Cooling System: Ensures the battery operates within safe temperature ranges to prevent degradation and maintain performance.
- Electric Motor Cooling: Manages the heat generated by the electric motor to enhance efficiency and prevent overheating, though a secondary lubrification loop.
- Power Electronics Cooling: Protects the power inverter and other electronic components from thermal stress.
- Cabin Heating and Cooling: Maintains passenger comfort while optimising energy usage.
- Heat Exchangers and Coolant Loops: Facilitate heat transfer between different components and the external environment.

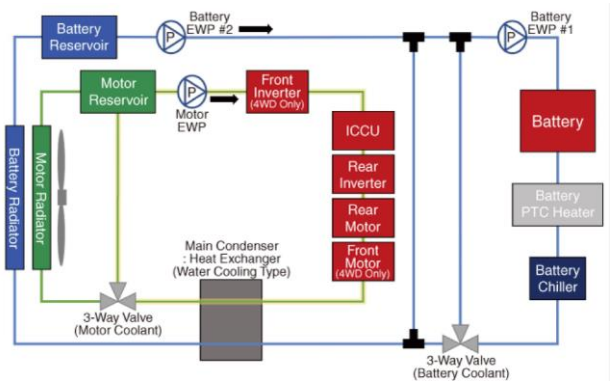


Figure 15. Overview of the thermal management system (Heat Pump is not represented in the figure here)

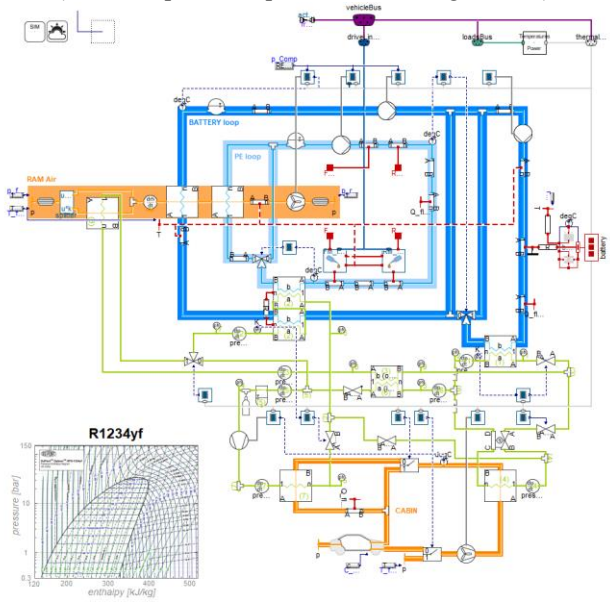


Figure 16. Thermal management system

The complete thermal management of the Ioniq5 has been built within Dymola, using the TIL library and its automotive add-on (pump, fan, compressor, TXV, valve, heat exchangers (plate & MPET), reservoir, cold plate…). It consists of five main paths or loops:

- the ram air for under the hood exchanges,
- the battery loop (glysantin 50%, dark blue),
- the e-drive and inverter loop (glysantin 50%, light blue)
- the cabin cooling path (orange)
- the reversible heat pump (R1234yf, green)

The work has been very meticulous, calibrating each component individually before building the system loop by loop. As much as possible, the model has been validated locally, using three different sets of measurement data: a partial model without the heat pump (replaced with imposed outlet temperature of coolant or air heat exchangers with the refrigerant) has been set-up for calibration.

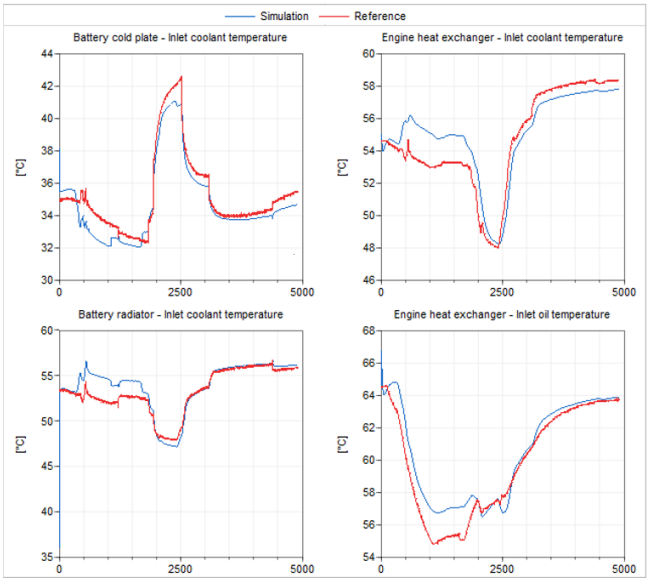


Figure 17. Calibration results for the Coolant loop

Once the loop is calibrated, the complete thermal management model has been assembled, and the control logic has been set-up.

The next step is to validate the complete model at vehicle level. The next step is to validate the complete model at vehicle level using the linkages with the correct initial settings.

3 Vehicle Modelling and Integration

By integrating all system models detailed above, the Ioniq5 electric vehicle model has been built as shown in **Figure 18**.

Since the vehicle model template is based on the VeSyMA library [5], there needs specific interfacing models for motor, battery and thermal management system models to exchange electrical and thermal signals with VeSyMA vehicle. The interfacing models make multi-physical system from different libraries to be simulated in one large vehicle environment.

DOI 10.3384/ecp218819

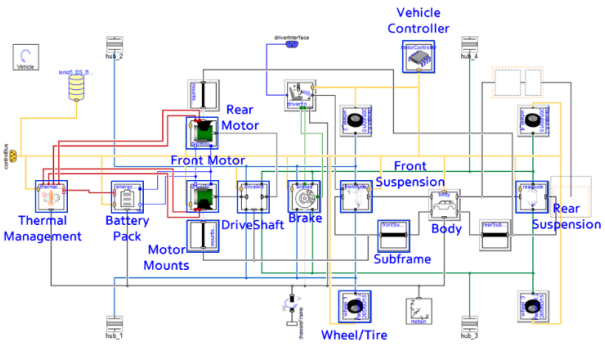


Figure 18. Integrated Electric Vehicle Model

Once the vehicle model is integrated, three representative durability scenarios previously defined in **Figure** 2 are utilised for the validation.

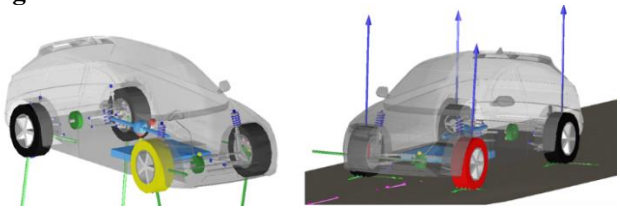


Figure 19. 2 ways of Simulation on Rough Roads

The first scenario for the validation is driving on a rough road which causes mainly longitudinal and vertical damage on suspension linkages and bush mountings. As a representative road roughness scenario, the event of consecutive potholes is chosen. For vehicle simulation, two different approaches are executed to check the availability for both options in the development stage depending on whether the pre-measured wheel forces exist or not. In **Figure 19**, the left case is replaying measured wheel forces directly on each wheel centres of the fixed-body vehicle model just to replicate the physical vehicle-level 24 channel road simulator. The right case is closed-loop simulation which the vehicle model runs on open CRG surface with grid Pacejka tire models simulating multi-contact points using the profile of measured time dependant velocity. **Figure 20** shows the spring displacement both measured and simulated as a representative validation signal for suspension model. Even though simple tire models are used for the simulation on rough road surface, the amplitude and trend are quite good matched to the measurement.

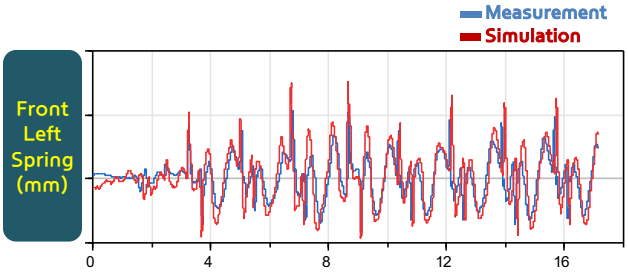


Figure 20. Spring displacement comparison between measurement and closed-loop simulation on CRG surface

The second scenario for validation is harsh acceleration and braking event which causes not only mechanical damage on brake and suspension components in longitudinal direction but also, electrical and thermal damage on the powertrain system.

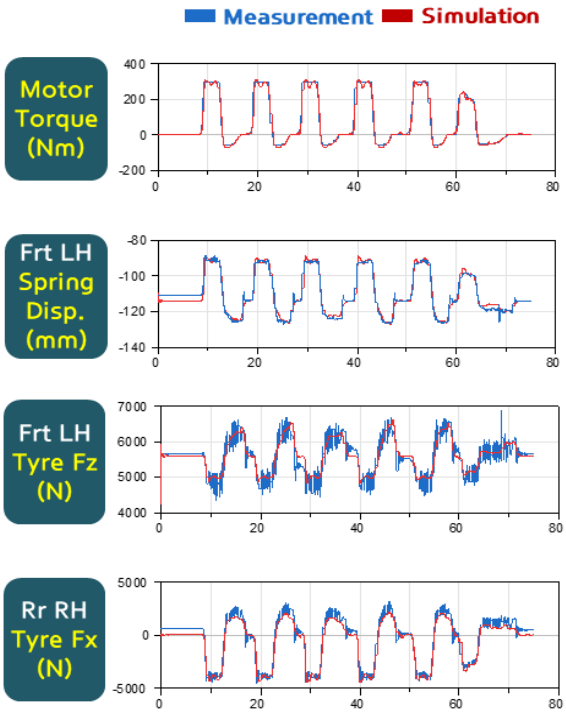


Figure 21. Suspension, Motor and Brake Validation for harsh acceleration and braking Scenario

The simulation is conducted using a closed-loop driver, by imposing measured vehicle speed as a target to drive the simulated vehicle to meet. As a result of the validation, it is due to the non-perfect logic in the vehicle controller, for the harsh braking condition, regenerative torques are not very accurate which also results in inaccuracies on brake pressure. With regards to the vehicle controller, a more accurate way of implementing the controller logic will be required in the future.

The last scenario for the validation is real-world driving scenario which is one of the mountain roads in Korea. First, road model was generated using latitude, longitude, altitude and measured velocity from GPS measurement. The road model defines the velocity target relative to distance which the driver model tries to achieve. In this scenario, powertrain dependant signals such as thermal and electrical behaviours on motor and battery are more taken into account for the durability. As shown in **Figure 23**, both front and rear motor speed are good matched to the measurement. With regards to the temperature on motor and brake, it's because there's no reference signals to be compared with, only simulated signals from the model are shown here but, they show reasonable behaviours relative to measured motor's behaviour which

the vehicle model to be believed that it's quite robust even in the real world scenario.

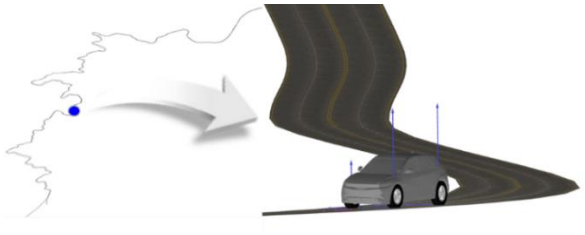


Figure 22. Road Model for field road in Korea

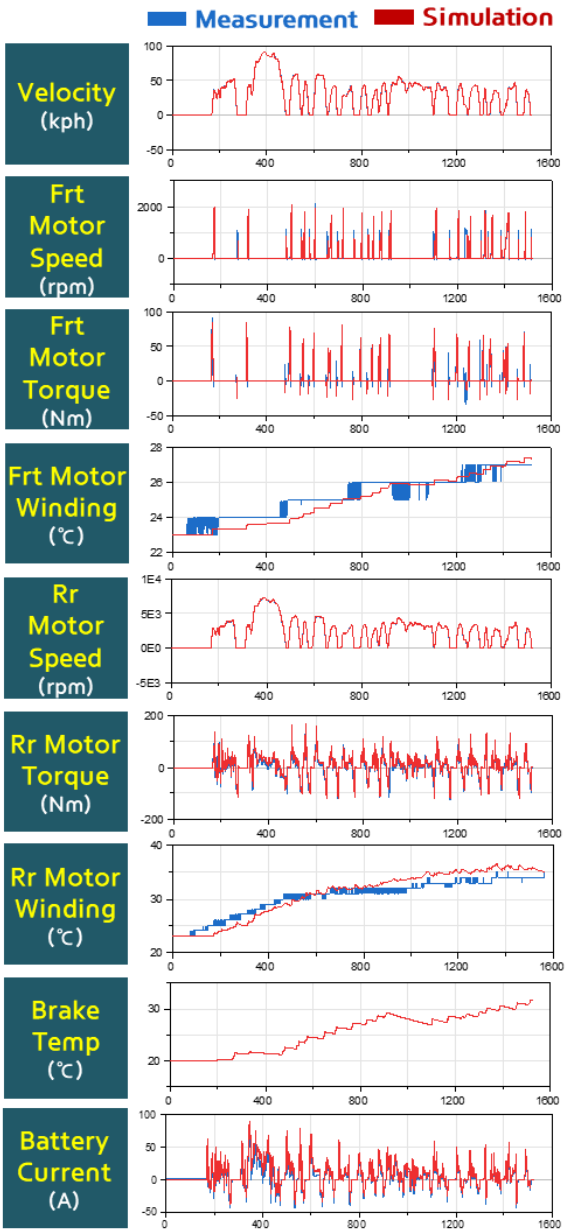


Figure 23. Powertrain / battery / brake related signals validation for the real-world scenario

4 Conclusion

In this study, Virtual RLDA platform has been developed for replacing vehicle measurement campaign for durability in early development stage as a best practice modelled for one of HKMC's electric vehicle.

For doing that first of all, required simulation signals aligned with the level of fidelity for system models are defined from vehicle simulation to quantify electrical, thermal and mechanical damages on each durability related components in systems.

Secondly, systems that are closely related to durability are modelled and validated with measurement. The system and vehicle level modelling templates from the study can be scalable and expendable for next HKMC vehicle development with minimal changes on the templates. In more detail, vehicle controller model is developed with Modelica library to control driving torque and regenerative torque for various driving situations. As for the degradation perspective of motor and battery system, in order to predict accurate working temperatures, electrical and thermal models for both motor and battery are developed along with the hydraulic brake model which generates brake disc heats. With those models, thermal management system modelled with TiL library for the entire vehicle level is coupled together with motor and battery model to cover the thermal behaviour occurred from actual driving conditions. For the perspective of chassis systems, one of the main development is the automated suspension modelling environment which directly imports HKMC's standard multi-body dynamics modelling template from the VeSyMA suspension library based suspension model not only for the initialisation of preloads on kinematic and compliance elements but also for the parameterisation of the suspension. As a result of this, it is expected that engineers can save huge amounts of time for modelling from basic K&C vehicle dynamics model to bushed vehicle dynamics model which can simulate nonlinear reactions on each suspension parts.

Lastly, the developed system models are integrated into one large vehicle model to simulate various different durability scenarios as similar as possible with the vehicle measurement. Different driver and road models are used for vehicle level simulation to validate vehicle model compared to measurement.

The Virtual RLDA platform developed in this study will be used and keep updated as a core model of our durability digital twin strategy to replicate realistic customer usages in virtual environment. By keeping the model upgraded, it is expected that many losses repeatably taken place from the development process could be minimised and it'll consequently result in reducing the number of prototypes for durability testing in the development.

Acknowledgements

This project has been supported by Claytex and Dassault Systèmes for 1 year. Various multi-physical systems and vehicle level modelling and their validities are proceeded. The model shows a lot of potential to be used for various systems in terms of durability development and will

continue to be updated in the future. I'd like to express my sincere gratitude to Claytex and Dassault Systèmes for their unbelievably enthusiastic support during the project.

References

[1] Thomas Beutlich and Dietmar Winkler. Efficient Parameterization of Modelica Models. In: Proceedings of the 14th International Modelica Conference. Ed. by Martin Sjölund, Lena Buffoni, Adrian Pop, and Lennart Ochel. Linköping, Sweden, September 2021. DOI: 10.3384/ecp21181141.
[2]https://www.3ds.com/assets/invest/2023-11/3ds-catia-electrified-powertrains-library-flyer.pdf
[3]https://www.tlk-thermo.com/de/software/til-suite
[4]https://www.3ds.com/assets/invest/2023-11/3ds-catia-battery-library-flyer.pdf
[5]https://www.claytex.com/products/dymola/model-libraries/vesyma/

DOI
10.3384/ecp218819

Solid-State Battery Systems and Thermal Management for Electric Long-Distance Buses

Jan Friedrich Hellmuth[1] Markus Pollak[2] Andreas Schulte[1] Wilhelm Tegethoff[1] Jürgen Köhler[1]

[1]Institut für Thermodynamik, Technische Universität Braunschweig, Germany
jan.hellmuth@tu-braunschweig.de
[2]TLK-Thermo GmbH, Braunschweig, Germany, m.pollak@tlk-thermo.com

Abstract

Solid-state batteries are promising for electric mobility as they potentially offer higher energy densities and therefore increased driving range, longer life and safety. They have different thermal properties and thermal management requirements than liquid battery systems. This paper presents a comparison of a solid-state battery system with a state-of-the-art liquid (Nickel-Manganese-Cobalt) NMC battery system for a simulated electric long-distance bus on a typical real-world long-distance route, requiring fast-charging. The battery characteristics are based on published data. Thermal management is performed by a reversible R744 heat pump. By designing both battery systems for the same energy capacity, the solid-state battery releases more heat during fast charging. For the same battery system size, the solid-state battery significantly outperforms the liquid battery system in driving range and does not require more cooling.

Keywords: Solid-State Battery, Thermal Management, Electric Mobility

1 Introduction

Electric mobility is on the rise worldwide and is recognized as a means to combat global warming and reduces local pollutant emissions (Sanguesa et al. 2021). Long-distance buses, also called coaches, are the most efficient transport option in terms of CO_2 emissions per person and kilometer (Thomas and Serrenho 2024; UK Department for Business Energy & Industrial Strategy 2022). In the current literature, solid-state batteries are discussed as the most promising alternative to state-of-the-art lithium-ion batteries because they have higher energy densities (thereby increasing driving range) and better safety (Schmaltz, Hartmann, et al. 2023). Solid-state batteries come with different electrical and thermal characteristics (Yang et al. 2024) than lithium-ion batteries and offer a larger operating temperature range, extending operation to higher temperatures in particular (Schmaltz, Wicke, et al. 2022). The main difference is the higher internal electrical resistance compared to a liquid battery operating at the same temperature level (Janek and Zeier 2023). First evaluations for the thermal management of solid-state battery vehicles investigated cold start procedures (Hughes and Vagg 2022) and heat pumps for battery pre-heating (Hellmuth, Pollak, T. Heinke, et al. 2023). Kan et al. (2024) reviewed the thermal effects of solid-state batteries and discussed the higher amount of heat generation and the higher operating temperatures of solid-state batteries. Yang et al. (2024) compared the thermal and electrical characteristics of conventional liquid, semi-solid-state and Lithium-Iron-Phosphate (LFP) batteries based on experimental data. Ma et al. (2025) experimentally investigated solid-state batteries and modeled thermal and electrical characteristics. Li et al. (2025) compared semi-solid-state and liquid batteries.

In this study, the effects of solid-state battery implementation in a coach for long-distance operation are evaluated using system simulation in Modelica. The battery data of Yang et al. (2024) are implemented. The following contributions are provided:

- Comparison of state-of-the-art and solid-state batteries in the context of a long-distance bus
- Battery system sizing comparison
- Evaluation of thermal effects in a real-world driving scenario
- Recommendations for the thermal management

2 Description of a coach model with solid-state batteries

To investigate the usage of solid-state batteries in a coach, we use a modular multi-scale vehicle model developed using Modelica (Modelica Association 2025) and integrated models from the TIL suite (TLK-Thermo GmbH 2024b). This approach allows for detailed simulation of electrical, mechanical and thermal interactions across various vehicle components. Similar modeling frameworks have been successfully applied to fuel-cell electric cars (S. Heinke et al. 2019) and to optimize thermal management systems in battery-electric trucks (Hellmuth, Steeb, Pollak, et al. 2023) as well as battery-electric coaches (Hellmuth, Pollak, Schulte, et al. 2024).

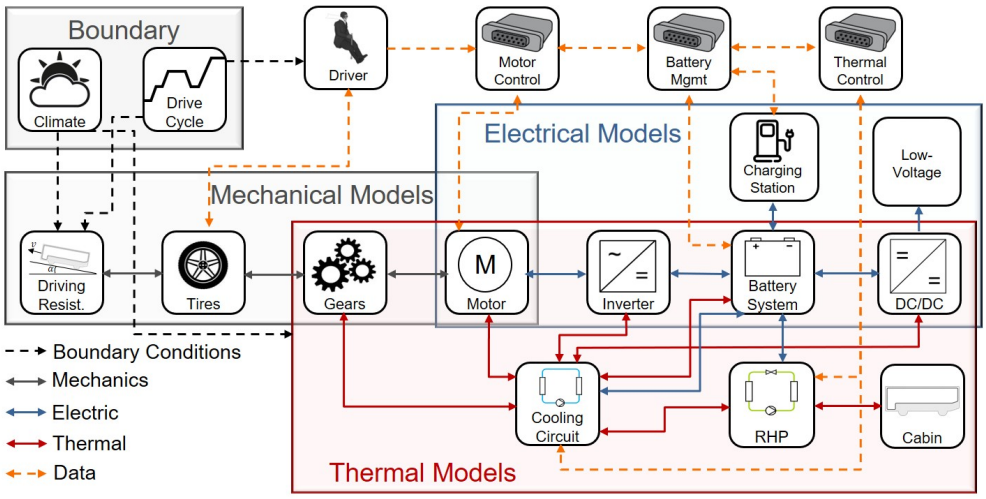


Figure 1. Model of an electric coach including balance rooms for flow variables (heat, electrical and mechanical power), control units, and data connections as well as boundary conditions and a charging station, adapted from Hellmuth, Pollak, Schulte, et al. (2024).

The comprehensive vehicle model is illustrated in Figure 1, highlighting key interactions and dependencies within the system. The balance rooms for the flow variables (heat, electrical, and mechanical power) as well as control units, data connections, and boundary conditions are visualized. The boundary conditions, thoroughly detailed in Section 3, encompass both climate data and a driving cycle. Specifically, we have defined ambient temperature, ambient humidity, and solar radiation as the primary climatic parameters. These factors not only affect driving resistance but also serve as boundary conditions for all thermal models. They have a significant impact on the cabin environment, the cooling circuit, and the reversible heat pump (RHP).

The drive cycle provides time-dependent data on the vehicle's speed and slope of the terrain. For the drive cycle we converted distance-based data into a time-based format. This fits well into the overall solution process of the formulated system of equations. A drive cycle from Karlsruhe to Munich is used in this study. This drive cycle was first described by Hebeler (2024) to investigate the potential of planetary gearboxes for the integration of a Rankine process in a coach. The drive cycle is displayed in Figure 2 showing the velocity and altitude along the route.

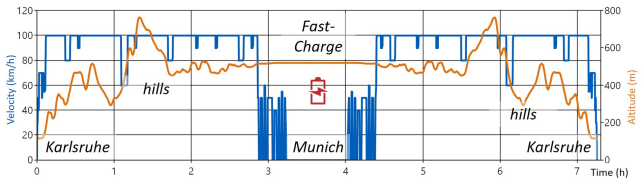


Figure 2. Drive Cycle showing the velocity and altitude profile for a long-distance coach trip from Karlsruhe to Munich and back and a fast-charging break, adapted from Hebeler (2024).

2.1 Mechanical Models for Drive-Train and Force

The mechanical subsystem includes models for driving resistance, tires, gears, and an electric motor to describe the driving dynamics of a configured vehicle. The driving resistance model, in particular, uses interchangeable parameter sets that include the drag coefficient c_w, frontal area A_{front}, vehicle weight $m_{vehicle}$, and rolling friction coefficient μ_{roll}. The parameters (Table 1) are taken from the *vanHool CX45E* (Sustainable Bus 2020). Traction force is calculated by summing the four resistances:

$$F_{Traction} = F_{Roll} + F_{Air} + F_{Grade} + F_{Acceleration} \quad (1)$$

Further description of the mechanical models can be found in (Hellmuth, Steeb, Pollak, et al. 2023; Hellmuth, Pollak, Schulte, et al. 2024).

Table 1. Vehicle parameters used in all scenarios.

Parameter	Value
Vehicle total mass	24 t
Frontal area	7.5 m^2
Drag coefficient	0.4
Rolling friction coefficient	0.006

2.2 Models of Electrical Components

The electrical motor model connects the mechanical and electrical models. Electrical power is calculated based on the required mechanical power. We assume a fixed motor efficiency of 95 % (Gobbi et al. 2024), and the waste heat is rejected to the coolant circuit. The motor is electrically connected to the inverter model, providing a constant voltage level for the motor. A fixed efficiency of 97 % (Sorokina et al. 2021) is assumed, and the waste heat is

DOI 10.3384/ecp218831

rejected to the coolant. The inverter is connected to the battery system that includes a high-voltage network. Furthermore, a low-voltage network provides energy for additional consumers such as ignition and infotainment. The high-voltage network is connected to an external charging station. Activation of the charging station can be requested by the drive cycle or can be time-based. A battery management controller sets the charge power based on charge maps. The charge maps use the actual battery temperature and state of charge to calculate the suitable charging power. While discharging, the battery management ensures de-rating for battery temperatures that are too high or too low as well as for voltages that are too low voltages.

2.3 Battery Model

The battery system is one of the key components of the vehicle model. Therefore, it is modeled in detail. Parts of the model were described before by Hellmuth, Steeb, Tegethoff, et al. (2022), Steeb et al. (2019), and Hellmuth, Steeb, Pollak, et al. (2023). We model a battery cell including copper and aluminum current arresters, a thermal interface material (*gapfiller*) and a simplified cooling plate model. The cooling plate is designed as bottom mounted cooling with round channel coils using a water-glycol coolant. Further discussion on cooling plate design can be found in Epp et al. (2023). One representative battery cell is modeled in details and the voltage, current, and heat flows are scaled up to the system level. The battery system is designed for a voltage supply of 800 V. The battery capacity for the baseline scenario is 473.6 kWh.

A pouch cell is chosen because solid-state batteries require high external pressure and only pouch cells can cope with this requirement (Yin et al. 2024). The battery cell is modeled in a 3D-discretization in Modelica. Figure 3 shows the modeling of the battery cell including the conductors, gapfiller, and cooling plate. In this sketch, the battery cell is discretized in length (x), thickness (y) and height (z). That approach is suited for vehicle simulation as the thermal interactions of the battery with other components are at the top (current arresters) and bottom (cooling plate). Automotive battery cells are usually relatively thin. The battery model comprises a network of 9 battery bundles. Each bundle represents a part of the battery cell including a part of the volume, mass, and thermal capacity along with the number of bundles (Veth et al. 2014). A thermal network is calculated in each bundle. The thermal network is built up from thermal resistors and a thermal capacity. The thermal capacity is implemented as a point mass in the center of each discretization element and is parameterized using the mass and heat capacity of the battery cell. Each resistor uses half of the element's size in a geometrical direction and the thermal conductivity. The thermal properties are assumed for the cell as a homogeneous block with anisotropic thermal conductivity (Wendland 2021).

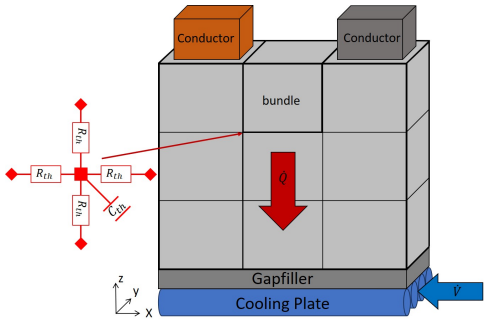


Figure 3. Illustration of a discretized battery cell with one bundle highlighted. Each bundle comprises a thermal network. The system includes conductors, a gapfiller and the cooling plate, adapted from Hellmuth, Steeb, Pollak, et al. (2023).

The battery cell model contains an equivalent circuit model (ECM) as well as a reversible heat model for calculating irreversible heat production ($\dot{Q}_{irrev}$) and reversible ($\dot{Q}_{rev}$) heat (Bernardi, Pawlikowski, and Newman 1985) The ECM calculates $\dot{Q}_{irrev}$ based on the internal resistance R_i of the battery cell:

$$\dot{Q}_{irrev} = R_i \cdot I^2 \tag{2}$$

ECM models contain electrical elements such as *Capacitor*, *Ohmic resistances* or *Warburg diffusion*. The models can be parametrized by resistance data of the cell or data obtained from *Electrochemical Impedance Spectroscopy* (EIS) measurements (Westerhoff 2019). For this study, a simple *R1RC* model (see Figure 4) is chosen as sufficient data are provided for that model complexity and accuracy.

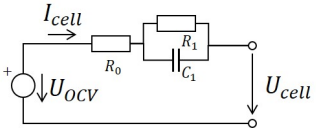


Figure 4. *R1RC* ECM model adapted from Westerhoff (2019).

The reversible heat model uses the entropy coefficient $\frac{\partial U_{OCV}(T,SoC)}{\partial T}$ to calculate the reversible heat $\dot{Q}_{rev}$ at a certain temperature and a given current:

$$\dot{Q}_{rev} = I \cdot T \cdot \frac{\partial U_{OCV}(T, SoC)}{\partial T} \tag{3}$$

From equations 2 and 3, the irreversible and reversible heat dependence on the current I can be analyzed. For calculating the irreversible heat, the current I is squared. The reversible heat depends linearly on the current I. The reversible heat can be released or consumed, depending on the entropy coefficient for different temperatures and different states of charge (SoC). A dependence on the battery state of health (SoH) is not implemented due to a lack of available data yet. A measurement for the load on the battery is the C-Rate, defined as: C-rate $= \frac{I}{Capacity}$

The battery data for this study are obtained from the publication of Yang et al. (2024). They compared the thermal characteristics of a solid-state battery (NMC-S) with one LFP battery and one liquid NMC (NMC-L) battery. Actually, the solid-state battery is a hybrid solid-liquid battery, also called semi-solid (Schmaltz, Hartmann, et al. 2023). In this study, we use the NMC-S and NMC-L battery data for comparison. The NMC-S battery was provided by *ProLogium Technology Ltd* and the NMC-L is a state-of-the-art battery from *CENAT New Energy Ltd*. The NMC-S battery uses NMC for the positive electrode and graphite for the negative electrode. The solid electrolyte is made of $Li_7La_3Zr_2O_{12}$ (*LLZO*). Some $LiPF_6$ liquid is added for filling the interface gap, making it a hybrid (Yang et al. 2024). *LLZO* is a *Garnet-type* oxide solid electrolyte and one of the most promising options (Schmaltz, Wicke, et al. 2022). Data of both batteries are compared in Table 2:

Table 2. Battery data for the NMC-S and NMC-L, adapted from Yang et al. (2024).

Parameter	NMC-S	NMC-L
Length X (mm)	300	311
Thickness Y (mm)	5.5	7.35
Height Z (mm)	100	100
Volume (L)	0.165	0.228
Mass (kg)	0.377	0.520
Nominal Capacity (Ah)	21.3	20
Nominal Voltage (V)	3.6	3.7
Operating Voltage (V)	$4.2-2.5$	$4.2-2.5$
Gravimetric Energy Density ($\frac{Wh}{kg}$)	203.4	142.3
Volumetric Energy Density ($\frac{Wh}{L}$)	464.7	323.7
Max Currents	$3C$	$2C$

The NMC-S battery cell is 27.6 % smaller in volume and 27.5 % lighter in mass and has a slightly higher battery capacity. This results in a much higher gravimetric and volumetric energy density. The operating voltages are similar. The NMC-S battery could be charged or discharged with $3\,C$, while the NMC-L is limited to $2\,C$.

In the first step, both battery systems are designed for the same energy capacity. A $800\,V$ battery system needs approximately 200 battery cells to be connected in series. For the lower nominal voltage, the NMC-S battery is equipped with 205 battery cells in series. To reach the same energy capacity, the NMC-S system is equipped with 30 cells in parallel, the NMC-L with 32. This results in a system capacity of approximately $470\,kWh$. The battery system data are compared in Table 3.

2.4 Thermal Management

The thermal management in the coach is based on a reversible heat pump using the natural refrigerant R744 and additional cooling circuits using a water-glycol coolant. The reversible heat pump provides cooling, heating, and dehumidifying for the cabin, as well as a connection to the cooling circuit. The cooling circuit provides cooling for the battery, motor, and inverter. Waste-heat recovery could also be enabled.

Table 3. Battery system comparison for NMC-S and NMC-L.

Parameter	NMC-S	NMC-L
Battery Cells ($-$)	6150	6400
System Volume (L)	1010	1460
System Mass (kg)	2318	3328
Energy Capacity (kWh)	471	473
Nominal Voltage (V)	738	740

A detailed cabin model (TLK-Thermo GmbH 2024b; Kaiser 2019) was implemented, which models the transient behavior. The interior model is divided into three discretization sections that enable consideration of specific effects, such as the solar radiation on the windshield, and the extraction of recirculated air and supply air from the roof-mounted system in the middle zone. Components, such as the windows and roof, are modeled with their thermal properties. The heat exchangers required for cabin air conditioning are placed in the *Roofbox* and the *Frontbox* (Kaiser 2019).

The reversible R744 heat pump is modeled with the liquid, refrigerant, and gas components provided by the *TIL suite* (TLK-Thermo GmbH 2024b) that enable numerically efficient modeling of thermodynamic systems (Schulze 2013). Components, such as the heat exchangers, are modeled in detail. The main cooling and heating capacity is provided for the cabin. The implemented heat pump uses a four-way-valve for reversing the refrigerant cycle and the secondary loop coolant circuit is used for cooling and heating of the *Frontbox* (Weinreuter and Yildirim 2024). Similar thermal management systems are discussed in previous publications (Hellmuth, Steeb, Pollak, et al. 2023; Hellmuth, Pollak, Schulte, et al. 2024).

The battery cooling circuit operates at temperature levels from $20-40\,^{\circ}\mathrm{C}$ for liquid batteries and up to $40-60\,^{\circ}\mathrm{C}$ for solid-state batteries. For the solid-state battery a de-rating of power is begins above $70\,^{\circ}\mathrm{C}$. A connection to the low temperature cooling circuit allows mixing cold coolant into the circuit, dissipating the battery heat and keeping the temperature level. Different control concepts for various operating strategies and ambient conditions are implemented with a set of *PI-Controllers*. The cabin cooling is prioritized during driving. For fast-charging, the operation strategy shifts to a battery cooling priority. Further optimizations are performed, such as using a dynamic *Gascooler* pressure correlation adapted from Kauf (1999) for controlling the system high pressure.

DOI 10.3384/ecp218831

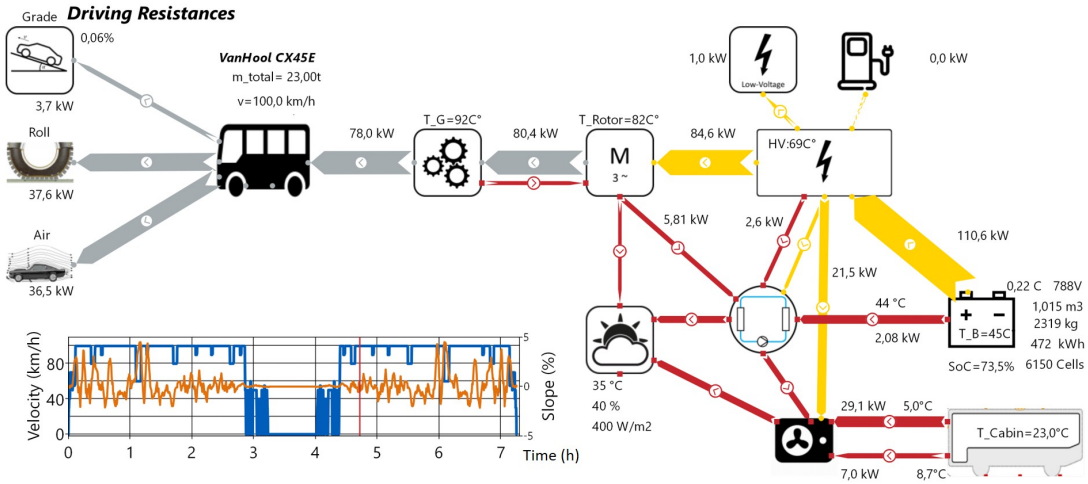


Figure 5. Analysis of energy flows for the NMC-S battery system in the context of the full vehicle, Drive cycle time is indicated by the vertical red line, adapted from Hellmuth, Steeb, Pollak, et al. (2023).

3 Comparison of Battery Systems: Solid-State and Liquid

The solid-state and liquid battery systems are compared in the context of the described long-distance route (see Figure 2). First, the energy flow in the full-vehicle context is evaluated. Next, the battery systems are compared for the same energy capacity. Finally, the systems are compared for the same battery system volume, but different energy capacities.

3.1 Analysis of Energy Flow in the Vehicle

The analysis of energy flows is performed with the software *DaVE* (TLK-Thermo GmbH 2024a). Analyzing energy flows in the full-vehicle context helps to understand cross-couplings in the complex system. The energy flows are depicted in Figure 5. The NMC-S system is taken. At the driving cycle time of the snapshot, the coach drives up a slight incline at a velocity of 100 km/h. This results in a relatively equal distribution of the required power into grade, roll and air resistance. The efficiencies of the mechanical and electrical components can be obtained from the higher electrical energy flow values towards the battery. 110.6 kW electrical power is provided. The battery has a SoC of 73.5 % and the C-rate is 0.22. The battery system voltage is at 788 V.

The snapshot was taken for the driving scenario at an ambient temperature of 35 °C, an ambient humidity of 40 % and a solar intensity of 400 $\frac{W}{m^2}$. Under these conditions, thermal management adds about 25 % to the electrical energy consumption required for the motor. Additionally, 1 kW of electrical power is needed in the low-voltage network for entertainment, lights, and further demands.

The solid-state battery is conditioned to 45 °C and at point in the drive cycle very little cooling is necessary. Therefore, only 2.1 kW of heat is transferred to the thermal management. The cabin is conditioned to 23 °C and therefore, fresh air is provided at a temperature of 5 °C for the main cabin and 8.7 °C for the *Frontbox*. A total of 36.1 kW of cooling power is required for cabin cooling.

3.2 Comparison for same Battery System Energy Capacity

As described above, both battery systems are designed for the same energy capacity. They are compared for the Karlsruhe - Munich - Karlsruhe (Figure 2) long-distance drive cycle. The comparison of both battery systems in terms of battery temperature and SoC is displayed in Figure 6. The NMC-L discharges a little bit faster because it needs slightly more energy for battery cooling while the NMC-S battery is not cooled while driving. The other reason for the higher energy consumption is the higher weight of the NMC-L bus. Also, the NMC-L battery system weighs 1000 kg more, resulting in higher grade and roll resistances while driving.

The battery temperature setpoint for the NMC-S is set to 45 °C. In the first driving section, prior to fast-charging which occurs just after the of hour three of the drive cycle, the battery temperature rises from an initial value of 35 °C up to close 40 °C and stays at that temperature level. Therefore, it is not cooled. The NMC-L is cooled to 30 °C to reduce battery degradation. During fast-charging, NMC-S warms up sharply to 59 °C, due mainly to the high internal resistances of the cell and high currents supplied in the fast-charging process. The NMC-L only warms up to 34 °C. This is attributable to the significantly lower internal resistance.

Figure 7 shows the heat produced by the battery

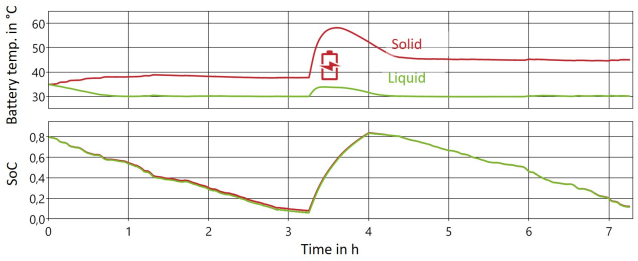


Figure 6. Comparison of the solid-state (NMC-S) and liquid (NMC-L) battery systems along the drive cycle: Battery temperature and SoC.

systems. The heat produced is generally relatively small in normal driving conditions due to the low C-rates. When comparing battery capacity and maximum motor power, the C-rate is limited to a value of 0.76 C while driving; $0.1 - 0.3$ C is typical. This results in relatively low irreversible heat as the currents are squared in calculating it. During fast-charging, the C-rate reaches a maximum close to a value of 2 C (1 MW charging). Therefore, the irreversible heat production peaks at 82 kW for the NMC-S and at 36 kW for the NMC-L. The higher irreversible heat production results from the internal resistances that are higher by the factor 2.3, even at the elevated temperature of the NMC-S.

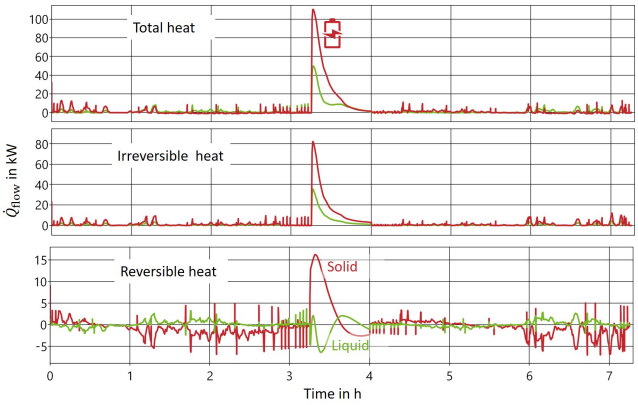


Figure 7. Comparison of battery systems along the drive cycle: Total battery heat, irreversible heat and reversible heat.

The reversible heat can be released or consumed. For the NMC-S, it is negative most of the time while it is discharging (SoC < 0.6), with some positive peaks for recuperation. If the reversible heat is negative, it reduces the total battery heat. The reversible heat dominates for low C-rates; therefore, the battery temperature does not increase for most of the time during normal driving for the NMC-S. Then, at the beginning of charging, it is positive and increases the total heat. For the NMC-L battery, the reversible heat turns to negative values while charging and reduces the produced heat.

3.3 Comparison for same Battery System Volume

In the previous section, it was shown that the solid-state battery heats up to significantly higher temperatures due to higher internal resistances. Because of the higher volumetric and gravimetric energy density, the NMC-S system is smaller and lighter. Assuming a given space for the battery system, sizing the battery systems for the same volume is a logical next step in their comparison. The adjusted parameters for the two battery systems are listed in Table 4:

Table 4. Battery system comparison for NMC- S and NMC- L. Batteries are sized for the same system volume.

Parameter	NMC-S	NMC-L
Battery Cells $(-)$	8815	6400
System Volume (L)	1450	1460
System Mass (kg)	3323	3328
Energy Capacity (kWh)	676	473
Nominal Voltage (V)	738	740

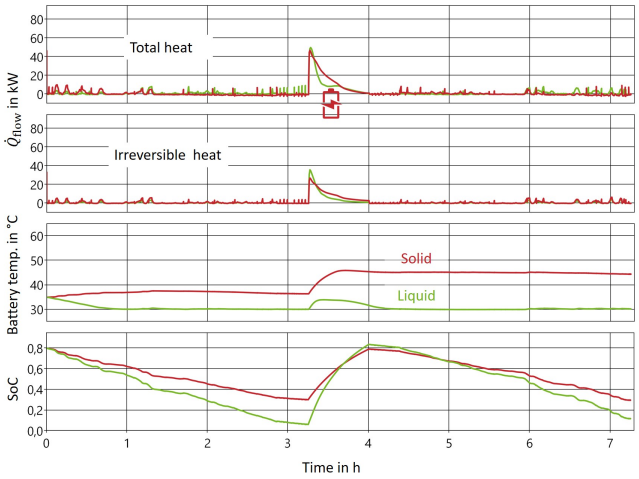


Figure 8. Comparison of batteries at the same system volume along the drive cycle: Total battery heat, irreversible heat, battery temperature and SoC.

In this configuration, both battery systems have approximately the same volume and weight. The NMC-S system has a 43 % higher energy capacity. A comparison of heat flow, battery temperature and SoC for the drive cycle is depicted in Figure 8. The higher energy capacity of the NMC-S leads to much lower C-rates during driving and charging. For this reason, the heat produced by the NMC-S battery is in the same range as that of the NMC-L. The lower C-rates due to the higher capacity compensate for higher internal resistances regarding heat production. Actually, not much cooling is necessary for the large NMC-S. It does not reach the upper setpoint temperature of 45 °C driving to Munich, but slowly heats

DOI 10.3384/ecp218831

up to it during fast-charging.

In the previous comparison, the solid-state battery system's full potential for fast charging is not utilized. In the fast charging it is only charged with a C-rate of 1 C, while up to 3 C would be possible. Also, the battery reaches the charging stop at an SoC of 0.3, so charging is not needed. For a simple scenario adjustment, the bus with the NMC-S battery starts at a SoC of only 0.6 and the charging break is reduced to 30 minutes. Therefore, the charging station is modified to allow up to 2 MW of charging power, which is likely to be possible at large charging stations in the future (Makoschitz 2022).

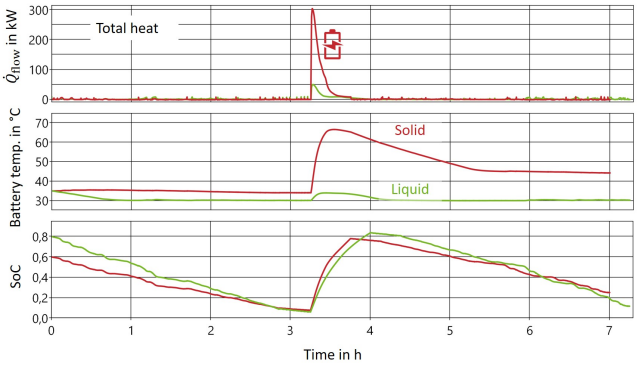


Figure 9. Comparison of batteries at the same system volume along the drive cycle with faster charging for NMC-S battery: Total battery heat, battery temperature and SoC.

An evaluation of the adjusted scenario is depicted in Figure 9. The bus with the NMC-S battery charges in just 30 minutes with a C-rate of up to 2.8 C to an SoC of 80 %. The bus reaches the final destination in Karlsruhe 15 minutes faster with a SoC of 0.25 remaining. During the first driving section from Karlsruhe to Munich, the NMC-S battery does not rise in temperature, due to the low currents and correspondingly low irreversible heat flows. The reversible heat is mostly negative at the low SoCs, so the battery consumes heat. Both effects roughly cancel each other out resulting in a constant battery temperature at the initial 35 °C. The high currents during fast charging result in a heat flow of up to 300 kW for a short time. The battery temperature rises rapidly during charging to 67 °C, this is still below the set de-rating temperature of 70 °C for the solid-state battery. After charging, the battery is slowly cooled down to the operating temperature of 45 °C.

The slow cool-down is caused by the current thermal management battery cooling capacities. More data on degradation of solid-state batteries would be needed for evaluating if such high temperatures for a longer time accelerate battery degradation significantly. The thermal management could be re-designed for enabling higher battery cooling capacities. Different operating strategies optimizing battery degradation and thermal management energy consumption could be tested by system simulation as has been done before for battery-electric heavy-duty trucks (Hellmuth, Steeb, Pollak, et al. 2023).

4 Conclusions

In our study, we demonstrated that a physics-based full-vehicle Modelica model of a battery-electric coach paired with real-world drive scenarios can be used efficiently to compare different types and configurations of battery systems. We implemented battery data from a recent study (Yang et al. 2024) for a liquid NMC battery and a (semi-) solid-state battery into the vehicle model and designed different battery systems. Furthermore, we modeled a state-of-the-art thermal management system using the natural refrigerant R744.

Our results show that in general the higher internal resistances of solid-state batteries lead to higher heat production in the battery system. This is especially the case for fast-charging with high C-rates. The higher operation temperatures of solid-state batteries partially compensate for that. For normal driving of a coach, C-rates are typically very low, resulting in small heat loads. Additionally, the reversible heat plays a role, providing cooling at some operation points. In our first battery system design, both systems are sized for the same energy capacity. The NMS-S produces more heat, but saves weight and only heats up significantly during fast-charging. The maximum temperature during charging of 59 °C is still within the operation limits.

The second battery system design uses the same size or battery volume. Therefore, the solid-state battery has a 43 % higher energy capacity and respectively, lower C-rates during operation. Therefore, the heat production is in a similar range as for the liquid battery.

Future research should evaluate use cases for cold ambient temperatures and discuss battery pre-heating strategies and cold-start performances. Furthermore, battery waste-heat utilization for cabin heating should be explored. Battery degradation, cabin comfort goals as well as thermal management energy consumption should be considered. For this, different thermal management system topologies and operation strategies will need to be evaluated.

Acknowledgments

Our work was funded by the European Regional Development Fund (ERDF) in the EUROPA FÜR NIEDERSACHSEN funding as part of the TOFEBAS research project.
The authors would like to thank the student assistants Jonte Tiedemann and Luca-Leon Lohnert for their help in preparing the data and models.

References

Bernardi, D., E. Pawlikowski, and J. Newman (1985). "A General Energy Balance for Battery Systems". In: *Journal of The Electrochemical Society* 132.1, pp. 5–12. DOI: 10.1149/1.2113792.

Epp, Alexander et al. (2023). "Simulative Investigation of Optimal Multiparameterized Cooling Plate Topologies for Different Battery System Configurations". In: *Energy Technology* 11.9. ISSN: 2194-4288. DOI: 10.1002/ente.202300405.

Gobbi, Massimiliano et al. (2024). "Traction motors for electric vehicles: Maximization of mechanical efficiency – A review". In: *Applied Energy* 357, p. 122496. DOI: 10.1016/j.apenergy.2023.122496.

Hebeler, Georg Maximilian (2024). "Modellbasierte Potenzialanalyse von Planetengetrieben zur Integration eines Rankine-Prozesses in einen Reisebus". PhD thesis. Technische Universität Braunschweig. DOI: 10.24355/DBBS.084-202412101034-0.

Heinke, Steffen et al. (2019). "High-Temperature Refrigeration System for Cooling of Automotive PEM Fuel Cells". In: 2nd Conference on High Temperature Heat Pumps, Copenhagen, Denmark, September 2019. URL: https://orbit.dtu.dk/en/publications/book-of-presentations-of-the-2nd-symposium-on-high-temperature-he.

Hellmuth, Jan Friedrich, Markus Pollak, Timo Heinke, et al. (2023). *Wärmepumpe für Nutzfahrzeuge mit Feststoffbatterien*. DKV-Conference 2023, Hannover, AA III.16.

Hellmuth, Jan Friedrich, Markus Pollak, Andreas Schulte, et al. (2024). "Assessment of natural refrigerants for thermal management in battery-electric long-distance buses". In: 16th IIR Gustav Lorentzen Conference on Natural Refrigerants, College Park, Maryland, USA 2024. DOI: https://dx.doi.org/10.18462/iir.gl2024.1153.

Hellmuth, Jan Friedrich, Michael Steeb, Markus Pollak, et al. (2023). "Innovative thermal management operathellmuing strategies for battery-electric heavy-duty trucks". In: Proceedings of the ECOS Conference 2023, Las Palmas de Gran Canaria. DOI: 10.24355/DBBS.084-202310260841-0.

Hellmuth, Jan Friedrich, Michael Steeb, Wilhelm Tegethoff, et al. (2022). "Thermische Simulation im Gesamtfahrzeugkontext". In: *Werkstoffsymposium 2022, Wolfsburg, 17.-18. Mai 2022*.

Hughes, Ryan and Christopher Vagg (2022). "Assessing the Feasibility of a Cold Start Procedure for Solid State Batteries in Automotive Applications". In: *Batteries* 8.2, p. 13. DOI: 10.3390/batteries8020013. URL: https://www.mdpi.com/2313-0105/8/2/13.

Janek, Jürgen and Wolfgang G. Zeier (2023). "Challenges in speeding up solid-state battery development". In: *Nature Energy* 8.3, pp. 230–240. ISSN: 2058-7546. DOI: 10.1038/s41560-023-01208-9. URL: https://www.nature.com/articles/s41560-023-01208-9.

Kaiser, Christian (2019). "Untersuchungen zur Effizienz- und Leistungssteigerung von Omnibusklimaanlagen". Dissertation. Braunschweig: Technische Universität Braunschweig. DOI: 10.24355/dbbs.084-202005181332-0.

Kan, Ruyu et al. (2024). "Thermal effects of solid-state batteries at different temperature: Recent advances and perspectives". In: *Energy Storage Materials* 68, p. 103366. DOI: 10.1016/j.ensm.2024.103366.

Kauf, Friedrich (1999). "Determination of the optimum high pressure for transcritical CO2-refrigeration cycles". In: *International Journal of Thermal Sciences* 38.4, pp. 325–330. DOI: 10.1016/S1290-0729(99)80098-2.

Li, Haibin et al. (2025). "Electro-thermal coupling modeling and heat generation decoupling analysis of semi-solid-state lithium-ion battery". In: *Electrochimica Acta* 512, p. 145455. ISSN: 0013-4686. DOI: 10.1016/j.electacta.2024.145455.

Ma, Xiaole et al. (2025). "Modelling and verification of the electro-thermal coupling model for solid-state batteries based on equivalent circuit model". In: *Applied Thermal Engineering* 269, p. 126063. DOI: 10.1016/j.applthermaleng.2025.126063.

Makoschitz, Markus (2022). "Key aspects to enable multi-megawatt fast charging". In: *e & i Elektrotechnik und Informationstechnik* 139.4-5, pp. 435–448. ISSN: 0932-383X. DOI: 10.1007/s00502-022-01037-w.

Modelica Association (2025). *Modelica – A Unified Object-Oriented Language for Systems Modeling: Language Specification: Version 3.6*. URL: https://specification.modelica.org/maint/3.6/MLS.pdf.

Sanguesa, Julio A. et al. (2021). "A Review on Electric Vehicles: Technologies and Challenges". In: *Smart Cities* 4.1, pp. 372–404. DOI: 10.3390/smartcities4010022.

Schmaltz, Thomas, Felix Hartmann, et al. (2023). "A Roadmap for Solid–State Batteries". In: *Advanced Energy Materials* 13.43. ISSN: 1614-6832. DOI: 10.1002/aenm.202301886.

Schmaltz, Thomas, Tim Wicke, et al. (2022). *Solid-State Battery Roadmap 2035+*. DOI: 10.24406/publica-68.

Schulze, Christian Wilhelm (2013). "A Contribution to Numerically Efficient Modelling of Thermodynamic Systems". Dissertation. TU Braunschweig. DOI: 10.24355/dbbs.084-201409121113-0.

Sorokina, Nina et al. (2021). "Inverter and Battery Drive Cycle Efficiency Comparisons of Multilevel and Two-Level Traction Inverters for Battery Electric Vehicles". In: *2021 IEEE International Conference on Environment and Electrical Engineering and 2021 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I&CPS Europe)*. IEEE, pp. 1–8. ISBN: 978-1-6654-3613-7. DOI: 10.1109/EEEIC/ICPSEurope51590.2021.9584705.

Steeb, Michael et al. (2019). "Avoiding Thermal Hotspots in Automotive Battery Systems using a Multiscale Full Vehicle Model". In: *16th Symposium on Modeling and Experimental Validation of Electrochemical Energy Technologies, Braunschweig, 12.-13. March 2019*.

Sustainable Bus (2020). *Van Hool electric coach is reality. The first unit of the CX45E shipped to the US*. URL: https://www.sustainable-bus.com/news/van-hool-electric-coach-cx45e-usa/.

Thomas, Hugh and André Cabrera Serrenho (2024). "Using different transport modes: An opportunity to reduce UK passenger transport emissions?" In: *Transportation Research Part D: Transport and Environment* 126, p. 103989. ISSN: 1361-9209. DOI: 10.1016/j.trd.2023.103989. URL: https://www.sciencedirect.com/science/article/pii/S1361920923003863.

TLK-Thermo GmbH (2024a). *DaVE*. URL: https://www.tlk-thermo.com/en/software/dave.

TLK-Thermo GmbH (2024b). *TIL-Suite*. URL: https://www.tlk-thermo.de/en/software/til-suite.

UK Department for Business Energy & Industrial Strategy (2022). *Government conversion factors for company reporting of greenhouse gas emissions*. URL: https://www.gov.uk/government/publications/greenhouse-gas-reporting-conversion-factors-2024.

Veth, C. et al. (2014). “Electro-Thermal Model Approach for the Prediction of Internal State Values in Large-Format Lithium Ion Cells and Its Validation”. In: *Electrochem. Soc. 161 (14), A1943-A1952*. DOI: https://doi.org/10.1149/2.1201412jes.

Weinreuter, Tobias and Kemal-Edip Yildirim (2024). *Modulares Thermomanagement in Bussen mit batterieelektrischem Antrieb in Verbindung mit einer R744 Wärmepumpe*. 13. VDI-Fachkonferenz Thermomagement für elektromotorisch angetriebene PKW, 27.11.2024, Aachen.

Wendland, Robert (2021). “Ein Beitrag zur thermischen Auslegung von Fahrzeug-Batteriesystemen”. Dissertation. Technische Universität Braunschweig. DOI: 10.24355/dbbs.084-202106241143-0.

Westerhoff, Uwe (2019). “Klassifizierung von Lithium-Ionen-Batteriezellen mittels kenngrößenbasierter Methoden”. Dissertation. Technische Universität Braunschweig, Verlag Dr. Hut. ISBN: 978-3-8439-4179-2.

Yang, Rui et al. (2024). “Comparative Study on the Thermal Characteristics of Solid-State Lithium-Ion Batteries”. In: *IEEE Transactions on Transportation Electrification* 10.1, pp. 1541–1557. DOI: 10.1109/TTE.2023.3289997.

Yin, Zhen et al. (2024). “Battery Management System Towards Solid-State Batteries”. In: *IEEE CHAIN* 1.4, pp. 319–353. DOI: 10.23919/CHAIN.2024.000011.

DOI
10.3384/ecp218831

A low complexity physics-based aging model for lithium ion cells with solid electrolyte interphase and lithium plating side-reactions

Alberto Romero[1] Elias Steinkellner[1] Johannes Angerer[1] Luca Belforte[2]

[1]Kreisel Electric, Austria, {alberto.romero, elias.steinkellner, johannes.angerer}@kreiselelectric.com
[2]Centro Ricerche FIAT (CRF), Italy, luca.belforte@crf.it

Abstract

In this paper, we extend our previously presented lithium ion battery model to include electrochemical aging dynamics, considering both calendaric and cyclic aging behavior. Accordingly, the Equivalent Hydraulic Model (EHM) is augmented with side reactions for Solid Electrolyte Interphase (SEI) growth and lithium plating. The model is validated against experimental results, showing a level of agreement that allows us to evaluate the effect of various charging and cooling strategies under real-world duty cycles. A case study concerning a battery electric passenger vehicle that operates under changing seasonal conditions illustrates the advantages of using such a model. Results show that the vehicle thermal management and the recharge strategy play a critical role in ensuring a long and safe operational life. In particular, and in agreement with the existing literature, more frequent charges that maintain a lower average state of charge, together with a lithium plating conscious thermal management, significantly extend battery lifetime.

Keywords: batteries, electrochemical model, aging, vehicle thermal management

1 Introduction

The development and analysis of battery electric vehicles (BEVs) in the context of degradation presents multiple challenges. To begin with, the characterization of aging stress factors of commercial cells is a laborious and expensive task. Moreover, the real life operation of the battery packs, including the thermal management system (TMS) and battery management system (BMS), largely influence the way in which the individual cells of a battery pack will age.

Understanding the mechanisms by which the different functional components of a cell age is crucial. Safety can only be ensured if degradation is mitigated to the extent that batteries do not present hazardous behavior. Some relevant degradation mechanisms include: electrolyte decomposition, solid electrolyte interphase (SEI) growth and breakdown, gas evolution, overcharge, lithium plating, overdischarge, self-discharge, loss of active material, migration of soluble species, particle fracture, mechanical stress and structural changes (Arora, White, and Doyle 1998; Broussely 2002; Vetter et al. 2005; M. Broussely et al. 2005; Christensen and Newman 2006). Recent studies show that both SEI growth and lithium plating constitute the major drivers of aging under typical operating conditions (Ali et al. 2023; Che et al. 2023; Schlasza et al. 2014). Since battery degradation is caused by different stress factors, like high currents, extreme temperatures, or wide voltage windows, recommended limits have been developed to slow down or even avoid unnecessary damage (Fig. 1).

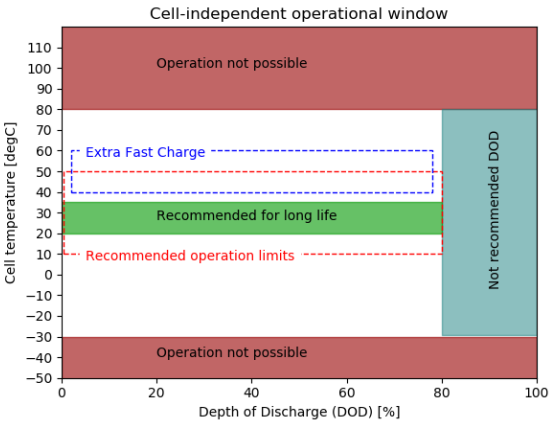


Figure 1. Summary of operational limits and recommendations for commonly used Li-ion cells (several sources)

Characterizing aging using mathematical expressions is needed to predict the likely degradation trajectory followed by the cell. Semi-empirical models have been proposed for this task as a default option because of its lower parametrization effort compared to physics-based models. The reader is referred to (Broussely et al. 2001; Ploehn, Ramadass, and White 2004; J. Wang et al. 2011; Rechkemmer et al. 2019) for different semi-empirical approaches. In general, however, these models can lump aging mechanisms unnecessarily, especially when more insight is needed in order to ensure a higher level of safety throughout the lifetime operation of the battery packs.

A novel physics-based aging model for Li-ion cells was described in (Yang, Leng, et al. 2017), that further enhances the electrochemical model previously developed by the authors (Gu and C. Wang 2000; Srinivasan and C.

Wang 2002). The aging model is based on the assumption that consumption of solvent and lithium plating (considered irreversible) contribute to the growth of the SEI layer, which in turns increases the film resistance and reduces the anode porosity. Since the aging model is applied on the so-called Newman's model (Marc Doyle, Fuller, and Newman 1993), the SEI growth and the appearance of lithium plating is spatially distributed. The model is reported to be effective capturing cyclic aging behavior observed experimentally.The approach is valid for the whole spectrum of C-rate and temperature when an Arrhenius dependency is added to the set of parameters (Yang and C.-Y. Wang 2018).

In a similar fashion (Li et al. 2019) adds an aging model on top of a reduced order electrochemical model, a further derivation of the Single Particle Model (SPM). This reduced order model assumes that the diffusion of lithium in the active material of the entire electrode can be represented by that of a characteristic spherical particle. The spatial gradient along the thickness of the electrodes, in the sense of concentration in solid and electrolyte phases, is therefore neglected. For a detailed description of the SPM and a discussion of the range of validity compared to the Newman's model, the reader is referred to (Chaturvedi et al. 2010).

Building upon these and more recent development in the area of Li-ion cells aging modeling, we have implemented a low complexity aging model that extends our previous reduced-order electrochemical mathematical description (Romero and Angerer 2023). The main feature of our contribution is the coupling of temperature sensitive aging dynamics, suitable for lithium plating side reactions in cyclic conditions (under load), together with calendaric (storage) degradation, all in a minimal implementation of the conservation laws in these kind of cells.

The remainder of the paper is organized as follows. First the electrochemical model with aging dynamics is presented. Next, we briefly describe the validation results for a particular cylindrical cell. A case study is subsequently introduced, and the result of the different operational alternatives analyzed. The last section is devoted to conclusions and future work.

2 Physics-based aging model implementation

2.1 Equivalent Hydraulic Model

The EHM has been used in the literature in order to capture electrochemical dynamics with the lowest mathematical complexity, in particular for control purposes (Romagnoli et al. 2017; Romero, Goldar, Couto, et al. 2019; Goldar et al. 2020). The previous application in the Modelica language was presented in (Romero, Goldar, and Garone 2019; Romero and Angerer 2023).

The EHM considers two electrochemical states, the bulk and surface concentration, in representative solid particles of the positive and negative electrodes. They are normalized with the maximum concentration ($c_{s,max}$) and denoted by SOC (state of charge) and CSC (critical surface concentration) respectively. If we assume that the intercalation dynamics is significantly faster in one of the electrodes (e.g. the positive), the SOC and CSC can be assumed identical and a linear combination of the bulk SOC of the other one (the negative). This assumption is needed for observability reasons(i.e., if the internal states need to be estimated from measured voltage and applied current, as it is normally the case) and can be justified depending on the chemistry and electrode design (Couto, Schorsch, et al. 2016). The model considers current density as input ($I = i/A_{cell}$, where A_{cell} is the electrode area, and i the electrical current, positive if discharge), which allows us to describe the dynamics independent of the cell size.

$$\frac{d\,\mathrm{SOC_n}}{dt} = -\gamma I \tag{1}$$

$$\frac{d\,\mathrm{CSC_n}}{dt} = \frac{g}{\beta(1-\beta)}\left(\mathrm{SOC_n} - \mathrm{CSC_n}\right) - \frac{\gamma}{1-\beta} I \tag{2}$$

$$\mathrm{SOC_p} = \rho\,\mathrm{SOC_n} + \sigma \tag{3}$$

$$\mathrm{CSC_p} = \mathrm{SOC_p} \tag{4}$$

$$V = U_p - U_n + \eta_p - \eta_n - \left(R_{cc} + \frac{R_{film}}{a\,l_n}\right) I \tag{5}$$

$$\eta_{p,n} = \frac{RT}{\alpha F} \sinh^{-1}\left(\frac{\theta_{p,n}\, I}{\sqrt{\mathrm{CSC_{p,n}}\left(1-\mathrm{CSC_{p,n}}\right)}}\right) \tag{6}$$

$$\gamma = \frac{3}{R_s\, a\, F\, l\, c_{s,max}} \qquad g = \frac{147}{20}\tau \qquad \beta = \frac{7}{10}$$

$$\tau = \frac{R_s^2}{D} \qquad a = 3\frac{\varepsilon}{R_s} \qquad \theta = \frac{1}{2\,a\,l\,r_{eff}\, c_{e0}^{1/2}\, c_{s,max}}$$

Equations 1-6, describe the dynamics of bulk and surface concentration, making use of lumped parameters γ, g, τ, and θ. They are used in the second-order Padé approximation (Couto, Schorsch, et al. 2016), and the overpotential η as defined in (Chaturvedi et al. 2010). The purpose of ρ and σ is to determine the static relationship between the state of charge of each electrode, which is a simplification of a more complex lithium dynamic balance. Their values (here -0.94 and 0.92 respectively) depend on the cell electrode design and formation process (Couto and Kinnaert 2018).

Open circuit voltages ($U_{p,n}$, dependent only on $\mathrm{CSC_{p,n}}$), overpotentials, and the internal resistances, are used to calculate the cell voltage. The previously so-called film resistance is now divided into pure ohmic losses (contact resistance, tabs and current collectors mainly, R_{cc}, and the SEI layer resistance R_{film}, time-dependent, which will be derived in the following section.

Exemplary parameters for a cylindrical 21700 Li-ion cell with a nominal capacity of 5 Ah are included in Table 1. Since the focus is on an EHM with fast dynamics

DOI 10.3384/ecp218841

in the cathode, only anode parameters are included. The values provided are temperature or concentration independent, except for r_{eff}, which is considered temperature dependent. The exact expression cannot be disclosed, but typically an Arrhenius relationship is used in the literature. Unlike our previous work, now the active material volume fraction (ε) changes in time as the battery ages. Accordingly, this variation represents the so called Loss of Active Material (LAM), and the details will be explained in the next section. For this reason, we considered ε_0 the initial value of the variable.

Table 1. Cell model parameters

Parameter	Units	Value
Electrode area, A_{cell}	[m^2]	0.1
Mass, m_{cell}	[kg]	0.07
Specific heat capacity, $C_{\text{p,cell}}$	[J/kg K]	800
Radial conductance, G_{rad}	[W/K]	1
Particle radius, R_{s}	[μm]	5
Electrode thickness, l	[μm]	80
Diffusion coefficient, D	[m^2/s]	1e-15
Active material vol. fraction, ε_0	%	75
Specific interfacial area, a	[m^2/m^3]	45e3
Effective reaction rate, r_{eff}	[$\frac{\text{A}}{\text{m}^2}(\frac{\text{m}^3}{\text{mol}})^{1.5}$]	7e-6
Maximum concentration $c_{\text{s,max}}$	[mol/m^3]	30e3
Electrolyte concentration, c_{e0}	[mol/m^3]	1200
Ohmic resistance, R_{cc}	[$\text{m}\Omega\,\text{m}^2$]	0.4

The thermal model assumes lumped properties concentrated on the cell core, typically the warmest region during normal operation. The thermal resistance consists of a serial summation of convection and conduction terms (external and internal heat flow respectively)

$$m_{\text{cell}} C_{\text{p,cell}} \frac{dT}{dt} = i(V - (U_{\text{p}} - U_{\text{n}}) + T\,(\frac{\partial U_{\text{p}}}{\partial T} - \frac{\partial U_{\text{n}}}{\partial T})) - \frac{1}{R_{\text{th}}}(T - T_{\text{amb}}) \quad (7)$$

$$A = \pi D H_{\text{c}} \quad R_{\text{th}} = (\frac{1}{4\pi H_{\text{c}} k} + \frac{\ln(D_{\text{can}}/D))}{2\pi H_{\text{c}} k_{\text{can}}} + \frac{1}{hA})$$

where R_{th} is the thermal resistance of the lumped thermal model of the cell, h is the heat transfer coefficient w.r.t. the cooled/heated section, and $\frac{\partial U_{\text{p(n)}}}{\partial T}$ define the so called entropic heat of the positive (negative) electrode (Dao and Schmitke 2015). We note here that this model approximates the behavior of an infinite cylinder with homogeneous heat generation. In reality, the active cooling/heating area is limited to a portion of the total height of the cell (assuming side cooling), while the rest is cooled passively by natural convection typically with the surrounding air. The cell housing conductivity is high enough, and its thickness sufficiently small, to neglect its contribution in the total resistance. The core to surface conductance in radial direction can be simplified as G_{rad}. The entropic heat, as well as the heat transfer to the air, can also considered to be negligible. Thus, the simplified model can be reduced to

$$m_{\text{cell}} C_{\text{p,cell}} \frac{dT}{dt} = i(V - (U_{\text{p}} - U_{\text{n}})) - UA\,(T - T_{\text{amb}}), \quad (8)$$

where U is the so called overall heat transfer coefficient, which includes internal conductance and external convection terms.

2.2 Aging model

The extension of the electrochemical model is based on the work of (Yang, Leng, et al. 2017). Our adaptation considers the main assumptions:

1. Main aging mechanisms are surface film (SEI) growth, irreversible lithium plating, and loss of active material
2. Negligible aging in cathode material
3. Homogeneous degradation throughout the electrode volume.

The first two assumptions are compatible with the model described in (Yang, Leng, et al. 2017), and the last one is required by the nature of the EHM. In their work, two states are needed to define the accumulated aging, namely the reacted ethylene-carbonate to form new SEI (c_{SEI}) and the lithium plated (c_{Li}). This accumulation in the form of molar concentrations (per unit volume of the electrode) can be represented in the following differential equations

$$\frac{dc_{\text{SEI}}}{dt} = -\frac{j_{\text{SEI}}}{2F} - \frac{j_{\text{lpl}}}{F}\chi \quad (9)$$

$$\frac{dc_{\text{Li}}}{dt} = -\frac{j_{\text{lpl}}}{F}(1 - \chi). \quad (10)$$

where χ is considered null as in the referenced work for simplicity. These two products form the so called SEI layer, a surface film whose thickness is proportional to their accumulation around the surface of the active material particles, or in mathematical form,

$$\delta_{\text{film}} = \frac{1}{a}(\frac{c_{\text{SEI}} M_{\text{SEI}}}{\rho_{\text{SEI}}} + \frac{c_{\text{Li}} M_{\text{Li}}}{\rho_{\text{Li}}}), \quad (11)$$

where $M_{\text{SEI(Li)}}$ and $\rho_{\text{SEI(Li)}}$ are the molar weight and density of each aging products. The thickness of the SEI layer defines in turns its resulting ionic resistance

$$R_{\text{film}} = \omega_{\text{SEI}} \frac{\delta_{\text{film}}}{K_{\text{SEI}}}, \quad (12)$$

with ω_{SEI},K_{SEI} being the volume fraction of SEI in the film and the ionic conductivity respectively .

At this point, we introduce the alternative approach with respect to the work of (Yang, Leng, et al. 2017). Instead of assuming that the drop in porosity dominates the aging behavior, caused by the clogging of the void fraction by the SEI growth, we focus on the loss of active material degradation mechanism. Since the EHM neglects the transport dynamics in the electrolyte, it is not directly affected by changes in porosity. In contrast, one could associate the SEI buildup to the progressive isolation of active material regions, causing an apparent loss of active material (aLAM). This approach is similar to that found in (Li et al. 2019), originally proposed by (Fu et al. 2015). Accordingly, we can define the loss of active material as the decrease in active material volume fraction (ε)

$$\frac{d\varepsilon}{dt} = -a\frac{d\delta_{\text{film}}}{dt}, \tag{13}$$

where it is explicit that the volume loss of accessible active material is proportional to the SEI volume increase. Equation 13 constitutes the derivative of the third aging state variable. This approach is similar to that followed by MapleSim's Battery Component Library (MapleSoft 2025), where the normalized remaining capacity depends on a cubic function of the SEI thickness. Therefore, our solution is a compromise between empirical simplicity and physical fidelity.

SEI growth is dominated by the concentration of ethylene-carbonate in the SEI layer and its thickness. The expressions found in (Yang, Leng, et al. 2017) were found to be suitable for our model in predicting cyclic aging, the original purpose of the model, but not calendaric aging. Therefore, a modified version of their work is presented here.

The volumetric current density associated to the SEI side reaction can be expressed as:

$$j_{\text{SEI}} = -a\,F\,k_{0,\text{SEI}}\,c_{\text{EC}}^{\text{s}} \exp\left(-\frac{\alpha_{\text{c,SEI}}\,F}{RT}\left(\phi_{\text{s}} - \phi_{\text{e}} - \frac{j_{\text{tot}}}{a}R_{\text{film}} - U_{\text{SEI}}\right)\right) \tag{14}$$

where j_{tot} is the current density, $k_{0,\text{SEI}}$ the reaction constant, and c_{EC}^{s} is the concentration of EC (ethylene carbonate) on the surface of the graphite (Yang, Leng, et al. 2017) or, more precisely, in the SEI layer (Zhao, Choe, and Kee 2018). According to the former, the solvent concentration can be computed with the following algebraic equation

$$-D_{\text{EC}}\frac{c_{\text{EC}}^{s} - c_{\text{EC}}^{0}}{\delta_{film}} = -\frac{j_{\text{SEI}}}{F} \tag{15}$$

which neglects the time dependency of the diffusion process.

The diffusion of EC is considered to be enhanced with temperature, as remarked by (Zhao, Choe, and Kee 2018) following the Arrhenius form

$$D_{\text{EC}} = D_{\text{EC,0}}\exp\left(\frac{E_{\text{a}}}{R}\left(\frac{1}{T} - \frac{1}{T_{\text{ref}}}\right)\right). \tag{16}$$

Most studies on physics-based aging models focus on cyclic aging, where these are fitted to match experimental results under cyclic loading. Research shows (Jin and Liu 2019) that it is possible to derive the calendaric aging based on the concentration of products and reactants in the SEI layer, an approach which may lead to a better performance of the model, both for cyclic and calendaric aging. We have opted for a hybrid approach, where the diffusion of EC during storage conditions depends on the state of charge and the temperature

$$D_{\text{EC}} = k_{\text{cal}}D_{\text{EC,0}}\,\text{SOC}\exp\left(\frac{-E_{\text{a}}}{R}\left(\frac{1}{T} - \frac{1}{T_{\text{ref}}}\right)\right), \tag{17}$$

where k_{cal} limits the increase of calendaric aging as the SOC tends to 100% compared with the cyclic aging. The proportionality with SOC allows to also increase the observed impact of high SOC storage, which may not be captured completely in Eq.(14). This expression has been developed to mimic the behavior of the calendaric aging of the cell considered, but adjustments for alternative chemistries are possible.

Lithium plating is a phenomenon that tends to take place in the interface between anode and separator, and the last assumption is only justified because the aim of the aging model is mainly to capture the cycle or point in time when lithium plating starts occurring. As indicated in the literature (Yang, Leng, et al. 2017), as the SEI grows, the likelihood of plating increases. Therefore, a charging cycle that does not trigger lithium plating in a fresh cell, may do so after certain time in operation. It will be shown, nonetheless, that a model with homogeneous lithium plating captures the capacity loss when it takes place reasonably well, causing the rapid deterioration of the electrode. According to (Yang, Leng, et al. 2017), the side reaction current associated to lithium plating can be formulated as

$$j_{\text{lpl}} = -a\,i_{0,\text{lpl}} \exp\left(-\frac{\alpha_{\text{c,lpl}}\,F}{RT}\left(\phi_{\text{s}} - \phi_{\text{e}} - \frac{j_{\text{tot}}}{a}R_{\text{film}} - U_{\text{lpl}}\right)\right) \tag{18}$$

where $i_{0,\text{lpl}}$ is the exchange current density of the lithium plating reaction. It should be noted that this side-reaction is active if the anode potential drops below the Li/Li^{+} equilibrium potential (U_{lpl}), and zero otherwise.

These equations and similar ones have been applied to electrochemical models of Li-ion cells, including SPM (Li et al. 2019). Since the EHM assumes constant electrolyte concentration, and it is therefore not modeled, the overpotential of reactions necessary to calculate the side reaction currents are expressed as follows:

$$\begin{aligned}\eta_{\text{SEI}} =& \phi_{\text{s}}^{-} - \phi_{\text{e}}^{-} + \frac{j_{\text{tot}}}{a^{-}}R_{\text{film}} - U_{\text{SEI}} = \\ & \eta^{-} + U^{-}(\text{C}_{\text{ss}}^{-}) - U_{\text{SEI}}\end{aligned} \tag{19}$$

DOI 10.3384/ecp218841

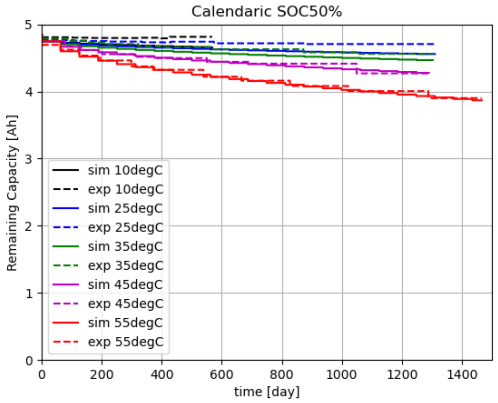
$$\eta_{\mathrm{lpl}} = \phi_{\mathrm{s}}^{-} - \phi_{\mathrm{e}}^{-} + \frac{j_{\mathrm{tot}}}{a^{-}} R_{\mathrm{film}} - U_{\mathrm{lpl}} = \eta^{-} + U^{-}(\mathrm{C}_{\mathrm{ss}}^{-}) - U_{\mathrm{lpl}} \tag{20}$$

In order to avoid side reactions, the overpotential of the considered side reaction should always be non-negative:

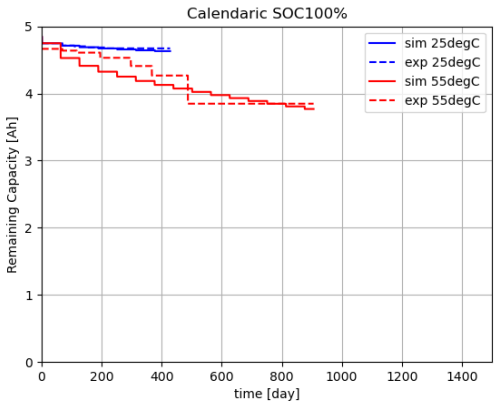
$$\eta_{\mathrm{SR}} = \eta + U(\mathrm{CSC}) - U_{\mathrm{SR}} \geq 0. \tag{21}$$

Table 2. Aging model parameters

Parameter	Units	Value
SEI growth reaction		
EC content in electrolyte, $c_{\mathrm{EC},0}$	[mol/m^2]	4.5e3
Diffusion of EC, D_{EC}	[m^2/s]	2e-15
Conductivity, κ_{SEI}	[S/m]	5e-6
Volume fraction, Ω_{SEI}	[-]	0.01
Activation energy, E_{a}	[J/mol]	1e4
Kinetic rate constant, $k_{0,\mathrm{SEI}}$	[m/s]	1e-12
Transport coefficient, $\alpha_{\mathrm{c,SEI}}$	[kg/m^3]	0.5
Molar mass, M_{SEI}	[kg/mol]	0.162
Density, ρ_{SEI}	[kg/m^3]	1690
Side reaction eq. potential, U_{SEI}	[V]	0.4
Lithium plating reaction		
Exchange current density, $i_{0,\mathrm{lpl}}$	[A/m^2]	1e-3
Transport coefficient, $\alpha_{\mathrm{c,lpl}}$	[kg/m^3]	0.5
Molar mass, M_{Li}	[kg/mol]	6.9e-3
Density, ρ_{Li}	[kg/m^3]	534
Side reaction eq. potential, U_{lpl}	[V]	0



(a) 50% SOC



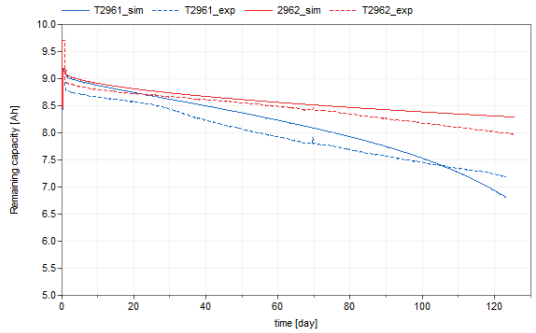
(b) 100% SOC

Figure 2. Calendaric aging validation results

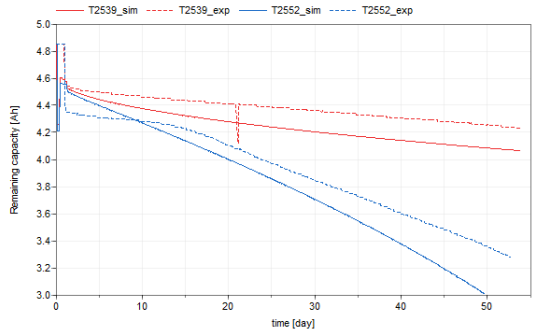
3 Model calibration

In the first place, the chemistry and the main parameters of the electrochemical model were determined. The cathode composition is high-niquel NMC, and the anode contains mainly graphite as active material, as external laboratory analysis indicate. The total capacity is 5 Ah approximately, and the format is the conventional cylindrical 21700 (21 mm diameter, 70 mm height). Parameters concerning geometry (electrode area and thickness, particle diameter) and weight where measured either in-house or by an external laboratory. The active material fraction was adjusted from a reference value borrowed from the literature (Chen et al. 2020). Thermal behavior ($C_{\mathrm{p,cell}}$, G_{rad}) could be adjusted from a large variety of input-output data from experimental tests under different boundary conditions (free convection at room temperature, controlled temperature chambers with forced air flow, and immersion cooling). The pure ohmic resistance was determined from ACIR (Alternating Current Internal Resistance at high frequency). Diffusion coefficients in the solid phase, as well as the remaining electrochemical properties (including $U_{\mathrm{p,n}}$) were adjusted from different literature sources or external laboratories. Values reported in Table 1 must be taken as exemplary, since the actual values are are kept confidential.

The required cyclic and calendaric aging tests have been conducted in-house to adjust the parameters of the aging model. Exemplary values are given in Table 2 since the actual ones cannot be disclosed. In addition, literature data and sensitivity analyses have been used to best characterize the cyclic and calendaric aging behavior. More specifically, U_{SEI} is taken from (Zhao, Choe, and Kee 2018), while the remaining parameters were taken directly from (Yang, Leng, et al. 2017), or adapted from this source (mainly D_{EC}, Ω_{SEI}). The activation energy was directly fitted from experimental data. Different temperature and SOC storage conditions were used in the storage tests (Fig. 2), while different current and temperature values were used under cycling, with emphasis on the corner cases where lithium plating was observed (Fig. 3). The observation involved a non-linear, accelerated capacity drop, as well as *post mortem* tear-down examinations. In particular, tests with CCCV charges 0.7C and 1C were chosen at temperatures below which, according to the manufacturer, lithium plating may occur (25°C and 35°C respectively).



(a) 0.7C charge / 1.5C discharge (two parallel cells, blue at 19/26°C, red at 26/31°C)



(b) 1C charge / 1.5C discharge (one single cell, blue at 22°C, red at 40°C)

Figure 3. Cyclic aging validation results at temperatures below and above the lithium plating recommended temperature

4 Case study

In order to highlight the benefits of the model presented above, we conducted a set of tests with varying operating conditions that affect the rates of calendaric and cyclic aging. The target cell corresponds to cylindrical cell similar to that described in previous sections. The full description of the pack, which uses immersion cooling in direct contact with the cells can be found in (Romero and Angerer 2023). Its size (60 kWh) is similar to that of a pack for mid-size battery electric vehicles. Only the behavior of a single cell is analyzed here, as though all cells would operate under the same boundary conditions. In reality, the resulting temperature spread within the pack will lead to heterogeneous aging, but this is out of the scope of this work.

All models described in this paper have been implemented in Dymola, but compatibility with other Modelica tools, i.e., OpenModelica, was ensured. Before describing the boundary conditions and the case study, we introduce two key components included in the case study simulation template.

4.1 Thermal control

Electric vehicles use a sophisticated thermal management system, with control algorithms set to ensure an optimal operating temperature window. In our application, we define a hysteresis loop with immersion cooling (ON: 30°C, OFF:25°C) and heating (ON: 10°C, OFF:15°C) that ensures practical operational temperatures. Sufficient thermal power is assumed, with a fixed temperature source of 30°C when heating and 25°C when cooling. In addition to the active cooling boundary, the cell model is thermally connected to the environmental temperature. The active thermal management guarantees around 100 W/m^{2}K of heat transfer. A convective overall heat transfer coefficient to the ambient of 1 W/m^{2}K is assumed. Both values have been experimentally determined for the pack considered where the cells are installed, and approximate values are reported.

4.2 Cycler

Modelica classes have been developed to automate, in a virtual environment, the classical laboratory performance and cyclic aging. This includes lifetime simulations, where a rich set of mixed protocols are linked sequentially. This includes, e.g., CCCV charges and discharges, duty cycles, capacity check cycles, and DCIR tests. The `ModelicaStateGraph2` library (Otter, Årzén, and Dressler 2005) is used to control the cycler states, but the building philosophy could be adapted to other similar libraries. An example of a custom cycler with profile input during the discharge state (without opportunity charge) is shown in Fig. 4. It must be noted that current, voltage or power are assigned in the equation section in the text layer (not shown here). The cycler variant used assumes eight different states or steps:

1. Initial step
2. Pre-charge
3. Constant current charge
4. Constant voltage charge
5. Idle time after charge
6. Pre-discharge
7. Duty-cycle
8. Pre-opportunity charge
9. Opportunity charge
10. Idle time after discharge.

Every day, a number of duty-cycles are repeated before and after a mid-day break, which considers a typical daily commute, including weekend leisure usage. Charging is allowed during nights of certain days (Wednesday, Saturday and Sunday) at a C-rate of 0.1C (w.r.t. nominal energy, 18 Wh), but opportunity charging below certain voltage is also possible (0.7C w.r.t. nominal capacity, 5 Ah). The effect on aging of these two charging possibilities is compared below.

The automotive driving profile was provided by Centro Ricerche FIAT (CRF) at cell level. The final power was re-scaled for an equivalent battery size of 50 kWh. The profile represents a typical WLTC with a duration of approximately 23 km and 1800 s. In the first option (light usage), we consider it is repeated once (1xWLTC) in the morning and once in the afternoon. In a second option (moderate usage), we increase to 2xWLTC, morning and

DOI 10.3384/ecp218841

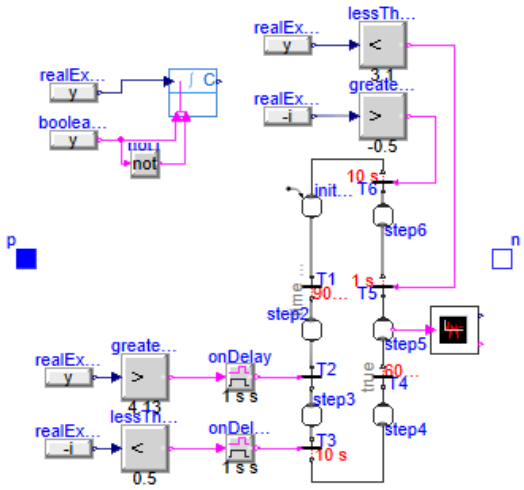


Figure 4. Simple cycler with power profile discharge using `ModelicaStateGraph2`

afternoon. Each of these options relate to vehicles driving around 150 and 300 thousand kilometers within ten years respectively.

4.3 Additional boundary conditions

Two locations are considered in our study, namely Seville and Gothenburg. They present two extreme climatic options, represented by their ambient temperature (Fig. 5).

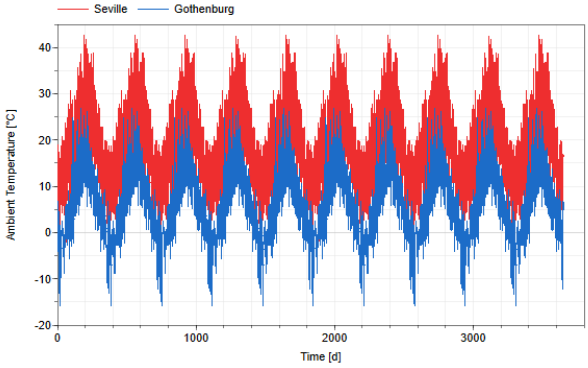


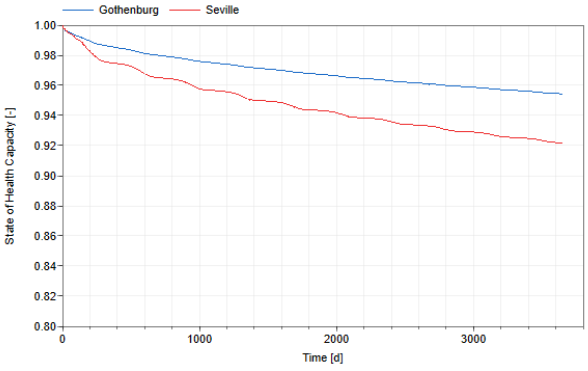
Figure 5. Local ambient temperatures (dry-bulb). Source: https://pvwatts.nrel.gov/

5 Results

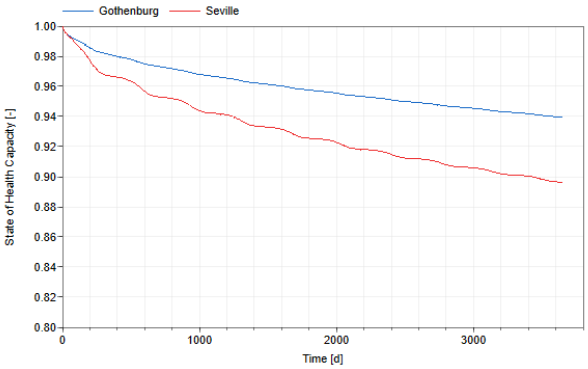
5.1 Night charge

We first compare different levels of usage of the battery in Seville and Gothenburg for the night charge condition. Results are shown in Fig. 6 including light and moderate usage and default thermal management, as well as moderate usage and higher temperature cooling.

Figure 7 present results for simulations where thermal control for cooling is limited to an hysteresis loop between 40°C (ON) and 35°C(OFF). This setup affects mainly Seville, since Gothenburg's temperatures seldom reach 35°C.



(a) 2xWLTC per day



(b) 4xWLTC per day

Figure 6. Comparison between the effect of number of cycles per day and ambient temperature

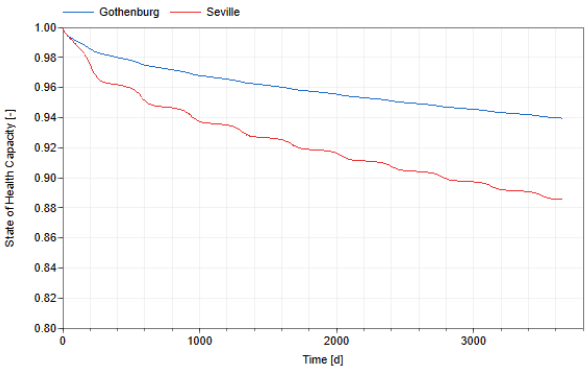


Figure 7. 4xWLTC per day, two locations, higher cooling temperatures

For the case in which the lower temperature is not sufficiently limited by the battery heater (i.e., hysteresis setup 5°C(ON), 10°C(OFF)), the battery may charge or regenerate energy in conditions close to lithium plating. We have also considered the fact that the thermal management is not active during the idling period, which exacerbates the impact of low temperatures (Fig. 8).

For comparison, Fig. 9 illustrates the effect of each charging strategy on the voltage during two consecutive weeks. It can be seen that in both cases the anode potential drops below the lithium plating threshold.

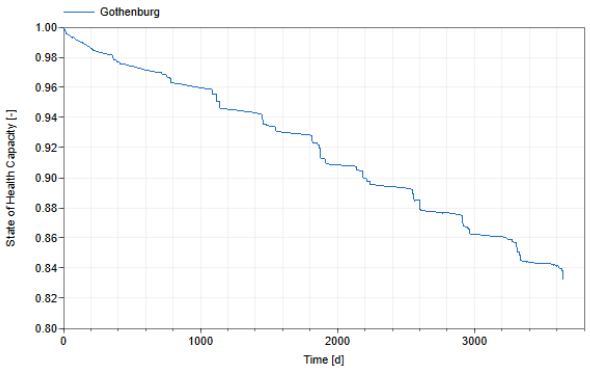


Figure 8. 4xWLTC per day, Gothenburg, lower heating temperature

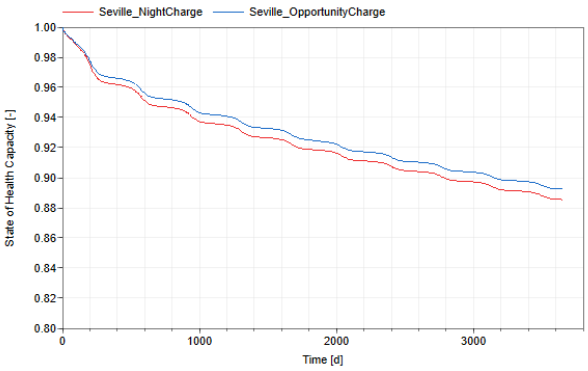


Figure 10. Remaining capacity results considering night and opportunity charge options

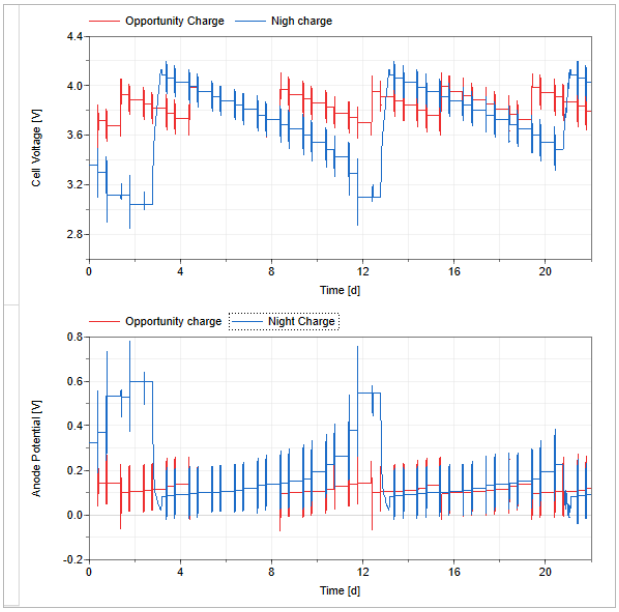


Figure 9. Cell voltage and anode potential considering night and opportunity charge options

5.2 Frequent charges at medium SOC

Finally, we present the results for the case in which the battery is allowed to recharge on-demand if the cell voltage drops below 3.6V, which leads to a lower DOD at lower SOC as in the previous case. As shown in Fig. 10, the opportunity charge can extend the life of the battery by over 400 days compared to night charge for the same SOH. This is so because the cell remains at a higher SOC for longer time than in the opportunity-charge strategy.

To minimize aging, as the literature indicates and we have seen in simulation, charging voltages and DOD should be selected as a function of the available charging locations and times, the daily usage, etc. We have discussed two cities with opposite climatic conditions. It is expected that locations with milder temperatures will require less energy for the thermal management (heater, chiller, etc.) and will result in intermediate SOH trajectories, assuming that the protection against plating is ensured.

6 Conclusions and future work

Aging of lithium ion batteries is a complex and challenging research area, and affects the safety and economics of electrification. This is of special importance in transportation electrification, where the multi-physics approach allows us to explore the effect of driving patterns, charging strategies, weather conditions and, more importantly, the particular aging dynamics of every cell chemistry.

Since full order electrochemical models with aging are computationally expensive and difficult to parameterize, we propose a reduced order model with SEI growth and lithium plating dynamics. The fitting effort is lower for the typical set of calendaric and cyclic tests, and conservative margins can be added to deal with parameter uncertainty and model limitations to better describe operational limits. As an example, we simulated different charging and thermal management strategies. Results show that lower temperatures and more frequent charging that reduce DOD and average SOC are beneficial, provided that anode potentials can be kept above the lithium plating onset.

Ongoing efforts are focused on improving the fidelity of the EHM, which involves an extension to cover electrolyte concentration dynamics (EHMe), as well as a P2D version with minimal discretization complexity to compare the influence of model inaccuracies. Additionally, experimental tests are currently being conducted to verify the simulated results presented.

Acknowledgements

This work has been developed within DigiCell, a project funded by the European Union's Horizon research and innovation action under grant agreement No. 101135486.

References

Ali, Hayder et al. (2023). "Assessment of the calendar aging of lithium-ion batteries for a long-term—Space missions". In: *Frontiers in Energy Research* 11, p. 1108269.

Arora, P, RE White, and M Doyle (1998). "Capacity fade mechanisms and side reactions in lithium-ion batteries." In: *Journal of The Electrochemical Society* 145.10, pp. 3647–3667.

Broussely, M (2002). "Aging Mechanisms and Calendar-Life Predictions in Lithium-Ion Batteries". In: *Advances in Lithium-Ion Batteries*. Ed. by WA van Schalkwijk and B Scrosati. 1st. New York: Springer US. Chap. 13, pp. 393–432.

Broussely, M et al. (2001). "Aging mechanism in Li ion cells and calendar life predictions." In: *Journal of Power Sources* 97-98, pp. 13–21.

Broussely, M. et al. (2005). "Main aging mechanisms in Li ion batteries". In: *Journal of Power Sources* 146.1-2, pp. 90–96.

Chaturvedi, Nalin A et al. (2010). "Algorithms for advanced battery-management systems". In: *IEEE Control systems magazine* 30.3, pp. 49–68.

Che, Yunhong et al. (2023). "Health prognostics for lithium-ion batteries: mechanisms, methods, and prospects". In: *Energy & Environmental Science* 16.2, pp. 338–371.

Chen, Chang-Hui et al. (2020). "Development of experimental techniques for parameterization of multi-scale lithium-ion battery models". In: *Journal of The Electrochemical Society* 167.8, p. 080534.

Christensen, John and John Newman (2006). "Stress generation and fracture in lithium insertion materials". In: *Journal of Solid State Electrochemistry* 10.5, pp. 293–319. arXiv: 0006334v1 [arXiv:cond-mat].

Couto, Luis D. and Michel Kinnaert (2018). "Internal and Sensor Fault Detection and Isolation for Li-ion Batteries". In: *IFAC-PapersOnLine* 51.24, pp. 1431–1438.

Couto, Luis D., Julien Schorsch, et al. (2016). "SOC and SOH estimation for Li-ion batteries based on an equivalent hydraulic model. Part I: SOC and surface concentration estimation". In: *2016 American Control Conference (ACC)*, pp. 4022–4028.

Dao, Thanh-Son and Chad Schmitke (2015). "Developing Mathematical Models of Batteries in Modelica for Energy Storage Applications". In: *Proceedings of the 11th International Modelica Conference, Versailles, France, September 21-23, 2015*. 118. Linköping University Electronic Press, pp. 469–477.

Doyle, Marc, Thomas F Fuller, and John Newman (1993). "Modeling of galvanostatic charge and discharge of the lithium/polymer/insertion cell". In: *Journal of the Electrochemical society* 140.6, p. 1526.

Fu, Rujian et al. (2015). "Development of a physics-based degradation model for lithium ion polymer batteries considering side reactions". In: *Journal of Power Sources* 278, pp. 506–521.

Goldar, Alejandro et al. (2020). "MPC strategies based on the equivalent hydraulic model for the fast charge of commercial Li-ion batteries". In: *Computers & Chemical Engineering* 141, p. 107010.

Gu, WB and CY Wang (2000). "Thermal-electrochemical modeling of battery systems". In: *Journal of The Electrochemical Society* 147.8, p. 2910.

Jin, Xing and Chang Liu (2019). "Physics-based control-oriented reduced-order degradation model for LiNiMnCoO2 - graphite cell". In: *Electrochimica Acta* 312, pp. 188–201.

Li, Yang et al. (2019). "Development of a degradation-conscious physics-based lithium-ion battery model for use in power system planning studies". In: *Applied Energy* 248, pp. 512–525.

MapleSoft (2025). *MapleSim Battery Component Library*. https://www.maplesoft.com/support/help/MapleSim/view.aspx?path=batteryComponentLibrary/Electrochem/LiIon&cid=1778. Accessed: 2025-04-23.

Otter, Martin, Karl-Erik Årzén, and Isolde Dressler (2005). "Stategraph-a modelica library for hierarchical state machines". In: *Modelica 2005 proceedings*. Halmstad University, pp. 569–578.

Ploehn, HJ, P Ramadass, and RE White (2004). "Solvent Diffusion Model for Aging of Lithium-Ion Battery Cells." In: *Journal of The Electrochemical Society* 151.3, A456–A462.

Rechkemmer, Sabrina Kathrin et al. (2019). "Empirical Li-ion aging model derived from single particle model". In: *Journal of Energy Storage* 21, pp. 773–786.

Romagnoli, Raffaele et al. (2017). "Computationally-efficient constrained control of the state-of-charge of a li-ion battery cell". In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, pp. 1433–1439.

Romero, Alberto and Johannes Angerer (2023). "Fast Charge Algorithm Development for Battery Packs under Electrochemical and Thermal Constraints with JModelica. org". In: *Proceedings of the 15th International Modelica Conference, Aachen, Germany, 2023*, pp. 121–130.

Romero, Alberto, Alejandro Goldar, Luis D. Couto, et al. (2019). "Fast charge of Li-ion batteries using a two-layer distributed MPC with electro-chemical and thermal constraints". In: *18th European Control Conference (ECC)*. IEEE, pp. 1796–1803.

Romero, Alberto, Alejandro Goldar, and Emanuele Garone (2019). "A model predictive control application for a constrained fast charge of lithium-ion batteries". In: *Proceedings of the 13th International Modelica Conference, Regensburg, Germany, March 4–6, 2019*. 157. Linköping University Electronic Press.

Schlasza, Christian et al. (2014). "Review on the aging mechanisms in Li-ion batteries for electric vehicles based on the FMEA method". In: *2014 IEEE Transportation Electrification Conference and Expo (ITEC)*. IEEE, pp. 1–6.

Srinivasan, Venkat and CY Wang (2002). "Analysis of electrochemical and thermal behavior of Li-ion cells". In: *Journal of The Electrochemical Society* 150.1, A98.

Vetter, J et al. (2005). "Ageing mechanisms in lithium-ion batteries". In: *Journal of Power Sources* 147.1-2, pp. 269–281.

Wang, John et al. (2011). "Cycle-life model for graphite-LiFePO4 cells". In: *Journal of Power Sources* 196.8, pp. 3942–3948.

Yang, Xiao-Guang, Yongjun Leng, et al. (2017). "Modeling of lithium plating induced aging of lithium-ion batteries: Transition from linear to nonlinear aging". In: *Journal of Power Sources* 360, pp. 28–40.

Yang, Xiao-Guang and Chao-Yang Wang (2018). "Understanding the trilemma of fast charging, energy density and cycle life of lithium-ion batteries". In: *Journal of Power Sources* 402, pp. 489–498.

Zhao, Yinyin, Song-Yul Choe, and Jungdo Kee (2018). "Modeling of degradation effects and its integration into electrochemical reduced order model for Li(MnNiCo)O2/Graphite polymer battery for real time applications". In: *Electrochimica Acta* 270, pp. 440–452.

DOI
10.3384/ecp218841

Integrating a seasonal thermal energy storage FMU in a MATLAB/Simscape thermal source network model

Simon Müller[1,*], Abdulrahman Dahash[2], Shariq Akbar[1], David Schmitt[1], Peter Bayer[3], Tobias Schrag[1]

[1]Institute for new Energy Systems, University of Applied Sciences Ingolstadt, Esplanade 10, 85049 Ingolstadt, Germany

[2]Austrian Institute of Technology GmbH, Vienna, Austria

[3]Department of Applied Geology, Martin Luther University Halle-Wittenberg, Von-Seckendorff-Platz 3, 06120 Halle, Germany

*Corresponding author e-mail: simon.mueller@thi.de

Abstract

The integration of Functional Mock-up Units (FMUs) enables effective multi-domain co-simulation and model exchange, offering new possibilities for energy systems modelling considering thermo-hydraulic aspects. This paper presents a case study on incorporating an FMU of a seasonal thermal energy storage (sTES), developed in Modelica/Dymola, into a MATLAB/Simscape model of a thermal source network (TSN). While traditional thermo-hydraulic models are typically confined to a single simulation environment, this study demonstrates the feasibility and advantages of FMU-based interoperability between Modelica/Dymola and MATLAB/Simscape.

A key focus is the methodological approach for integrating FMUs into Simscape, addressing interface requirements, solver settings, and model coupling challenges. To illustrate the impact of FMU integration, the authors present simulation results for exemplary operation scenarios: TSN operation with and without an sTES. Additionally, our simulation results offer valuable insights into the influence of sTES on TSN operation and system layout.

Keywords: Functional Mock-up Unit (FMU), Thermal Source Network (TSN), Seasonal Thermal Energy Storage (sTES), MATLAB/Simscape, Modelica/Dymola, Thermo-Hydraulic Modelling

1 Introduction

The ongoing development of heating and cooling networks, such as district heating systems, requires constant integration of emerging technologies and components. Carefully planning these evolving systems demands flexible modelling approaches, which can be challenging when specialized models for new components reside in different software environments. For instance, an advanced sTES model might be developed in Modelica, whereas other system elements or advanced control libraries might only be available in MATLAB/Simscape. In many research and industrial settings, software licensing constraints further limit the ability to adopt multiple environments simultaneously, constraining researchers and engineers to a single "in-house" simulation tool.

To overcome this shortcoming, FMUs have recently gained traction as an enabling technology for sharing tailored simulation models across diverse platforms. By exporting a component from one domain and importing it into another, FMUs allow modelers to exploit specialized models without having to re-produce or re-validate them. This interoperability is particularly valuable in modelling of thermal source networks, where accurate representations of devices such as sTES units are essential for analysing and optimizing network performance.

This paper aims to demonstrate the feasibility of integrating a sTES model, originally developed in Modelica, into a TSN model built in MATLAB/Simscape using the Functional Mock-up Interface (FMI) standard. This approach enables the reuse of a validated sTES model, thereby avoiding the need for reimplementation and validation in Simscape. The study focuses on two core aspects of feasibility: (1) the technical effort required to set up the interface and configure the inputs and outputs, and (2) the stability and computational efficiency of the resulting co-simulation. Two realistic operational scenarios are evaluated as use cases: first, TSN operation without an sTES, and second, TSN operation where the sTES is coupled via a heat exchanger. Overall, this study shows how FMU-based co-simulation can increase the flexibility of thermo-hydraulic modelling across tools and domains, reducing software-related barriers in academic and industrial contexts.

2 Background and Related Work

The planning and design of sTES systems is a highly interconnected and complex process, requiring the evaluation of multiple parameters such as volume, geometry, insulation quality, and system integration (Dahash et al. 2019b). To mitigate risks associated with real-world implementation and to ensure feasibility and

effectiveness, modelling and simulation have become integral to sTES system planning. Moreover, energy system simulations are well-suited for evaluating integration strategies and control approaches that optimize system operation. Commonly used energy system simulation (ESS) tools include TRNSYS, EnergyPlus, Modelica/Dymola, MATLAB/Simulink, and Simscape.

Among these, TRNSYS has been widely adopted for modelling buried water-based TES, owing to its modular structure and availability of specific models like XST and ICEPIT, which facilitate coupling with buildings, heating plants, and other system-level components. For instance, (Li et al. 2019) used the Type 534 model to simulate a 3000 m³ pit sTES in a solar-assisted district heating system, evaluating different control strategies.

On a bright note, other environments have also explored buried sTES modelling. For example, (Dahash et al. 2020) developed and validated a buried tank sTES model in Modelica/Dymola, coupled with a ground model. However, this work was limited to component-level simulations and did not address the broader energy system context. Similarly, (Bott et al. 2022) developed an underground sTES model in MATLAB/Simscape, benefiting from its multi-domain capabilities. Yet, this model also focused exclusively on component-level analysis.

Consequently, validated sTES models integrated into full energy system simulations remain rare. In this context, co-simulation emerges as a promising solution, allowing the integration of component-level sTES models into broader system-level simulations. (Lyden et al. 2022) highlighted this gap, especially for borehole and aquifer TES models, while (Dahash et al. 2019b) noted a similar lack for pit and tank TES technologies.

To address this gap, the present work focuses on two key objectives:

1) Co-simulation of sTES and energy systems through the integration of an sTES Functional Mock-up Unit (FMU) into a top-level model of a real-world energy system, and
2) Optimization of system operation by evaluating the impact of sTES integration compared to a reference case without sTES.

3 Description of the Thermal Source Network

The TSN at the incampus site in Ingolstadt, Germany, serves as the case study in this work.

3.1 Overview of the Case Study

The TSN at the incampus in Ingolstadt (DE), is a pioneering energy system designed to facilitate CO_2-neutral industrial energy supply. The system's core principles include sustainability, modularity, and long-term viability (Müller et al. 2024). Due to the lower system temperature, the TSN offers a significant flexibility in integrating renewable and low-temperature energy sources, unlike conventional district heating systems. Key components in this system approach are decentralized heat pumps, which enable each building to extract or reject heat from/into the system independently from the TSN temperature level.

The incampus TSN undergoes development in multiple construction phases, eventually spanning 9 km of piping and connecting approximately 70 buildings, including:

- Offices, laboratories, and test benches, each with their own decentralized heat pump.
- A data centre acting as a major source of waste heat.
- Energy storage and distribution components, including an sTES and firefighting water tanks repurposed as thermal buffers.

A distinguishing feature of this TSN is that its pipes are uninsulated, allowing for thermal interaction with the surrounding ground. Unlike conventional district heating, where insulation is used to prevent energy loss, this TSN leverages its thermal environment as part of the storage and distribution system, potentially gaining heat or dissipating excess energy depending on the operational strategy. This can be attributed to the temperature levels foreseen in this TSN.

3.2 Energy Flows and System Operation

The TSN operates by balancing local, renewable, and waste heat sources with building-level heating and cooling demands. Unlike conventional heating networks, it functions bidirectionally, meaning buildings can both consume and supply heat leading to the so-called "prosumers." For this, each building is equipped with a decentralized heat pump, which allows it to either extract heat from the TSN for heating or reject excess heat into the network for cooling when operating in reverse operation.

The key energy sources include:

- Waste heat from the data centre (initially 1.83 MW, expandable to 3 MW), a stable and continuous source of low temperature heat in in the range of 18 to 38°C.
- Groundwater wells (1 MW thermal power), originally installed for remediation purposes and later adapted for thermal energy exchange.
- The Danube River (up to 10 MW thermal potential) as an additional cooling and heating source.
- Re-cooling plants (6 MW cooling capacity) for peak load scenarios.
- A seasonal thermal energy storage, designed as a water-gravel pit storage, re-purposing existing basin infrastructure, which reduces investment costs.

The primary objective of the system's control strategy is to balance and manage energy flows dynamically while minimizing operational costs, which for example depend on thermal losses or electricity consumption of heat

DOI 10.3384/ecp218851

pumps. The primary challenge in operating the TSN is finding the optimal strategy for system-wide efficiency. Accordingly, decisions shall be made regarding:

- The optimal TSN supply temperature levels to minimize losses while maximizing heat pump efficiency.
- The selection of thermal sources at different times considering cost, availability of renewable energy or other factors.
- The charging and discharging strategy of the sTES.

3.3 Motivation for FMU-Based Modelling of the TSN

To evaluate and validate different control strategies for the described system, a physical simulation model is a valuable tool. In general, FMUs can be utilized to integrate advanced, pre-existing simulation models of key components, which gives greater flexibility in testing and validation. Herein, this approach also might allow for faster and more accurate simulations, as an existing state-of-the-art sTES model developed in Modelica was directly incorporated rather than recreated in MATLAB/Simscape.

4 MATLAB/Simscape Model Development

4.1 Overall Model Structure

The MATLAB/Simscape implementation is encapsulated in a single .slx file, referencing a custom library to emulate the incampus with four main categories: Energy Hub, Pipes, Substations/Heat Pumps, and Buildings/ Loads.

All basic components which are used in the main subsystems have been taken from the two Simscape libraries *Fluids/Thermal Liquid* and *Foundation Library/Thermal.* The Simscape Toolbox is a MATLAB/Simulink add-on with a physically based, acausal modelling approach. It differs from the causal MATLAB/Simulink approach but is similar to the philosophy of Modelica/Dymola or other tools. This structure automatically derives the governing equations from the topology of the system, making it well suited to multi-domain energy systems. However, this approach has rarely been explored in Simscape (Dahash et al. 2019a).

In the base model, the sTES FMU is not yet included. However, an additional heat exchanger is already integrated into the Energy Hub to facilitate future FMU connections. Specific interface details are provided in Section 5.2 FMU integration.

4.2 Sub-Model Groups and Assumptions

Energy Hub

The Energy Hub comprises separate Simscape blocks for its centrifugal pump, heat exchangers, flow-coefficient valves, and a buffer tank (see Figure 1):

- **Pump:** A variable-speed centrifugal pump, parameterized with reference head/capacity curves, maintains the required volume flow in the TSN.
- **Heat Exchangers:** Two standard Simscape exchanger blocks connect the TSN to data centre waste heat and groundwater wells. A third exchanger is reserved for integrating the sTES FMU. Other heat exchangers to additional thermal sources were omitted.
- **Valves:** Simple flow-coefficient valves direct water through the appropriate heat exchanger or buffer tank, depending on an on/off control signal.

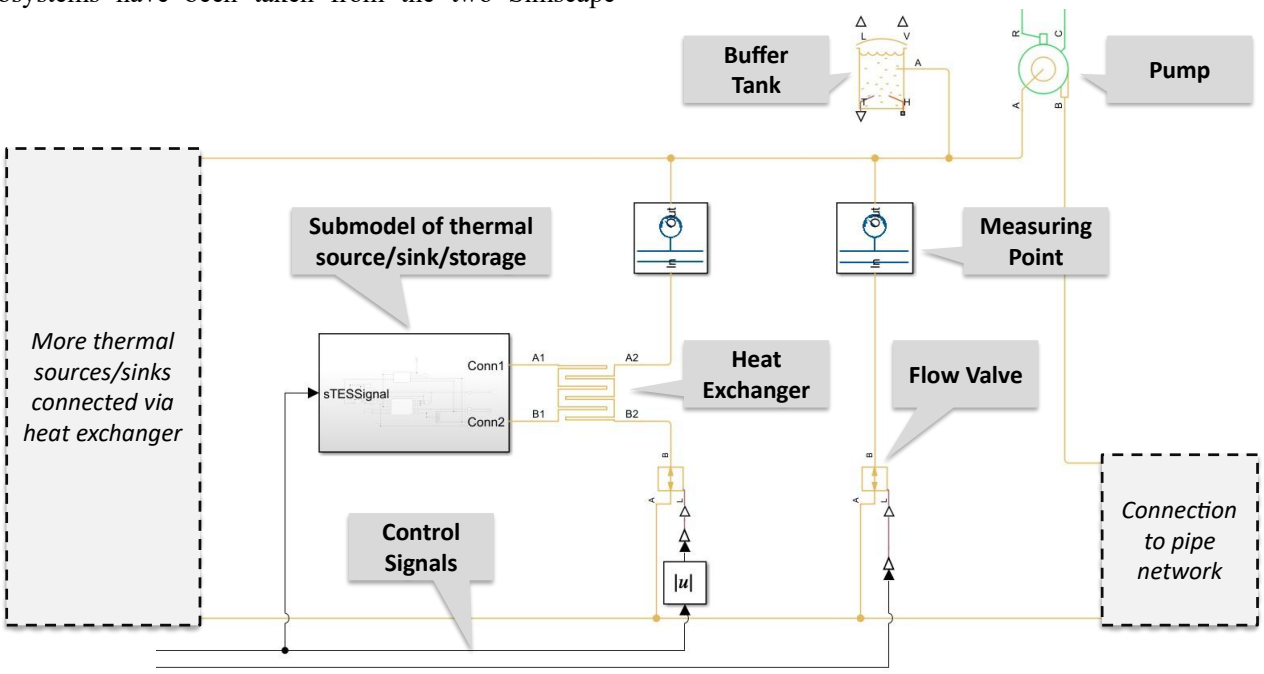


Figure 1. Components and Connections of the Energy Hub Sub-Model.

- **Buffer Tank:** Mainly accommodates thermal expansion (i.e. expansion vessel) in the simulated scenarios.
- **Measurement and Control:** The Hub's interfaces with the TSN piping and the thermal sources are monitored by thermal/hydraulic sensors (flow, pressure, temperature).

Pipes

The supply and return lines of the TSN are each represented by *Pipe (TL)* blocks from the Simscape *Fluids/Thermal Liquid* library, sized according to the incampus layout.

These blocks represent:

- Pressure Drop due to viscous friction.
- Heat Transfer with a seasonally varying ground temperature, using a fixed thermal resistance for the pipe wall.

To capture changes in diameter and branching points for buildings, the network is divided into multiple pipe segments. Thermal capacities of the pipe wall and ground are herein neglected. Flow is governed by the central pump in the Energy Hub, with no additional regulation in the pipes themselves. Pressure and temperature sensors placed at key nodes monitor conditions for system control and analysis.

Substations/Heat pumps

Each building in the TSN is equipped with a single heat pump block, which interfaces between the TSN and the building's heating and cooling circuits. Physically, this design yields six connection ports: supply and return lines for the TSN, and separate supply and return lines for both the heating and cooling circuits on the building side, all in the thermal liquid domain. In practice, the heat pump acts as a mediator, drawing energy from (or rejecting energy to) the TSN as required to satisfy the building's thermal demands.

Within the heat pump model, a Carnot-based approach is used to capture energy flows rather than simulating the full refrigeration cycle. In particular, the model measures the evaporator and condenser temperatures to calculate the theoretical Carnot efficiency, which is then scaled by a Carnot effectiveness factor (typically 0.4 to 0.5 for water-to-water heat pump units). This simplification emerges as a balance between realism and computational efficiency by focusing on overall heat exchange and compressor power, rather than diving into detailed phase-change processes and fluid properties.

From an operational perspective, the heat pump is capable of switching between four distinct modes: 1) heating (with the TSN serving as the heat source), 2) free cooling (where no compressor action is required and the TSN serves as the heat sink), 3) mechanical cooling (with the compressor actively dissipating heat into the TSN via the condenser), and 4) dual operation (with simultaneous building heating and cooling, using the TSN to cover any unbalanced demand).

The operation of these modes is governed by a local control scheme focusing on target supply temperatures in the building's heating and cooling circuits, along with the mass flow present in each loop. At instances where flow exceeds a pre-determined threshold, the heat pump starts automatically, with the objective of maintaining the selected supply temperature. The maximum power output is then limited by a rated thermal capacity. On the TSN side, the reference mass flow is calculated with respect to a prescribed temperature difference between TSN supply and return.

The heat pump model logs thermal energy flows and mass flow rates within the connected circuits, while alongside providing the COP and compressor power consumption as outputs.

Buildings/Loads

Currently, the incampus area is still under development and only two buildings from construction phase one are connected to the TSN (next to the Energy Hub and the Data Centre). Thus, this work simulates a future scenario where two more buildings are connected to the TSN.

The load profiles of the buildings are derived from planning documentation for the incampus energy system. This report provided the total floor area of each planned building, along with its annual heating and cooling energy demands (expressed in kWh/m²a) and peak thermal loads (in kW/m²). By virtue of this data, hourly heating and cooling profiles can be generated for each building and implemented as input signals for the heat flow sources.

Each building model comprises two separate circuits: one for heating and one for cooling. These circuits are each connected to an individual buffer tank, resulting in two tanks per building. Both tanks are connected to the heat pump's supply and return lines in the thermal liquid domain, while a thermal port on each tank interfaces with a controlled heat flow rate source. This source applies the building's time-varying thermal load, based on pre-defined demand profiles.

A key assumption in the building model is the exclusion of a specific thermal node or building capacity. The thermal inertia (i.e. thermal mass) of the buildings is implicitly included in the demand profiles, eliminating the need for detailed physical modelling of the building envelope. The tank serves to bridge the gap between the thermo-hydraulic domain of the heat pump and the demand-driven thermal domain of the load profiles. It is the interface point, facilitating heat exchange between the heat pump and the imposed thermal load. Each buffer tank is sized according to planning specifications (e.g. 1.8 m³ for each tank in the functional building).

To maintain appropriate supply conditions, a thermostat logic governs each tank. In the heating circuit, the tank can be heated up to a maximum of 45°C and allowed to cool to 35°C. In the cooling circuit, the tank is cooled down to 14°C and permitted to warm up to 20°C.

DOI 10.3384/ecp218851

5 Functional Mock-up Units: Creation and Integration

5.1 FMU creation

The sTES FMU is based on a stratified thermal energy storage model integrated with thermo-hydraulic components – valves, pumps, and source/sink elements – from the Modelica Buildings library, enabling dynamic charging and discharging during simulations. The core sTES model is developed using the *PartialStratifiedTank* from the Buildings library. The model dividing the tank into *n* segments with uniform temperatures and allowing for multiple inlet and outlet streams. In the partial tank model, a key enhancement and modification is the adjustment of thermal conductivity to better represent diffusion effects. The model captures buoyancy-driven natural convection, caused by the temperature-dependent density of water, leading to internal recirculation between hot and cold regions. Notably, heat loss at the top of the tank may induce an inverse thermocline, a local stratification anomaly.

This model is coupled to a 2-D, axisymmetric ground model in order to simulate underground sTES configurations. The sTES model was cross-compared against a valid model showing its credibility as reported in (Dahash et al. 2020).

The model allows individual specification of heat transfer coefficients at the top, side walls, and bottom of the tank, reflecting different insulation levels.

Subsequently, the sTES model – comprising both the storage tank and surrounding ground domain – is connected to the aforementioned thermo-hydraulic components to represent a hydraulically decoupled sTES system. In this configuration, the sTES can be integrated into a broader energy system via a heat exchanger interface. The complete model is exported from the Dymola simulation environment as an FMU with the type set to Model Exchange and Co-Simulation using CVode, compliant with FMI version 2.0. This FMU is license-free and allows user-defined inputs for key parameters, including storage volume, height, initial temperature, and insulation quality (top, sidewalls, and bottom). The number of thermal segments within the sTES tank is fixed at 15.

The surrounding ground is discretized into 15 axial segments, with a user-defined outer radius. The inner radius is automatically determined by the sTES tank radius. The radial discretization is fixed at 40 segments. Additionally, the model includes a bottom ground domain extending 15 m beneath the tank, discretized into 15 axial and 65 radial segments, with an inner diameter of zero. The model supports spatially inhomogeneous ground properties, allowing users to define thermophysical parameters independently for the lateral and bottom ground domains.

The FMU requires three primary inputs to operate: the injected mass flow rate (*mDot_in*), its temperature (*T_in*), and a control signal (*yDis*) indicating the charging or discharging phase. Based on the value of *yDis*, the model dynamically determines the appropriate injection port – either at the top or bottom of the sTES – reflecting the current operational mode. Additionally, the user can supply a time-series profile for ambient temperature as input, as well as define the undisturbed ground temperature, enabling realistic boundary conditions for long-term thermal simulations.

The FMU provides several outputs over the simulation period, including the evolution of the top, bottom, and mean temperatures of the sTES. Additionally, it reports the total thermal losses, along with a breakdown of these losses by location (e.g., top, sidewalls, bottom). The FMU also outputs the extraction temperature during both charging and discharging phases.

5.2 FMU integration into Simscape

The sTES model, which was developed in Modelica, was integrated into the TSN simulation environment via the *FMU block* from the Simulink *Extras* library. The FMU was coupled at the Simulink level rather than directly within the Simscape physical domain. This is due to the signal-based nature of FMI, which requires the conversion of acausal physical ports in Simscape to causal signal lines in Simulink, and vice versa, using standard *PS-Simulink* and *Simulink-PS* converter blocks. To prevent algebraic loops resulting from feedback between Simscape and the FMU (e.g. temperature and flow dependencies across the interface), *Unit Delay* blocks were introduced at the FMU outputs. To enable efficient multirate simulation, the FMU was imported in co-simulation mode. This allowed it to operate with its own internal solver and step size, decoupled from the Simulink solver. In contrast, model exchange mode forced Simulink to integrate the FMU dynamics at its global solver rate. This prevented multirate execution and resulted in simulation times that were approximately 300% longer in test runs. Co-simulation was therefore better suited to this application, where the TSN and sTES have dynamics with different timescales. The influence of the communication step size between the FMU and the main model/sample time of the FMU on computational time is shown in Chapter 7.

The sTES FMU was linked to the TSN via a heat exchanger in the Energy Hub. This component remains entirely within the Simscape thermal liquid domain and serves as the hydraulic interface for charging and discharging the storage. On the FMU side of the heat exchanger, a set of interface components was implemented as can be seen in Figure 2:

- A *controlled temperature source* (Simscape) defines the outlet temperature of the sTES.
- A *mass flow rate source*, governed by a custom sTES control block, imposes charging or discharging flow rates.
- A *reservoir* serves as a pressure and temperature reference node in the thermal liquid domain.

- *Mass flow rate sensors* and *temperature sensors* capture input data for the FMU.

The FMU inputs include:

- Mass flow rate measured downstream of the flow source.
- Inlet temperature, measured in the return line from the heat exchanger.
- A charging/discharging signal, generated by the sTES control block.
- Ambient air temperature and undisturbed ground temperature, provided as time series inputs.

The main output of the FMU is the outlet temperature. It is forwarded to the controlled temperature source connected to the supply line of the heat exchanger. Additional outputs—the top and bottom temperatures of the sTES—serve as inputs to the control function that determines which storage operational phase is possible.

Signal conversion and unit consistency between Simscape and the FMU were ensured using *Simulink-PS* and *PS-Simulink* Converter blocks, with units set to Kelvin for temperature and kg/s for mass flow.

The sTES control block was implemented as a MATLAB function block, receiving a high-level charge/discharge request from the Energy Hub controller. It evaluates temperature conditions within the sTES and TSN to determine if the requested action can proceed (as described in detail in Chapter 6).

The sTES FMU employed in this study is a water-based thermal energy storage system whose parameters, including volume, height, initial temperature, and thermal resistances of each boundary surface, can be adjusted. For all simulations, the system was configured with a volume of 10,000 m³ and a height of 5 m, resulting in a nominal thermal capacity of approximately 230 MWh based on a design temperature differential of 20 K. Thermal insulation was modelled by assigning thermal resistances of approximately 0.5 W/m²K for the top surface and 1.3 W/m²K for the sidewalls.

While FMU-based integration enables cross-platform modularity, it also introduces structural limitations. The internal structure of the FMU cannot be modified directly. Changes to core physical assumptions, such as converting from a water-based model to a water-gravel model, would require access to the original Modelica source code and regeneration of the FMU. Nevertheless, the parameterised interface provided sufficient flexibility for the simulation scenarios considered in this study. This integration approach is designed for modularity. If future FMUs follow the same interface logic (inputs for mass flow, inlet temperature, and control signal, and outputs for outlet temperature, top, and bottom temperatures) they can be easily integrated with minimal adjustments, allowing for rapid adaptation to other storage models or components.

6 Control Structures

The overall system is governed by four independent control schemes:

- A central controller in the Energy Hub regulates the TSN supply temperature by activating available thermal sources.
- In each building, a local controller maintains the temperature in the heating and cooling buffer tanks within a hysteresis threshold.
- A dedicated controller for the sTES manages its charging and discharging based on system state and surplus or shortage of energy.
- Additionally, a pump controller regulates the TSN mass flow by adjusting the speed of the main hydraulic pump to ensure stable network operation.

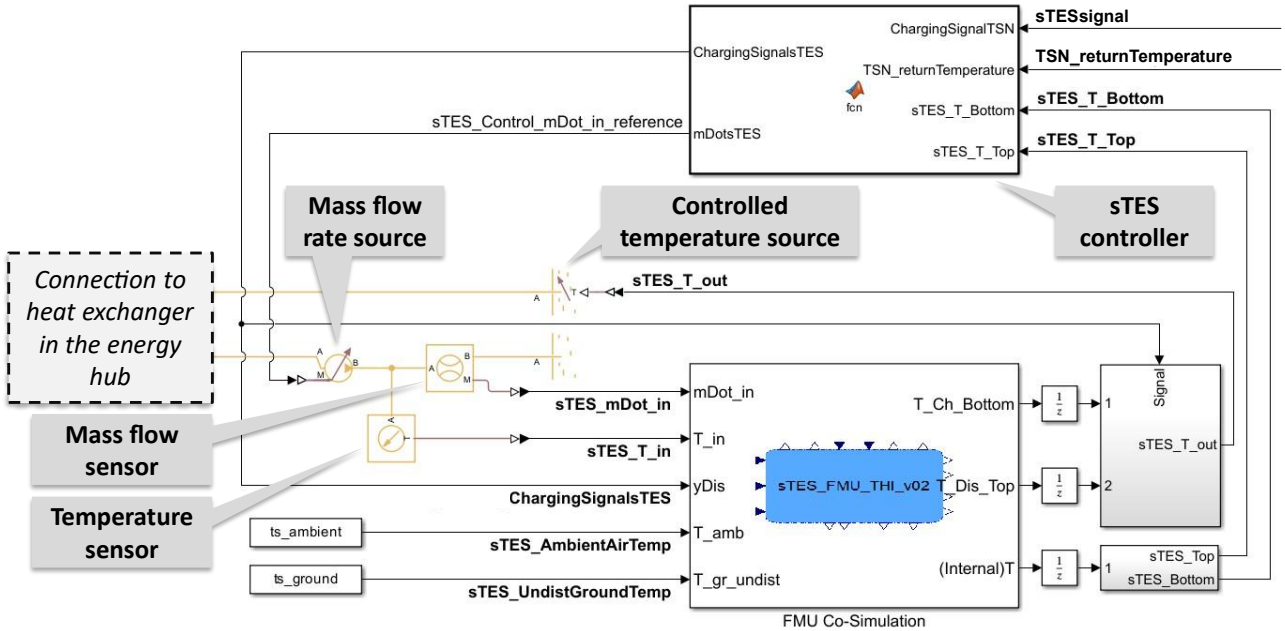


Figure 2. Connections and Interfaces of the sTES FMU.

DOI 10.3384/ecp218851

6.1 Energy Hub Control

The central control scheme ensures that the TSN supply temperature remains within defined limits, adjusting the activation of thermal sources accordingly as depicted in Figure 3. During summer, the supply temperature is maintained between 15°C and 35°C. While in winter, it is regulated within a range of 10°C to 20°C. In summer, a higher temperature reduces cooling losses to the warmer soil, while in winter, a lower temperature helps minimize heat losses to the colder ground. Additionally, the selected temperature bands consider the efficient operation of decentralized heat pumps. To achieve this, the controller can activate three thermal sources: waste heat from the data centre, groundwater wells, and, in future scenarios, the sTES. Herein, the selection follows a simple priority strategy. For both heating and cooling the sTES is prioritized. Only when the sTES cannot meet the demand, the waste heat from the data centre supplies heat as a secondary source, while for cooling, groundwater is used. This overall control logic is implemented using a Stateflow model, which allows for condition-based switching between sources.

6.2 sTES control

The sTES controller, depicted in Figure 4, receives a charging or discharging request from the central Energy Hub controller. Based on this input, it evaluates whether the current temperature levels in the TSN and the sTES allow for the requested operation. Charging is only activated if the bottom temperature of the sTES is at least 2 K below the TSN return temperature, while discharging is allowed if the top temperature of the sTES is at least 2 K above it. If the temperature conditions are met, the

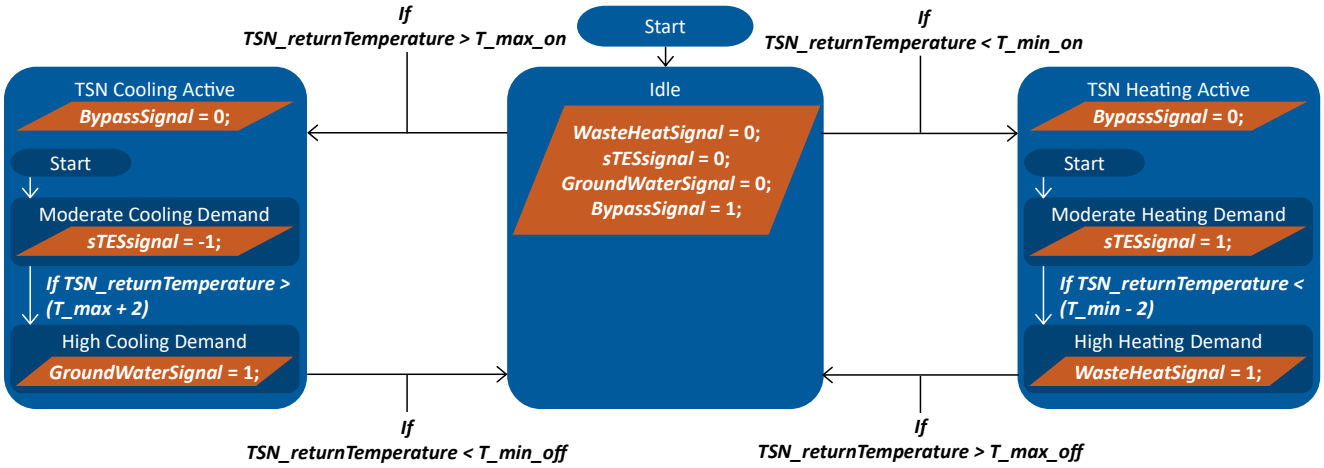


Figure 3. Energy Hub Control Strategy represented as a State Flow Chart.

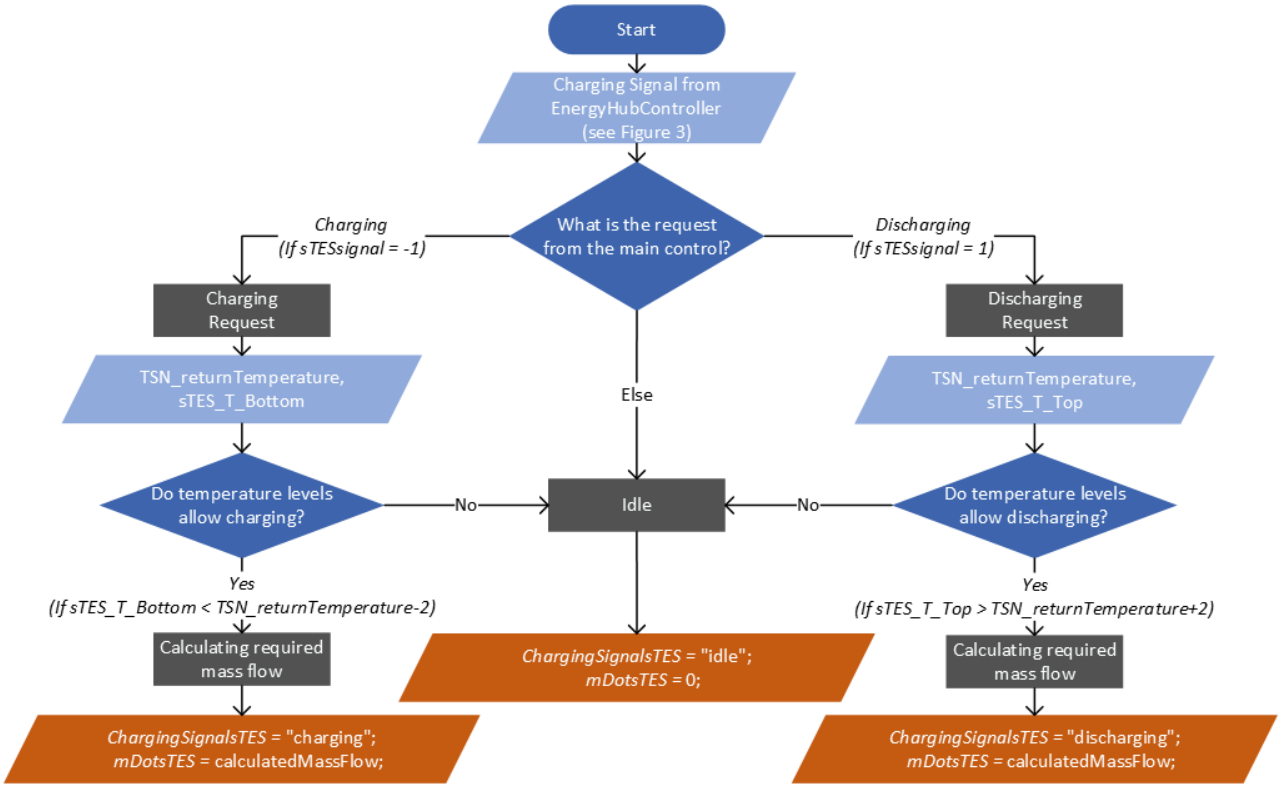


Figure 4. sTES Controller Flow Chart.

controller calculates the required mass flow and sets the operation mode accordingly. If not, the sTES remains idle and no flow is initiated.

6.3 Control Assumptions

To maintain model simplicity, idealized valve behaviour is assumed, and most control actions are treated as instantaneous. However, for pump control, a basic ramp-up function is applied to the control signal to better reflect realistic system dynamics.

7 Results

7.1 FMU Integration Results and Solver Settings

To assess the impact of solver settings on simulation efficiency, the model was evaluated using different combinations of co-simulation step sizes and solver types over a simulation period of 24 hours. Across all configurations, ode23t consistently outperformed all others yielding the shortest simulation times for short-term simulations, while daessc produced the longest, in line with its role in fixed-step real-time applications. Although ode23t was slightly faster than ode15s, it was found to be less stable in long-term simulations, often resulting in errors or early termination. Thus, ode15s was selected for the final year-round simulations, as it provided greater numerical robustness. These outcomes are consistent with MathWorks' recommendations for physical simulations involving stiff systems (The MathWorks, Inc. 2025).

According to MathWorks recommendations, enabling the use of a local solver for Simscape networks can significantly improve simulation performance and numerical stability—particularly in models with complex dynamics or coupled components from different domains. This setup allows the physical network to be updated independently of the global solver, providing greater flexibility in tuning solver configurations for performance and robustness.

In line with this recommendation, a fixed-step local solver was implemented for the Simscape thermo-hydraulic network and evaluated in combination with varying FMU communication step sizes. Both the FMU communication step size and the Simscape solver sample time affected simulation runtime, as illustrated in Figure 5. Larger values for either parameter mostly led to shorter simulation times. However, this came at the risk of reduced numerical stability in long-term simulations.

In the region where the simulation remained stable (specifically including long-term simulations), the Simscape solver sample time was found to be the dominant factor influencing simulation time.

The simulation was stable for Simscape sample times up to 5 seconds and FMU communication step sizes up to 600 seconds, which is attributable to the high thermal inertia of the system. Beyond these values, the model increasingly exhibited temperature divergence and numerical errors, particularly when the Simscape solver was configured with a larger sample time.

For the final configuration, a Simscape solver sample time of 5 seconds and an FMU communication step size of 300 seconds were selected. This combination provided a good compromise between simulation speed and numerical stability and is indicated as the black dot in Figure 5.

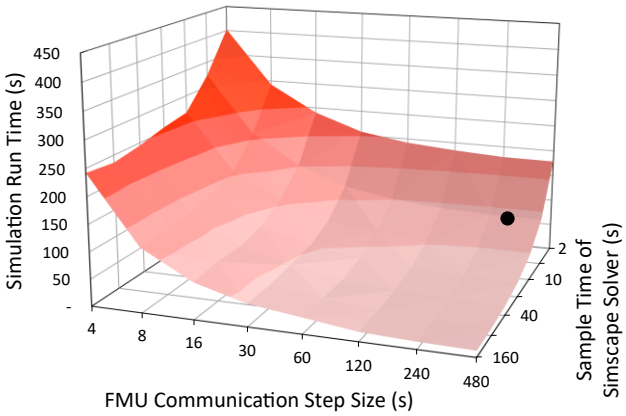


Figure 5. Simulation Run Time of Energy System Model (for 24 Hour Simulated Time) as a Function of FMU Communication Step Size and Simscape Solver Sample Time.

7.2 Energy System Simulation Results

The TSN is analysed in two scenarios:

- Scenario 1 – TSN without sTES: The thermal source network operates using only the available primary sources—waste heat from the data centre and cooling via groundwater wells.
- Scenario 2 – TSN with sTES: The same network configuration is used, but with the addition of a sTES connected to the TSN via a heat exchanger in the Energy Hub. In this setup, the sTES is treated as an integrated extension of the TSN, meaning it is only charged from and discharged to the network. It does not interface directly with any thermal sources or sinks. Charging and discharging occurs according to the control logic presented in Figures 3 and 4.

Figure 6 shows that the TSN return temperatures follow a similar trend in both scenarios. The control logic of the Energy Hub and available thermal sources were sufficient to maintain the requested temperature levels in the network. In the case without sTES, the network temperature fluctuated more drastically, reflecting the direct dependency on real-time availability from primary sources. By contrast, the inclusion of the sTES led to a noticeably smoother return temperature profile. These results showed that the sTES effectively dampened temperature peaks and maintained thermal stability in the network, acting as a buffer between fluctuating demands.

The logged outputs of the sTES FMU provided further insights into the behaviour of the storage unit throughout the simulation year (Figure 7). During the first half of the year, the sTES is gradually charged with surplus heat from the TSN, as seen in the increasing top, bottom, and mean

DOI 10.3384/ecp218851

temperatures (see Figure 7.a). This stratified thermal profile indicates effective layering within the tank. In the second half of the year, the sTES transitions into discharge mode, providing additional thermal energy to the TSN during periods of increased demand. The corresponding thermal power curve (see Figure 7.b) confirms this pattern: negative values represent controlled discharging events, while positive values indicate charging of the sTES. Over the course of the simulation, a total of 289 MWh of heat was charged into the sTES, while 197 MWh was discharged back into the TSN.

8 Discussion

8.1 FMU Integration Results and Solver Settings

From a practical perspective, the integration process using Simulink's built-in FMU block was straightforward, requiring only signal conversion between physical and non-physical domains and careful attention to unit consistency. The most critical aspect was establishing reliable synchronization between the Simscape solver and the FMU co-simulation loop.

Importantly, performance benchmarks showed that the FMU was not a limiting factor in overall simulation time. Instead, simulation speed and stability were more strongly influenced by Simscape solver settings and the choice of global solver.

8.2 Energy System Simulation Results

The simulation results demonstrated that the thermal behaviour of the sTES is strongly influenced by the temperature level of the TSN itself. Since the sTES was connected to the network through a heat exchanger and can only be charged from the TSN, the maximum storage temperature was inherently limited by the network's supply conditions. As a result, the charging potential of the sTES was constrained, particularly during periods when TSN temperatures were suboptimal for storage loading. This limitation could be addressed in future system configurations by enabling charging from alternative sources or introducing a heat pump interface between the TSN and the sTES, thereby decoupling storage charging and discharging from the TSN temperature level leading to addressing sector coupling.

In terms of energy performance, the sTES received a total of 289 MWh of thermal energy during the simulation period and discharged 196 MWh into the TSN, yielding a storage efficiency of approximately 68%. This efficiency reflected the thermal losses to the surrounding ground and environment, as modelled within the FMU, and is within a reasonable range for seasonal storages (Bott et al. 2020).

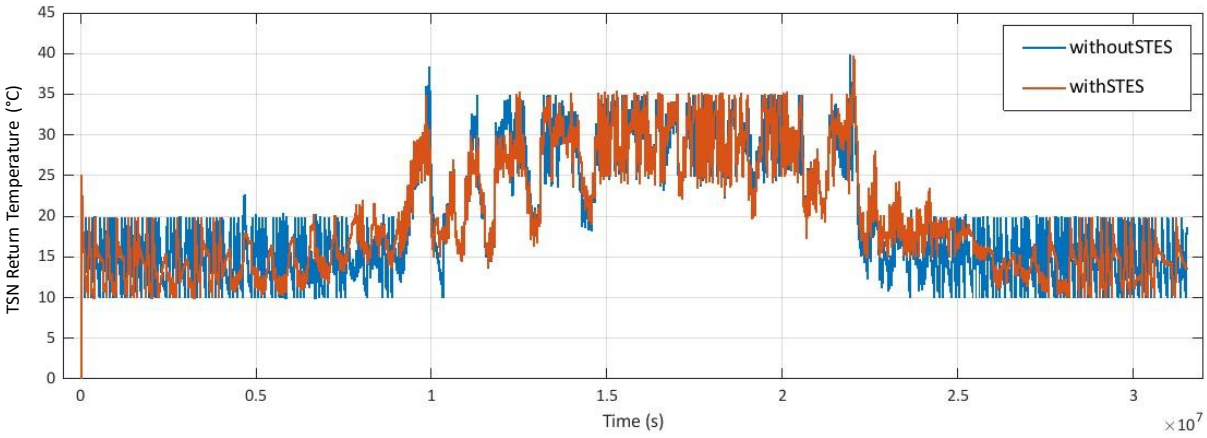


Figure 6. Comparison of TSN return temperature over time for two simulation scenarios.

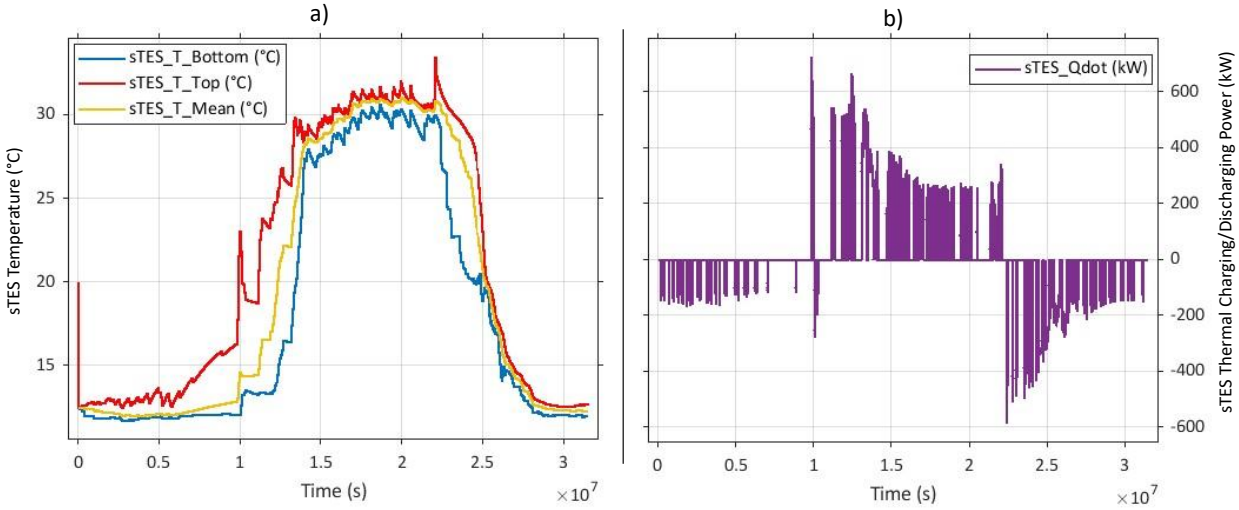


Figure 7. a) sTES temperatures and b) thermal power exchange between sTES and TSN.

8.3 Limitations

While the model successfully demonstrated the integration of an FMU-based sTES and its impact on TSN operation, several assumptions were made to keep the simulation tractable and focused. The pipe-ground interaction was considered using a static undisturbed ground temperature. Furthermore, the heat pump and building models were simplified, relying on Carnot efficiency and load profiles, respectively, rather than detailed thermodynamic or building physics. Additionally, the simulation was conducted over a single year, which limits the ability to fully assess long-term thermal interactions between the sTES and the surrounding ground. The implemented control strategies—both for the Energy Hub and the sTES—were designed for stability and clarity but do not include predictive or optimized logic, and the sTES was treated strictly as a passive TSN extension with no alternative source coupling. Additionally, while the study concentrated on runtime and stability, a more detailed evaluation of numerical precision and energy conservation across the FMI interface is yet to be conducted. Future work will include convergence studies and error analysis to quantify the consistency of energy between the FMU and Simscape subsystems.

9 Conclusion and Future Work

This paper presented a cross-platform simulation approach for modelling TSNs by integrating a Modelica-based FMU of an sTES into a MATLAB/Simscape environment. The co-simulation setup was implemented successfully, highlighting the interoperability potential between modelling domains and enabling the reuse of validated component models without reimplementation.

In the simulated case study, the sTES functioned as an integrated buffer within the TSN, helping to stabilize return temperatures and balance seasonal loads. With 289 MWh of thermal energy charged and 196 MWh discharged, the observed storage efficiency of approximately 68% reflects both thermal losses and current operational limitations, such as temperature coupling through a heat exchanger leading to indirect charging/discharging implying further thermal losses.

Future work will explore more advanced control strategies, such as model predictive control and reinforcement learning, with the aim of improving the operation of the TSN and the sTES. Annual simulations will be used to evaluate the long-term thermal interactions, particularly those with the surrounding ground. Additionally, coupling the sTES via a heat pump or enabling charging from alternative sources could increase flexibility, improve overall system performance, and address sector coupling. Finally, the approach could be extended in reverse by exporting the full TSN model as a MATLAB FMU for integration into Modelica, further demonstrating the bidirectional potential of FMI-based digital twins. It would also be valuable to compare FMU-based sTES integration with a simplified native Simscape implementation to evaluate trade-offs in terms of model fidelity, numerical performance, and cross-domain reusability. Similarly, a quantitative comparison of co-simulation and model-exchange FMUs could help assess the impact of FMU type on accuracy, stability, and energy consistency.

Acknowledgements

This work has been supported by the EU Horizon Europe project INTERSTORES (project no. 101136100).

References

Bott, Christoph; Dressel, Ingo; Bayer, Peter (2020): Closed, In-ground, Water-based Seasonal Storage Systems for Thermal Energy. State-of-technology in Europe. Proceedings World Geothermal Congress 2020. Reykjavik, Iceland.

Bott, Christoph; Ehrenwirth, Mathias; Trinkl, Christoph; Bayer, Peter (2022): Component-based modeling of ground-coupled seasonal thermal energy storages. In: *Applied Thermal Engineering* 214, S. 118810. DOI: 10.1016/j.applthermaleng.2022.118810.

Dahash, Abdulrahman; Mieck, Sebastian; Ochs, Fabian; Krautz, Hans Joachim (2019a): A comparative study of two simulation tools for the technical feasibility in terms of modeling district heating systems: An optimization case study. In: *Simulation Modelling Practice and Theory* 91, S. 48–68. DOI: 10.1016/j.simpat.2018.11.008.

Dahash, Abdulrahman; Ochs, Fabian; Janetti, Michele Bianchi; Streicher, Wolfgang (2019b): Advances in seasonal thermal energy storage for solar district heating applications: A critical review on large-scale hot-water tank and pit thermal energy storage systems. In: *Applied Energy* 239, S. 296–315. DOI: 10.1016/j.apenergy.2019.01.189.

Dahash, Abdulrahman; Ochs, Fabian; Tosatto, Alice (2020): Simulation-Based Design Optimization of Large-Scale Seasonal Thermal Energy Storage in Renewables-Based District Heating Systems. BauSIM 2020 September 23-25, Online Conference.

Li, Xiaoxia; Wang, Zhifeng; Li, Jinping; Yang, Ming; Yuan, Guofeng; Bai, Yakai et al. (2019): Comparison of control strategies for a solar heating system with underground pit seasonal storage in the non-heating season. In: *Journal of Energy Storage* 26, S. 100963. DOI: 10.1016/j.est.2019.100963.

Lyden, A.; Brown, C. S.; Kolo, I.; Falcone, G.; Friedrich, D. (2022): Seasonal thermal energy storage in smart energy systems: District-level applications and modelling approaches. In: *Renewable and Sustainable Energy Reviews* 167, S. 112760. DOI: 10.1016/j.rser.2022.112760.

Müller, Simon Nikolaus; Strobel, Rainer; Faigl, Markus; Schrag, Tobias (2024): Analysis of Industrial 5GDHC System in Ingolstadt. In: *2976-2030* 1. DOI: 10.52825/isec.v1i.1152.

The MathWorks, Inc. (Hg.) (2025): Making Optimal Solver Choices for Physical Simulation. Online available at https://de.mathworks.com/help/simscape/ug/making-optimal-solver-choices-for-physical-simulation.html, last updated on 07.04.2025, last checked on 07.04.2025.

Scalable Higher-order Nonlinear Solvers via Higher-order Automatic Differentiation

Songchen Tan[1] Keming Miao[1] Alan Edelman[1] Christopher Rackauckas[1]

[1]Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA,
{songchen,xbtmkmmm,edelman,crackauc}@mit.edu

Abstract

This paper demonstrates new methods and implementations of nonlinear solvers with higher-order of convergence, which is achieved by efficiently computing higher-order derivatives. Instead of computing full derivatives, which could be expensive, we compute directional derivatives with Taylor-mode automatic differentiation. We first implement Householder's method with arbitrary order for one variable, and investigate the trade-off between computational cost and convergence order. We find that the second-order variant, i.e., Halley's method, to offer the best trade-off between computational cost and convergence rate, and further generalize Halley's method to systems of nonlinear equations and demonstrate that it can scale efficiently to large-scale problems. We further apply Halley's method on solving large-scale ill-conditioned nonlinear problems, as well as solving nonlinear equations inside stiff ODE solvers, and demonstrate that it could outperform Newton's method.

Keywords: nonlinear solvers, automatic differentiation, Householder's method, Halley's method

1 Introduction

Solving nonlinear equations accurately and efficiently is crucial in many disciplines of science, where nonlinear equations can emerge either directly from the models, or as an intermediate step of solving ordinary differential equations (ODEs)(Byrne and Hindmarsh 1987), partial differential equations (PDEs)(Ames 2014), differential-algebraic equations (DAEs)(Kunkel 2006), and integral equations(Atkinson 1992).

Various methods are available to solve nonlinear equations, and the mostly used ones are Newton's method and its variants(Kelley 2003), which rely on the first-order derivative, i.e. the Jacobian, or its approximations. Newton's method is proved to have quadratic convergence under certain assumptions of the nonlinear function and initial condition(Galántai 2000).

There has been extensive research on improving the convergence order of nonlinear solvers, which can be categorized into two different approaches: (1) multi-step methods that evaluate nonlinear functions and/or derivatives at multiple points in each iteration(Potra 1984; X. Xiao and Yin 2015; Madhu and Jayaraman 2017; X.-Y. Xiao 2022; Singh and Sharma 2023); (2) methods that utilize higher-order derivatives of nonlinear functions(Alefeld 1981; A. Cuyt and Cruyssen 1983; A. A. M. Cuyt and Rall 1985; A. A. Cuyt 1982; K. I. Noor, M. A. Noor, and Momani 2007; Ogbereyivwe, Izevbizua, and Umar 2023; Sariman and Hashim 2020; Germani et al. 2006). Despite their higher convergence order, in order to be more efficient than Newton's method, they must be computationally efficient for each iteration. For example, some multi-step methods(Singh and Sharma 2023) evaluate the function and Jacobian at multiple points, but only invert the Jacobian at one point; in practice, this would mean that for a nonlinear function with n inputs and n outputs, the $O(n^3)$ factorization of the Jacobian, which is the bottleneck of Newton's method, only needs to be done once for each iteration. This ensures that these multi-step methods are not much more expensive than Newton's method for each iteration.

The same considerations also apply to methods of the second category, where higher-order derivatives introduce additional computation burden. It is commonly believed that computing the higher-order full derivative of a function with large input and output dimension n would be much more expensive than the computation of the function itself(Margossian 2019). As a result, they are often approximated rather than exactly computed(Sariman and Hashim 2020; Suleiman 2009; Albeanu 2008; Steihaug 2006), and it is still an open question whether nonlinear solvers can utilize higher-order derivative information for large-scale nonlinear problems in an exact way.

This paper demonstrates new methods and implementations that, instead of computing the full derivative, efficiently and exactly compute the higher-order directional derivative, which is sufficient for nonlinear solvers. This paper is organized as follows. We will first introduce the technique of Taylor-mode automatic differentiation (AD) for efficiently computing higher-order directional derivatives, and its implementation in Julia, in section 2. Then we will demonstrate that Taylor-mode AD can be used to efficiently implement Householder's method with arbitrary convergence order, in section 3. Finally, we demonstrate that a special case of Householder's method, the cubic-convergence Halley's method can scale elegantly to large-scale problems, and could outperform first-order methods by a significant margin, in section 4.

2 Taylor-mode automatic differentiation for higher-order directional derivatives

We first introduce the notation used in this paper. Let U be an open subset of $\mathbb{R}^n$, $f : U \to \mathbb{R}^m$ be a function that is sufficiently smooth. The derivative of f at a point $x \in \mathbb{R}^n$ is a linear operator $Df(x) : \mathbb{R}^n \to \mathbb{R}^m$ such that it maps a vector $v \in \mathbb{R}^n$ to $Df(x)[v]$, which is the directional derivative of f at x in the direction of v, also known as the Jacobian-vector product.

Similarly, for $p \in \mathbb{Z}^+ > 1$, the p-th order derivative of f at a point x is a multilinear operator $D^p f(x) : (\mathbb{R}^n, \cdots, \mathbb{R}^n) \to \mathbb{R}^m$ such that it maps a tuple of vectors $(v_1, \ldots, v_p)$ to $D^p f(x)[v_1, \cdots, v_p]$, the directional derivative of f at x in the direction of $(v_1, \ldots, v_p)$.

First-order forward-mode AD can be viewed as an algorithm to propagate directional derivative information through compositions of functions(Revels, Lubin, and Papamarkou 2016). For example, let $f(x) = g(h(x))$ to be a composite function for $\mathbb{R}^n \to \mathbb{R}^m$, and $x \in \mathbb{R}^n$ to be a specific point in its domain. Suppose that we already have $h_0 = h(x)$ to represent the primal output of h, and $h_1 = Dh(x)[v]$ representing the perturbation of h along some direction $v \in \mathbb{R}^n$. Now for the composite function, we want to know what is the perturbation of f along direction v; in other words, we want to know $f_1 = Df(x)[v]$ in addition to $f_0 = f(x)$. The answer to that is simply the chain rule:

- $f_0 = g(h_0)$
- $f_1 = Dg(h_0)[h_1]$

Similarly, the Taylor-mode AD can be viewed as an algorithm to propagate higher-order directional derivative information through compositions of functions. Suppose that we already have a Taylor coefficients for h at x along v:

$$(h_0, h_1, \ldots, h_p) = (h(x), Dh(x)[v], \ldots, D^p h(x)[v, \ldots, v])$$

and we want to know

$$(f_0, f_1, \ldots, f_p) = (f(x), Df(x)[v], \ldots, D^p f(x)[v, \ldots, v])$$

the answer to that is, in turn, the Faà di Bruno's formula:

- $f_0 = g(h_0)$
- $f_1 = Dg(h_0)[h_1]$
- $f_2 = D^2 g(h_0)[h_1, h_1] + Dg(h_0)[h_2]$
- $\ldots$

In our practical implementation of Taylor-mode AD, a pushforward rule is defined for every "simple" function, so that derivatives of any complicated functions that are composed of these simple functions can be computed automatically, either via operator-overloading or source-code transformation techniques. Our previous work (Tan 2023a) has shown that for any functions composed of elementary functions and arbitrary control flow, the p-th order directional derivative is at most $O(p^2)$ times more expensive than computing the function itself, in contrast to naively nesting first-order forward-mode AD which could be $O(\exp(p))$ times expensive(Bettencourt, Johnson, and Duvenaud 2022). We also provided a reference implementation of this method in the Julia language(Bezanson, Edelman, et al. 2017) using its multiple-dispatch mechanism(Bezanson, Bolewski, and Chen 2018), TaylorDiff.jl(Tan 2023b), featuring an automatic generation(Gowda, Ma, Cheli, et al. 2021) of higher-order pushforward rules from first-order pushforward rules defined in ChainRules.jl(White et al. 2023).

Below, we will demonstrate how to use Taylor-mode AD to implement higher-order nonlinear solvers. For all numerical experiments in this paper, we use the Julia language v1.11.2 on a single core of Intel Xeon Platinum 8260 CPU provided by the MIT SuperCloud system.

3 Efficient implementation of Householder's method

We begin by implementing Householder's method(Householder 1970) for solving nonlinear equations with one variable, i.e. $x \in \mathbb{R}$ in the nonlinear function $f(x)$. This does not immediately generalize to multiple variables, but it is useful to compare the behavior of solvers with different convergence order. The method is defined as follows: given an initial guess x_0, in each iteration we compute,

$$x_{n+1} = x_n + p\frac{(1/f)^{(p-1)}(x_n)}{(1/f)^{(p)}(x_n)}.$$

which requires derivatives up to p-th order and has convergence order $p+1$. Note that when $p = 1$, this method is equivalent to Newton's method; and when $p = 2$, this method is equivalent to Halley's method.

Given f and x_n, we can compute $f(x_n)$ and its derivatives up to p-th order using Taylor mode AD, and then invert this Taylor polynomial to get the value and derivatives of $1/f$ up to p-th order. Our analysis leads to the algorithm 1:

3.1 Cost-effectiveness for Householder's method with different orders.

We implement algorithm 1 as a part of SimpleNonlinearSolve.jl(Pal et al. 2024), which is a package of non-allocating nonlinear solvers that have low-overhead and could solve small nonlinear problems very efficiently. We make several univariate nonlinear functions composed of various elementary functions, and use different orders of Householder's method to solve $f(x) = 0$ given appropriate initial guess. Results are shown in fig. 1.

DOI 10.3384/ecp218861

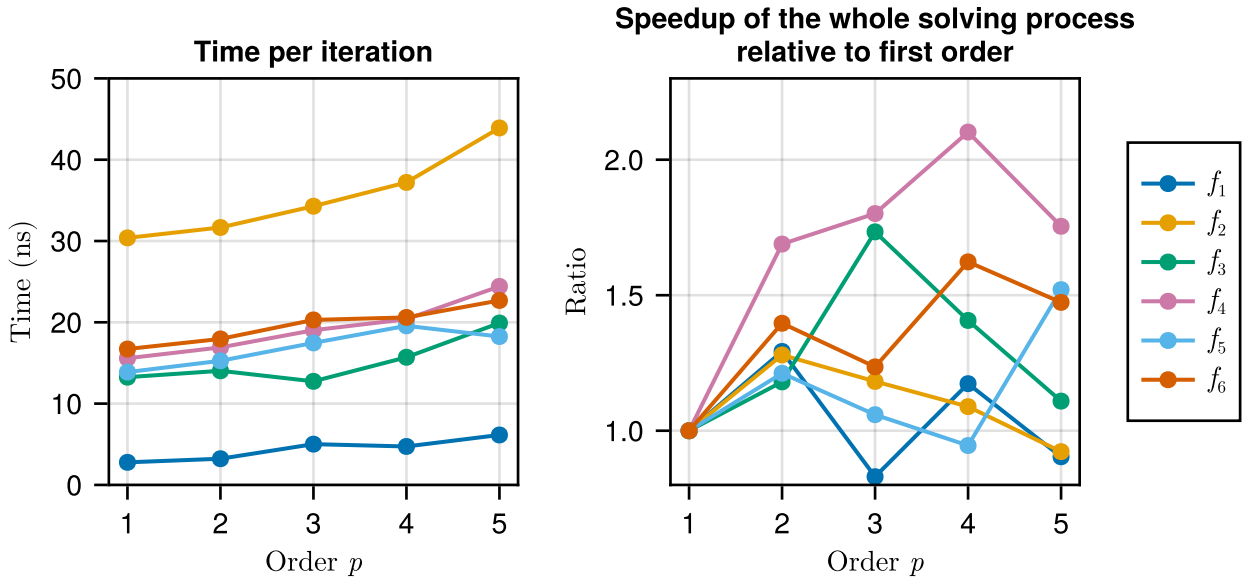


Figure 1. Cost-effectiveness for Householder's method with different orders. Left: the computation time per iteration for each nonlinear function and each Householder's method with different order. Right: the relative speedup for each function of the total computation time to solve the nonlinear equation, normalized by the $p = 1$ solver (Newton's method). The functions and initial conditions are: (1) $f_1(x) = x^2 - 2$, $x_0 = 1.0$; (2) $f_2(x) = \sqrt{x} - \pi$, $x_0 = 10.0$; (3) $f_3(x) = x - \exp(-x)$, $x_0 = 0.0$; (4) $f_4(x) = x^2 - 2^x$, $x_0 = 3.3$; (5) $f_5(x) = x + \sin(x) - 1$, $x_0 = 0.5$; (6) $f_6(x) = \log(x) + x$, $x_0 = 1.0$.

Algorithm 1 Householder's method implementation based on higher-order AD

Define f and initial value x; tolerance t; order p
while $|f(x)| > t$ **do**
  Initialize Taylor coefficients $(x, 1, 0, \cdots, 0)$
  Apply the pushforward rule of f to get $(f(x), f'(x), \cdots, f^{(p)}(x))$
  Apply the pushforward rule of $(\cdot)^{-1}$ to get $((1/f)(x), (1/f)'(x), \cdots, (1/f)^{(p)}(x))$
  Update $x := x + p\frac{(1/f)^{(p-1)}(x)}{(1/f)^{(p)}(x)}$.
end while
return x

For each of the nonlinear functions, the computation cost per iteration only slightly increases as the order p goes from 1 (Newton's method) to 5, but the number of iterations needed to converge could be cut down significantly by the higher-order methods. As the result, the total time needed to converge could be shorter for higher-order methods. We also notice that the most significant improvement is from the first order to the second order, and the improvement from the second order to an even higher order is less significant.

3.2 Generalization to multivariate functions

The theory of abstract rational approximation(A. Cuyt and Cruyssen 1983) gives a method to generalize Householder's method to multivariate functions. In the sections below, we will focus on implementing $p = 2$ case, i.e. Halley's method, with abstract rational approximation formulas, for multivariate functions. Based on the observations from simple univariate problems, we hypothesize that higher-order methods with $p \geq 3$ yield diminishing returns in practice.

4 Scalable and Efficient implementation of Halley's method

Halley's method ($p = 2$) for a multivariate real function $f(x) : \mathbb{R}^n \to \mathbb{R}^n$, derived from abstract rational approximation(A. Cuyt and Cruyssen 1983; A. A. M. Cuyt and Rall 1985), could be expressed as follows: given an initial guess x_0, in each iteration we compute

$$x_{n+1} = x_n + (a_n \odot a_n) \oslash (a_n + [Df(x_n)]^{-1} D^2 f(x_n)[a_n, a_n]/2) \quad (1)$$

where $\odot$ and $\oslash$ are element-wise multiplication and division, Df and $D^2 f$ are the Jacobian and Hessian of f, and a_n is the solution to $Df(x_n)a_n = -f(x_n)$. In the equation above, $D^2 f(x_n)a_n a_n$ can be computed easily using Taylor-mode AD, with just two times more expensive than the computation of $f(x_n)$. After the first linear solve for a_n, we need another linear solve to get $[Df(x_n)]^{-1} D^2 f(x_n)a_n a_n$. Among the two linear solves, the factorization of Jacobian could be reused, similar to the case of multi-step methods(Singh and Sharma 2023). Our analysis leads to algorithm 2:

Looking into the details of algorithm 2, we note that for

Algorithm 2 (Multivariate) Halley's method implementation based on higher-order AD

Define f and initial value x; tolerance t
while $\|f(x)\|_{\text{inf}} > t$ **do**
  Obtain via first-order AD the Jacobian $Df(x)$
  Solve a from $Df(x)[a] = -f(x)$
  Initialize Taylor coefficients $(x, a, 0)$
  Apply the pushforward rule of f to get $(f(x), Df(x)[a], D^2f(x)[a,a])$
  Solve b from $Df(x)[b] = D^2(x)[a,a]$
  Update $x := x + a \odot a \oslash (a + b/2)$
end while
return x

either dense or sparse Jacobian, the construction and factorization of $Df(x)$ (via LU or QR factorization) should be the bottleneck of solving, and these only need to be done once for the two linear solves. In other words, given the factorization, the additional linear solve for b can be computed in $O(n^2)$ time. Therefore, each step of Halley's method can be asymptotically as cheap as Newton's method, which makes it potentially faster than Newton's method.

4.1 Solving nonlinear problems with dense Jacobian

We first test algorithm 2 on nonlinear problems where the Jacobian of the nonlinear function $f(x) : \mathbb{R}^n \to \mathbb{R}^n$ is dense and unstructured. In this case, the factorization of the Jacobian is the bottleneck of the linear solve, and the cost of each factorization is $O(n^3)$. For defining such a problem, we consider the Chandrasekhar's H-function(Chandrasekhar 2013) which is the solution to an integral equation that arises in the study of radiative transfer in astrophysics. The equation is defined as follows:

$$H(\mu) = 1 + \mu H(\mu) \int_0^1 \frac{\Psi(\mu')H(\mu')}{\mu + \mu'} \mathrm{d}\mu' \tag{2}$$

where $H(\mu)$ is defined on $[0,1]$ and $\Psi(\mu)$ is an even polynomial. We consider the most simplified case, where Ψ is a constant. Upon discretization onto n grids for integration, the equation becomes a system of nonlinear equations of n variables, and since each equation depends on the value of H on all positions, the Jacobian of this nonlinear problem is inherently dense. The detailed form of the discretized nonlinear function is available as the 23rd problem in NonlinearProblemLibrary.jl(Pal et al. 2024) which the reader can refer to.

We solve this problem with different problem sizes $n = 4, 8, 16, 32, 64, 128$, and we compare the performance of (1) Newton's method, (2) Halley's method as implemented in algorithm 2, and (3) naive Halley's method implemented in a conventional way(A. A. M. Cuyt and Rall 1985) that computes the full Hessian and then contracts it with vectors. Each of them is solved to the default tolerance. The results are shown in fig. 2.

While the performances are similar for small problems, the naive Halley's method quickly becomes infeasible for large problems(Pal et al. 2024), while our implementation of Halley's method scales similarly as Newton's method. In addition, as the problem size gets larger, the advantage of Halley's method over Newton's method becomes more significant, as a result of fewer iterations needed to converge. For example, for $n = 128$, Halley's method is approximately 40% faster than Newton's method. Finally, more detailed work-precision diagrams comparing Halley's method and Newton's method, within a broader range of problem sizes ($n = 4, 16, 64, 256, 1024$), is shown in fig. 3.

4.2 Solving large-scale ill-conditioned nonlinear problems with sparse Jacobian

Nonlinear problems that appear in solving PDEs are often large, sparse and ill-conditioned(Ames 2014), imposing additional challenges for nonlinear solvers. Below, we will use a two-dimensional Brusselator reaction-diffusion PDE problem as an example to demonstrate the capability of Halley's method to handle these nonlinear problems in practical applications. The Brusselator is a model for auto-catalytic chemical systems that exhibit oscillations in time domain and pattern formation in space domain(Prigogine and Lefever 1968). Let functions $u(x,y,t)$ and $v(x,y,t)$ be concentrations of substances that are defined on $(x,y) \in [0,1] \times [0,1]$ and $t \in (0, +\infty)$, satisfying the following differential equations,

$$\begin{aligned} u_t &= B + u^2 v - (A+1)u + \alpha \Delta u + f(x,y,t) \\ v_t &= Au - u^2 v + \alpha \Delta v \end{aligned} \tag{3}$$

where the source f is defined as

$$f(x,y,t) = \begin{cases} 5 & \text{if } (x-0.3)^2 + (y-0.6)^2 \le 0.1^2 \text{and } t \ge 1.1 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

and initial conditions,

$$\begin{aligned} u(x,y,0) &= 22(y(1-y))^{3/2} \\ v(x,y,0) &= 27(x(1-x))^{3/2} \end{aligned} \tag{5}$$

and periodic boundary conditions.

$$\begin{aligned} u(0,y,t) &= u(1,y,t) \\ u(x,0,t) &= u(x,1,t) \\ v(0,y,t) &= v(1,y,t) \\ v(x,0,t) &= v(x,1,t) \end{aligned} \tag{6}$$

In the following numerical experiments, the parameters are set to $A = 3.4$, $B = 1$, $\alpha = 10$.

We first consider the time-independent PDE problem, where for $t \gg 1.1$ we solve the steady-state equations $u_t = v_t = 0$ to get the long-time behavior of the spatial distribution of concentrations u and v. In order to apply nonlinear solvers, the spatial discretization of u and v is

DOI 10.3384/ecp218861

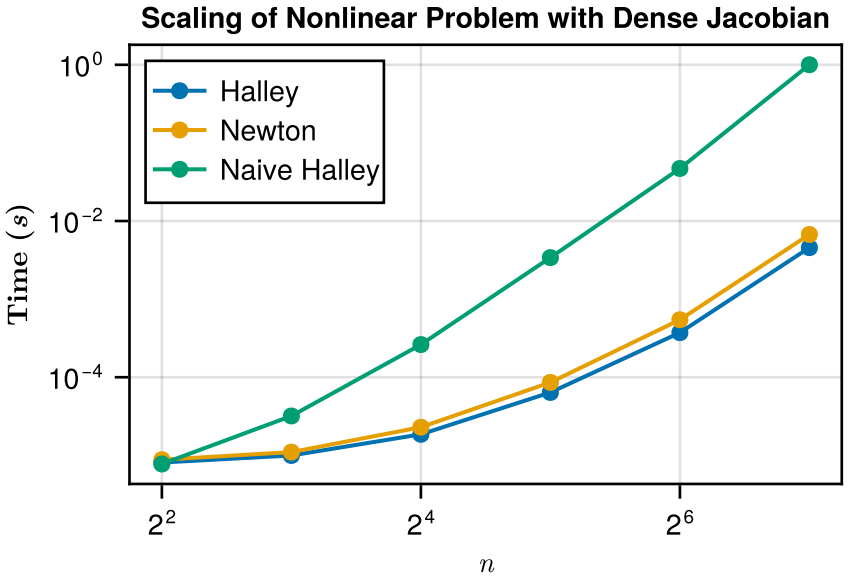


Figure 2. Scaling for solving a nonlinear problem that has a dense Jacobian, at different problem sizes. Solvers: Newton, Halley (Taylor-mode AD), and Naive Halley (full Hessian).

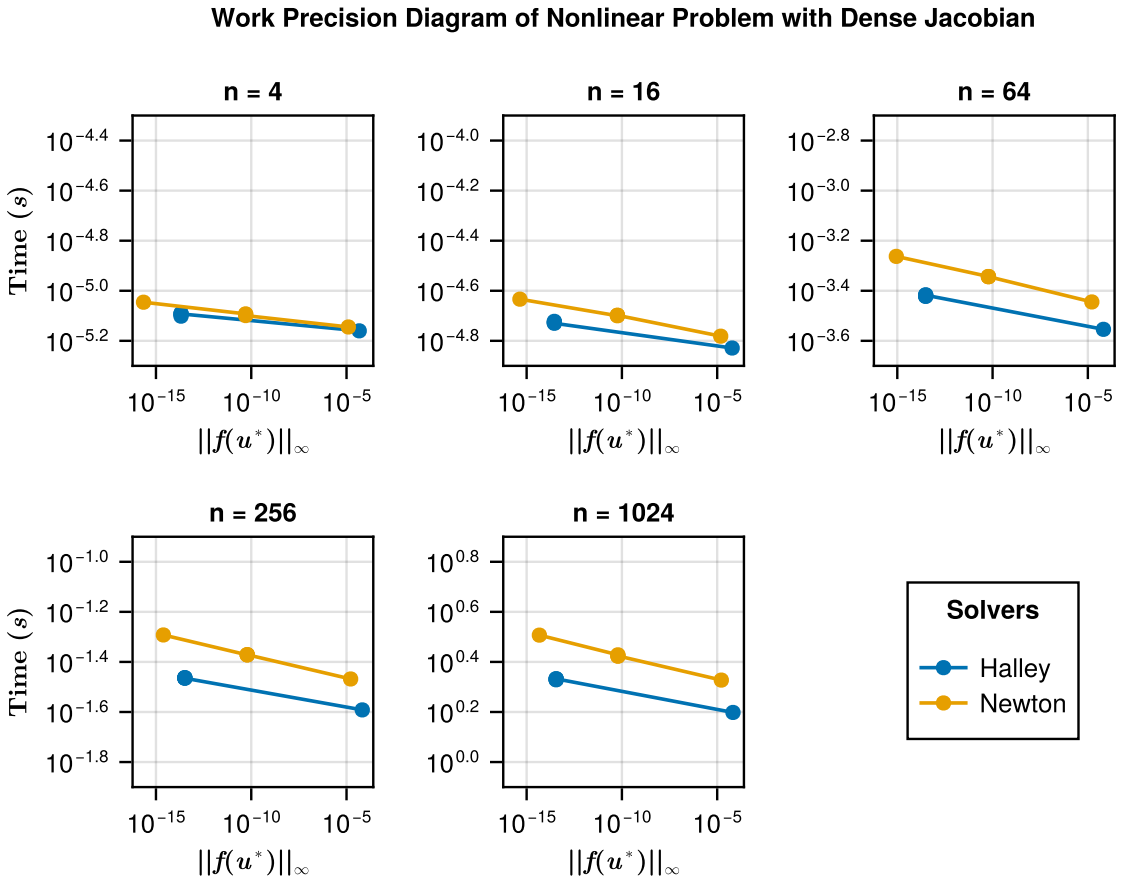


Figure 3. Work-precision diagram for solving a nonlinear problem that has a dense Jacobian, with different sizes. Solvers: Newton and Halley (Taylor-mode AD).

carried out for $K = 4, 8, 16, 32, 64, 128$ grids on x and y directions, and the Laplacian operator Δ is implemented with finite differences. Therefore, each of u and v discretizes to K^2 variables, and the total problem size of the nonlinear system is $n = 2K^2$. Since the variables only interact with nearby grids, the resulting Jacobian is sparse; in addition, when K increases, the coefficients of the finite difference operator become larger and make the Jacobian ill-conditioned.

In order to handle these characteristics, we first try to generate a sparse matrix for Jacobian in order to make factorization easier(Averick et al. 1994). This could be achieved with sparse detection techniques, either in a numerical way(Giering, Kaminski, and Slawig 2005; Walther and Griewank 2009) or in a symbolic way(Gowda, Ma, Churavy, et al. 2019). After sparsity detection, we further use the graph coloring algorithm(Gebremedhin, Manne, and Pothen 2005) to compute the matrix. These tools are available in SparseDiffTools.jl(*JuliaDiff/SparseDiffTools.jl* 2024) and SparseConnectivityTracer.jl(Hill and Dalle 2024) as a part of the Julia community. Finally, factorization of the sparse Jacobian is appropriately handled with LinearSolve.jl(*SciML/LinearSolve.jl* 2024) which has reasonable polyalgorithms for handling ill-conditioned sparse matrices. In addition, we also include the naive method of computing dense Jacobian and then doing dense factorization. The results are shown in fig. 4.

Clearly, the naive method of computing dense Jacobian and then doing dense factorization is inefficient for large problems, while the sparse method scales better. In addition, Halley's method is more efficient than Newton's method for all problem sizes and for both sparse and dense handling of the Jacobian matrix, and the advantage becomes more significant as the problem size increases. For example, for $K = 32$ and corresponding problem size $n = 2048$, Halley's method is approximately 25% faster than Newton's method.

4.3 Solving stiff ODEs

We then consider the time-dependent PDE, where we compute the time evolution of the Brusselator system given initial conditions and periodic boundary conditions above. In order to implement, the PDE is again spatially discretized with K grids on x and y directions, turning it into an ODE. Due to the Laplacian operator, the linear part of this equation has a high frequency, while the nonlinear part has a low frequency. Therefore, the resulting ODE is stiff. In solving stiff ODEs, implicit solvers are often used to ensure accuracy and efficiency(Kim et al. 2021), and these solvers requires nonlinear solve steps. Since the nonlinear solve steps are often the bottleneck of implicit solvers, impoving the nonlinear solver could accelerate the whole ODE solving process.

We choose problem size $K = 8$, $n = 2K^2 = 128$ to construct the ODE and solve with several common stiff solvers from $t_0 = 0$ to $t_e = 11.5$, and in each of the stiff solvers we test both Newton's method and Halley's method for the underlying nonlinear solver. When solving the ODE, we provide different tolerances which will determine the time step by applying adaptive time-step strategies available in DifferentialEquations.jl(Rackauckas and Nie 2017). We then use a very small tolerance to obtain very accurate solutions to the ODE, namely u_{ref} and v_{ref}, and we define the error to be the relative L^2 error of the final solution at t_e on domain $D = [0,1] \times [0,1]$, i.e.

$$\frac{\int_D \left(|\delta u(x,y,t_e)|^2 + |\delta v(x,y,t_e)|^2\right) \mathrm{d}x\mathrm{d}y}{\int_D \left(|u_{\text{ref}}(x,y,t_e)|^2 + |v_{\text{ref}}(x,y,t_e)|^2\right) \mathrm{d}x\mathrm{d}y} \tag{7}$$

where $\delta u = u - u_{\text{ref}}$ and $\delta v = v - v_{\text{ref}}$ respectively. The results are shown in fig. 5.

In the work-precision diagram, we observe that Halley's method is more efficient than Newton's method for many kinds of implicit solvers, and on average it gives a 5-10% speedup for the whole ODE solving process.

5 Conclusions

In this paper, we have demonstrated that higher-order nonlinear solvers can be implemented efficiently and scalable with higher-order automatic differentiation. We have implemented Householder's method with arbitrary convergence order and investigated the trade-off between computational cost and accelerated convergence. We have implemented $p = 2$ variant, Halley's method, for multivariate functions and have shown that Halley's method can be more efficient than Newton's method for several cases, including problems with dense Jacobian, problems with sparse Jacobian, as well as being integrated as a part of stiff ODE solvers. Backed by the Julia language and NonlinearSolve.jl framework, this method can be applied to a wide range of problems in science and engineering, and we believe that it could be a valuable tool for the scientific community.

The most noticeable limitation of this work is that we have not yet considered the case where the linear equations in each iteration must be solved iteratively, such as the case that the Jacobian is a matrix-free operator, which is common in even larger systems(Knoll and Keyes 2004). In this case, no factorization could be reused between two linear solves required by Halley's method, and the cost of each iteration could be twice as expensive as Newton's method, cancelling out the advantage of higher convergence order. In future work, we might consider constructing alternative versions of Halley's method that could reuse Krylov iterations of large linear solves.

Acknowledgements

This material is based upon work supported by the U.S. National Science Foundation under award Nos CNS-2346520, PHY-2028125, RISE-2425761, DMS-2325184, OAC-2103804, and OSI-2029670, by the Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR00112490488, by the Department of Energy,

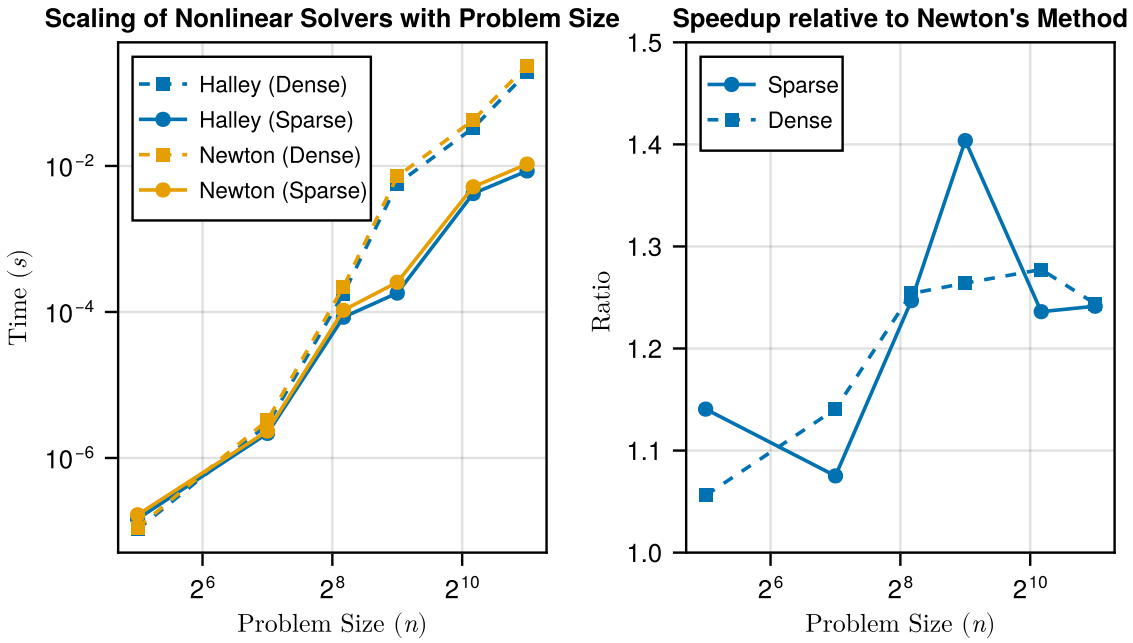


Figure 4. Scaling for solving discretized two-dimensional Brusselator steady-state problem with different problem sizes. The problems are solved to default tolerance. Left: computation time for Newton and Halley's method, each with or without sparsity detection, for each problem size. Right: relative speedup of Halley's method over Newton's method with or without sparsity detection, for each problem size.

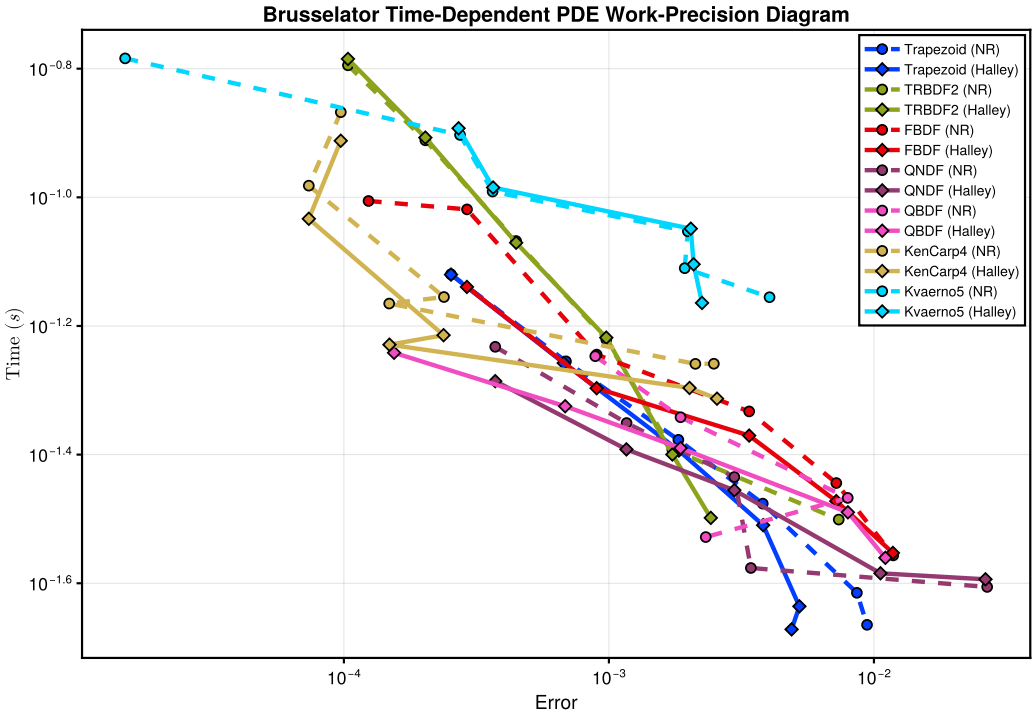


Figure 5. Work-precision diagram for solving discretized two-dimensional Brusselator time-dependent PDE. Implicit Solvers: Trapezoid(Vladimirescu 1994), TRBDF2(Hosea and L. F. Shampine 1996), FBDF(L. F. Shampine 2002), QNDF(Lawrence F. Shampine and Reichelt 1997), QBDF (alias of QNDF with $\kappa = 0$), KenCarp4(Kennedy and Carpenter 2003), Kvaerno5(Kværnø 2004). For each solver, the Newton's method is drawn in circle and solid line, and the Halley's method is drawn in diamond and dashed line.

National Nuclear Security Administration under Award Number DE-NA0003965 and by the United States Air Force Research Laboratory under Cooperative Agreement Number FA8750-19-2-1000. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the United States Air Force or the U.S. Government.

References

Albeanu, Grigore (2008). "On the generalized Halley method for solving nonlinear equations". In: *ROMAI J* 4.2, pp. 1–6.

Alefeld, Georg (1981). "On the convergence of Halley's Method". In: *The American Mathematical Monthly* 88.7, pp. 530–536.

Ames, William F (2014). *Numerical methods for partial differential equations*. Academic press.

Atkinson, Kendall E (1992). "A survey of numerical methods for solving nonlinear integral equations". In: *The Journal of Integral Equations and Applications*, pp. 15–46.

Averick, Brett M et al. (1994). "Computing large sparse Jacobian matrices using automatic differentiation". In: *SIAM Journal on Scientific Computing* 15.2, pp. 285–294.

Bettencourt, Jesse, Matthew J. Johnson, and David Duvenaud (2022-07). "Taylor-Mode Automatic Differentiation for Higher-Order Derivatives in JAX". en. In: URL: https://openreview.net/forum?id=SkxEF3FNPH (visited on 2022-12-22).

Bezanson, Jeff, Jake Bolewski, and Jiahao Chen (2018). "Fast flexible function dispatch in Julia". In: *arXiv preprint arXiv:1808.03370*.

Bezanson, Jeff, Alan Edelman, et al. (2017-01). "Julia: A Fresh Approach to Numerical Computing". In: *SIAM Review* 59.1. Publisher: Society for Industrial and Applied Mathematics, pp. 65–98. ISSN: 0036-1445. DOI: 10.1137/141000671. URL: https://epubs.siam.org/doi/10.1137/141000671 (visited on 2022-12-22).

Byrne, George D and Alan C Hindmarsh (1987). "Stiff ODE solvers: A review of current and coming attractions". In: *Journal of Computational physics* 70.1, pp. 1–62.

Chandrasekhar, Subrahmanyan (2013). *Radiative transfer*. Courier Corporation.

Cuyt, Annie and Paul van der Cruyssen (1983-01). "Abstract Padé-approximants for the solution of a sytem of nonlinear equations". In: *Computers & Mathematics with Applications* 9.4, pp. 617–624. ISSN: 0898-1221. DOI: 10.1016/0898-1221(83)90119-0. URL: https://www.sciencedirect.com/science/article/pii/0898122183901190 (visited on 2024-02-22).

Cuyt, Annie A. M. and L. B. Rall (1985-03). "Computational implementation of the multivariate Halley method for solving nonlinear systems of equations". en. In: *ACM Transactions on Mathematical Software* 11.1, pp. 20–36. ISSN: 0098-3500, 1557-7295. DOI: 10.1145/3147.3162. URL: https://dl.acm.org/doi/10.1145/3147.3162 (visited on 2024-02-20).

Cuyt, Annie AM (1982). "Numerical stability of the Halley-iteration for the solution of a system of nonlinear equations". In: *Mathematics of Computation* 38.157, pp. 171–179.

Galántai, Aurel (2000). "The theory of Newton's method". In: *Journal of Computational and Applied Mathematics* 124.1-2, pp. 25–44.

Gebremedhin, Assefaw Hadish, Fredrik Manne, and Alex Pothen (2005-01). "What Color Is Your Jacobian? Graph Coloring for Computing Derivatives". In: *SIAM Review* 47.4. Publisher: Society for Industrial and Applied Mathematics, pp. 629–705. ISSN: 0036-1445. DOI: 10.1137/S0036144504444711. URL: https://epubs.siam.org/doi/abs/10.1137/S0036144504444711 (visited on 2024-12-09).

Germani, A. et al. (2006-12). "Higher-Order Method for the Solution of a Nonlinear Scalar Equation". en. In: *Journal of Optimization Theory and Applications* 131.3, pp. 347–364. ISSN: 1573-2878. DOI: 10.1007/s10957-006-9154-0. URL: https://doi.org/10.1007/s10957-006-9154-0 (visited on 2024-10-09).

Giering, R., T. Kaminski, and T. Slawig (2005-10). "Generating efficient derivative code with TAF: Adjoint and tangent linear Euler flow around an airfoil". In: *Future Generation Computer Systems* 21.8, pp. 1345–1355. ISSN: 0167-739X. DOI: 10.1016/j.future.2004.11.003. URL: https://www.sciencedirect.com/science/article/pii/S0167739X04001785 (visited on 2024-12-09).

Gowda, Shashi, Yingbo Ma, Alessandro Cheli, et al. (2021-09). "High-performance symbolic-numerics via multiple dispatch". en. In: *ACM Communications in Computer Algebra* 55.3, pp. 92–96. ISSN: 1932-2240. DOI: 10.1145/3511528.3511535. URL: https://dl.acm.org/doi/10.1145/3511528.3511535 (visited on 2022-12-22).

Gowda, Shashi, Yingbo Ma, Valentin Churavy, et al. (2019-09). "Sparsity Programming: Automated Sparsity-Aware Optimizations in Differentiable Programming". en. In: URL: https://openreview.net/forum?id=rJlPdcY38B (visited on 2024-12-09).

Hill, Adrian and Guillaume Dalle (2024-11). *SparseConnectivityTracer.jl*. DOI: 10.5281/zenodo.14216520. URL: https://zenodo.org/records/14216520 (visited on 2024-12-09).

Hosea, M. E. and L. F. Shampine (1996-02). "Analysis and implementation of TR-BDF2". In: *Applied Numerical Mathematics*. Method of Lines for Time-Dependent Problems 20.1, pp. 21–37. ISSN: 0168-9274. DOI: 10.1016/0168-9274(95)00115-8. URL: https://www.sciencedirect.com/science/article/pii/0168927495001158 (visited on 2024-12-09).

Householder, Alston Scott (1970). "The numerical treatment of a single nonlinear equation". In: *(No Title)*.

JuliaDiff/SparseDiffTools.jl (2024-11). original-date: 2019-03-27T19:57:51Z. URL: https://github.com/JuliaDiff/SparseDiffTools.jl (visited on 2024-12-09).

Kelley, Carl T (2003). *Solving nonlinear equations with Newton's method*. SIAM.

Kennedy, Christopher A. and Mark H. Carpenter (2003-01). "Additive Runge–Kutta schemes for convection–diffusion–reaction equations". In: *Applied Numerical Mathematics* 44.1, pp. 139–181. ISSN: 0168-9274. DOI: 10.1016/S0168-9274(02)00138-1. URL: https://www.sciencedirect.com/science/article/pii/S0168927402001381 (visited on 2024-12-09).

Kim, Suyong et al. (2021). "Stiff neural ordinary differential equations". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 31.9. Publisher: AIP Publishing LLC, p. 093122.

Knoll, D.A. and D.E. Keyes (2004). "Jacobian-free Newton–Krylov methods: a survey of approaches and applications". In: *Journal of Computational Physics* 193.2, pp. 357–397. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2003.08.010. URL: https://www.sciencedirect.com/science/article/pii/S0021999103004340.

Kunkel, Peter (2006). *Differential-algebraic equations: analysis and numerical solution*. Vol. 2. European Mathematical Society.

Kværnø, A. (2004-08). "Singly Diagonally Implicit Runge–Kutta Methods with an Explicit First Stage". en. In: *BIT Numerical Mathematics* 44.3, pp. 489–502. ISSN: 1572-9125. DOI: 10.1023/B:BITN.0000046811.70614.38. URL: https://doi.org/10.1023/B:BITN.0000046811.70614.38 (visited on 2024-12-09).

Madhu, Kalyanasundaram and Jayakumar Jayaraman (2017). "Some higher order Newton-like methods for solving system of nonlinear equations and its applications". In: *International Journal of Applied and Computational Mathematics* 3.3, pp. 2213–2230.

Margossian, Charles C (2019). "A review of automatic differentiation and its efficient implementation". In: *Wiley interdisciplinary reviews: data mining and knowledge discovery* 9.4, e1305.

Noor, Khalida Inayat, Muhammad Aslam Noor, and Shaher Momani (2007). "Modified Householder iterative method for nonlinear equations". In: *Applied mathematics and computation* 190.2, pp. 1534–1539.

Ogbereyivwe, Oghovese, Orobosa Izevbizua, and Salisu Shehu Umar (2023). "Some high-order convergence modifications of the Householder method for nonlinear equations". In: *Communications in Nonlinear Analysis* 11.2, pp. 1–11.

Pal, Avik et al. (2024-03). *NonlinearSolve.jl: High-Performance and Robust Solvers for Systems of Nonlinear Equations in Julia*. en. arXiv:2403.16341 [cs, math]. URL: http://arxiv.org/abs/2403.16341 (visited on 2024-06-15).

Potra, FA (1984). "Nondiscrete induction and iterative processes". In: *Pitman Publ.*

Prigogine, Ilya and René Lefever (1968). "Symmetry breaking instabilities in dissipative systems. II". In: *The Journal of Chemical Physics* 48.4, pp. 1695–1700.

Rackauckas, Christopher and Qing Nie (2017). "Differentialequations.jl–a performant and feature-rich ecosystem for solving differential equations in julia". In: *Journal of Open Research Software* 5.1, p. 15.

Revels, Jarrett, Miles Lubin, and Theodore Papamarkou (2016). "Forward-mode automatic differentiation in Julia". In: *arXiv preprint arXiv:1607.07892*.

Sariman, Syahmi Afandi and Ishak Hashim (2020). "New optimal Newton-Householder methods for solving nonlinear equations and their dynamics". In: *Computers, Materials & Continua* 65.1, pp. 69–85.

SciML/LinearSolve.jl (2024-12). original-date: 2021-07-02T19:56:38Z. URL: https://github.com/SciML/LinearSolve.jl (visited on 2024-12-09).

Shampine, L. F. (2002-12). "Solving 0 = F(t, y(t), y'(t)) in Matlab". en. In: 10.4. Publisher: De Gruyter Section: Journal of Numerical Mathematics, pp. 291–310. ISSN: 1569-3953. DOI: 10.1515/JNMA.2002.291. URL: https://www.degruyter.com/document/doi/10.1515/JNMA.2002.291/html (visited on 2024-12-09).

Shampine, Lawrence F. and Mark W. Reichelt (1997-01). "The MATLAB ODE Suite". en. In: *SIAM Journal on Scientific Computing* 18.1, pp. 1–22. ISSN: 1064-8275, 1095-7197. DOI: 10.1137/S1064827594276424. URL: http://epubs.siam.org/doi/10.1137/S1064827594276424 (visited on 2024-12-09).

Singh, Harmandeep and Janak Raj Sharma (2023-02). "Simple yet highly efficient numerical techniques for systems of nonlinear equations". en. In: *Computational and Applied Mathematics* 42.1, p. 22. ISSN: 2238-3603, 1807-0302. DOI: 10.1007/s40314-022-02159-9. URL: https://link.springer.com/10.1007/s40314-022-02159-9 (visited on 2024-06-03).

Steihaug, Trond (2006). "Newton and Halley are one step apart". In: *Institute for Scientific Computing*.

Suleiman, Sara Tagelsir Mohamed (2009). "Solving System of Nonlinear Equations Using Methods in the Halley Class". MA thesis. The University of Bergen.

Tan, Songchen (2023a-06). "Higher-Order Automatic Differentiation and Its Applications". en. MA thesis. Massachusetts Institute of Technology. URL: https://dspace.mit.edu/handle/1721.1/151501.

Tan, Songchen (2023b-05). *TaylorDiff.jl*. original-date: 2022-11-09T17:21:07Z. URL: https://github.com/JuliaDiff/TaylorDiff.jl (visited on 2023-05-11).

Vladimirescu, Andre (1994-03). *The Spice Book*. USA: John Wiley & Sons, Inc. ISBN: 978-0-471-60926-1.

Walther, Andrea and Andreas Griewank (2009). "Getting Started with ADOL-C." In: *Combinatorial scientific computing* 1.06.02.

White, Frames et al. (2023-04). *JuliaDiff/ChainRules.jl: v1.49.0*. DOI: 10.5281/zenodo.7870094. URL: https://zenodo.org/record/7870094 (visited on 2023-05-11).

Xiao, Xiao-Yong (2022-09). "New techniques to develop higher order iterative methods for systems of nonlinear equations". en. In: *Computational and Applied Mathematics* 41.6, p. 243. ISSN: 2238-3603, 1807-0302. DOI: 10.1007/s40314-022-01959-3. URL: https://link.springer.com/10.1007/s40314-022-01959-3 (visited on 2024-05-17).

Xiao, Xiaoyong and Hongwei Yin (2015-08). "A new class of methods with higher order of convergence for solving systems of nonlinear equations". In: *Applied Mathematics and Computation* 264, pp. 300–309. ISSN: 0096-3003. DOI: 10.1016/j.amc.2015.04.094. URL: https://www.sciencedirect.com/science/article/pii/S0096300315005561 (visited on 2024-10-09).

DOI
10.3384/ecp218861

MultiEnergySystem: A Modelica Library for Dynamic Modeling and Simulation of District Heating and Gas Networks

Marcelo Muro[1] Guido Sassaroli[1] Riccardo Lazzari[1]

[1]Materials and Generation Technologies Department, Ricerca sul Sistema Energetico RSE S.p.A., Milan, Italy
{marcelo.muro, guido.sassaroli, riccardo.lazzari}@rse-web.it

Abstract

This paper presents the *MultiEnergySystem* Library (MESL), an open-source Modelica library designed for the dynamic simulation of integrated district heating and gas/hydrogen networks. MESL introduces a fully equation-based formulation of fluid properties, ensuring model interpretability and enabling compatibility with control and optimization applications. The library includes dedicated models for hydrogen blending studies and supports gas quality indicators such as Wobbe Index (WI), Higher Heating Value (HHV), and Specific Gravity (SG). District heating models have been experimentally validated using data from a real-world test facility, and these datasets are provided to support reproducibility. While MESL currently focuses on thermal and gas domains, planned expansions include a more comprehensive electrical package and experimental validation of hydrogen components. The paper describes the library's design and validation strategy, concluding with illustrative case studies and ongoing development plans.

Keywords: Modelica library, District Heating, Natural Gas, Hydrogen Blending.

1 Introduction

In recent years, the increasing complexity and integration of various energy systems—such as district heating and gas/hydrogen networks—driven by decarbonization objectives, have motivated the development of sophisticated modeling and simulation tools. Aligned with this context, one of the primary activities of our research department at RSE S.p.A. is the development of precise models to study phenomena within and across different energy systems, facilitating both operational and control analyses.

We are currently involved in two research projects that, while not directly correlated, share similar objectives. The first project focuses on applying novel operational and control approaches to integrate thermal and gas energy vectors with the electrical network, using detailed models that can be experimentally validated at the RSE Distributed Energy Resources Test Facility (DER-TF) (Gandolfi et al. 2022). The second project is dedicated to studying the integration of biogas and/or hydrogen into gas distribution networks, with particular attention to understanding the main dynamics of this phenomenon and its impact on gas quality. Both projects highlight the critical need for precise models and share the gas/hydrogen vector as a common energy domain, demonstrating the value of a unified modeling approach.

The Modelica programming language has emerged as a key modeling tool due to its a-causal, multi-domain, object-oriented nature, enabling dynamic and modular modeling of interconnected systems. Indeed, the use of Modelica for studying multi-energy networks has grown significantly. This growth has led to the creation of modeling libraries tailored for these applications, each offering different levels of complexity and focus.

Focusing on open-source Modelica libraries, the *DistrictHeating* library (Giraud et al. 2015) stands out for its emphasis on accurate and reliable District Heating and Cooling (DHC) simulations. More recently, the open-source *AixLib* library (Maier et al. 2024) has proven valuable for simulating large-scale district heating systems and their interaction with building energy systems. However, both libraries focus solely on DHNs and do not include gas or hydrogen modeling, limiting their applicability to multi-energy system studies.

The *ThermoPower* library (Casella and Leva 2006) instead, is one of the most established Modelica libraries for dynamic modeling of conventional thermal power plants. It contains models from multiple energy domains, including thermal and gas, and has been widely used over the past two decades (Casella and Leva 2024). The *TransiEnt* library (Senkel et al. 2021) is a more recent development designed for transient simulations of coupled energy networks, integrating district heating, gas, and hydrogen networks into a unified framework.

Despite their strengths, these libraries rely on external fluid property backends, either via the Modelica Standard Library or through compiled libraries such as *TILMedia* (GmbH et al. 2025). While these tools are effective for simulation, they limit symbolic manipulation, making them less suitable for applications involving control design, optimization, or sensitivity analysis, where full access to model equations is essential. Furthermore, these libraries lack dedicated tools for modeling hydrogen blending and gas quality indicators, such as Wobbe Index (WI), Higher Heating Value (HHV), and Specific Gravity (SG)—parameters that are central to assessing hydrogen integration feasibility and regulatory compliance. Fi-

nally, although many libraries have undergone theoretical validation, they often lack publicly available experimental datasets that allow direct comparison with real-world measurements. To address these needs, and inspired by previous libraries, we developed the *MultiEnergySystem* library (MESL).

MESL provides a comprehensive framework for simulating the dynamic behavior of integrated energy systems, focusing primarily on thermal and gas domains. A key feature of the library is its fully equation-based formulation of fluid properties, implemented entirely in Modelica. Unlike existing libraries such as *ThermoPower* or *TILMedia*, which rely on external function calls (e.g., C-based backends) for fluid property calculations, MESL uses symbolic equations to compute thermodynamic properties. This approach improves customizability, interpretability, and compatibility with symbolic-based applications such as optimal control.

The library also offers dedicated models for hydrogen blending studies, including the ability to compute gas quality indicators—such as WI, HHV, and SG—which are essential for evaluating the technical feasibility and regulatory compliance of hydrogen integration into gas networks. While the initial focus is on gas distribution systems where ideal-gas behavior is often assumed, MESL also incorporates real-gas models (e.g., Papay equation (Papay 1968)) to support advanced studies on transmission networks and gas mixing phenomena.

Another distinctive feature is the library's emphasis on experimental validation. MESL includes a set of district heating models that have been validated against real-world measurements obtained from a scaled-down District Heating Test Facility (DHTF) (Anderis, Muro Alvarado, and Lazzari 2024). These validation datasets are publicly available and integrated with the library, enabling reproducibility and fostering trust in simulation-based studies. Finally, MESL is released as an open-source Modelica library, ensuring accessibility for the research community and encouraging collaborative development.

Despite these features, some limitations remain. MESL currently includes includes only basic electrical modeling, allowing basic electric power calculations of components such as electric pumps, boilers, and combined heat-and-power (CHP) units. A more comprehensive electrical package is still under consideration, with future integration planned alongside existing Modelica libraries. Meanwhile, planned expansions of our DER-TF, which consist of a Gas Test Facility (GTF) implementation, will enable direct experimental validation of gas/hydrogen components; current validation is limited to measuring natural gas mass flow for thermal devices (e.g., boilers, CHP). Once realized, these additions will enhance the library's validated hydrogen models.

The primary goal of this study is to present MESL, describing its design, implementation, and validation, highlighting potential operational and control applications. The remainder of the paper is organized as follows: Section 2 introduces the library's structure, Section 3 explains the validation methodology, Section 4 presents selected study cases, and Section 5 concludes with an outlook on future work.

2 Library Overview

The *MultiEnergySystem* Library contains three main packages.

1. **`DistrictHeatingNetwork`**: All core components needed to model a district heating system.
2. **`H2GasFacility`**: Models for studying gas networks, including biomethane and hydrogen injection.
3. **`TestFacility`**: Various systems and plant configurations representative of RSE's test facilities, plus related validation models and data.

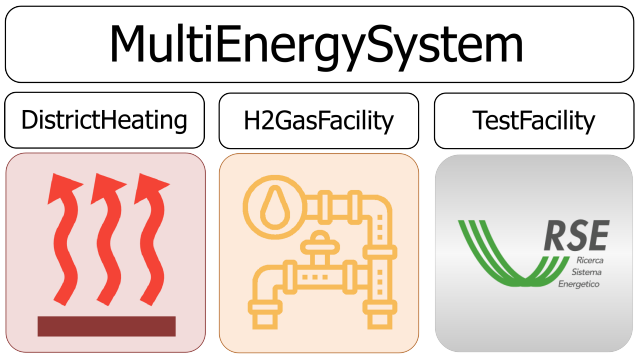


Figure 1. Main packages from MESL.

Within `DistrictHeatingNetwork` and `H2GasFacility`, sub-packages are arranged by functionality. For instance,

- `Interfaces` provides the interfaces to use for the specific energy vector.
- `Fluids` defines water for district heating and the various gas mixtures for gas networks,
- `Sources` contains ideal source and sink pressure and mass flow rate models.
- `Components` contains models like pipes, valves, and other thermo-hydraulic elements,
- `Tests` provides models for verifying the functionality of individual components.

These sub-packages appear in both `DistrictHeatingNetwork` and `H2GasFacility` packages, facilitating a consistent approach to component development. The `TestFacility` package, in contrast, follows its own structure, reflecting the setup of RSE's DHTF and related configurations; it also stores validation data and scenarios. Further details about `TestFacility` will be presented later in this section.

In the following subsections, we introduce each main package or MESL in more detail.

DOI
10.3384/ecp218871

2.1 DistrictHeatingNetwork

As mentioned previously, this package contains different components that can be used to model a district heating network. Because our focus is primarily on building these models for operational and control purposes, we have chosen certain simplifications. For example, the fluid is treated strictly as liquid water, without phase-change effects. In fact, it is possible to consider a constant specific heat capacity and constant density by the assumption that the system works at low pressures; in any case, it is also possible to find variations of the liquid water model with temperature-dependent specific heat capacity and constant density if the user requires this detail; however, this remains liquid water. Likewise, heat-generation devices such as boilers, CHP or chiller are assumed to be ideal outlet-temperature controlled. These assumptions reduce model complexity while still preserving the key dynamics required for control-oriented studies.

Typical components in `DistrictHeatingNetwork` include:

- **`Pipes/RoundPipe1DFV`**: Pipe model described by the traditional 1D finite-volume approach that implements mass, momentum, and energy balances, used for modeling the RSE's DHTF.
- **`Pipes/BrazedPlateHeatExchanger`**: Allow for heat transfer between heat generation and cooling systems.
- **`ThermalMachines/[ControlledGasBoiler,ControlledCHP]`**: Idealized outlet temperature-controlled units, including mass and energy balances as well.
- **`TurboMachines/ControlledPump`**: Support simple control modes if needed for optimization tasks (ideal mass-flow or pressure control pumps).

For further details, readers can refer to (Muro Alvarado et al. 2024).

2.2 H2GasFacility

The `H2GasFacility` package was developed to enable both the integration of gas networks with thermal energy systems and the study of decentralized alternative gas injection in gas networks.

2.2.1 H2GasFacility/Media

A key feature of this package is the custom gas model design. Although our primary focus is on gas distribution networks—where ideal-gas behavior is often assumed, we developed our own gas modeling approach to handle both low-pressure distribution networks and high-pressure transmission lines within a single framework. Rather than limiting ourselves to an ideal-gas assumption, we implemented a fully equation-based formulation that effortlessly switches between ideal and real-gas models. This approach enhances portability, interpretability, and compatibility with optimization and control applications, ensuring that the library can support a wide range of research needs, from conventional gas distribution to high-pressure pipelines and hydrogen blending studies.

At the core of this flexibility is the `PartialMixture` base model, located in the `Media/BaseClasses` sub-package. This foundational model defines essential variables such as pressure, temperature, and mass composition. By extending `PartialMixture`, users can implement custom gas mixture models by specifying the desired thermodynamic equations. The class diagram in Figure Figure 2 illustrates this structure. Each gas mixture model (e.g., `IdealMixture`, `PapayMixture`) defines its own equation of state, enabling tailored simulation of different gas behaviors within the same framework. From these models, users can then define specific gases by setting their composition and other relevant properties, enabling detailed and customizable gas modeling for a wide range of applications.

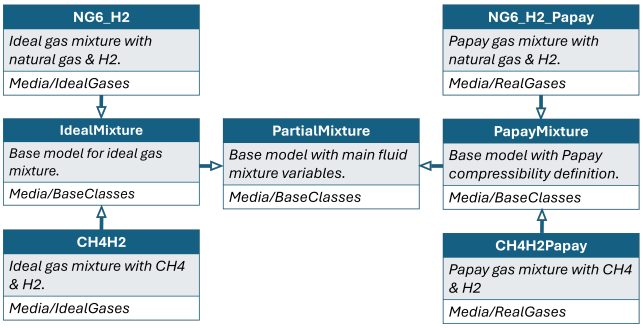


Figure 2. Class diagram for fluid models in `H2GasFacility` package.

2.2.2 H2GasFacility/Components

The `Components` package contains the fundamental building blocks for modeling gas networks. Among these, two key models stand out:

- **`Round1DFV`**: A detailed 1D finite-volume pipe model that implements mass, momentum, and energy balances, as well as the composition mass balance for gas mixtures. This level of detail is especially useful for research that requires transient or multi-species gas flow analysis.
- **`IdealUser`**: An idealized mass-flow consumer that can be placed within a gas network to represent generic demands or loads.

Additionally, `Components` includes the `Manifold` sub-package, which provides manifold and connector models that can facilitate initialization and improve overall network organization.

2.2.3 `H2GasFacility/Tests`

The `Tests` package contains validation models designed to assess gas quality parameters and the accuracy of component models. In particular, the `Subsystem` sub-package includes test models based on literature examples, ensuring consistency with existing studies on hydrogen blending and gas network behavior. Other sub-packages within `Tests` focus on verifying the functionality of individual components.

2.2.4 Considerations

In addition to predefined test cases, users can construct their own gas network models by combining fundamental components from the `Components` package. A basic gas network can be constructed by connecting an ideal pressure source to one or more `Round1DFV` pipes, then attaching one or more `IdealUser` blocks. All pipe connections must have defined inlet and outlet boundaries—either additional pipes, a gas source, or a user. Figure 3 shows the diagram view of one such medium-pressure gas network from (Cheli et al. 2021). Additional details about this test case will be covered in Section 4.

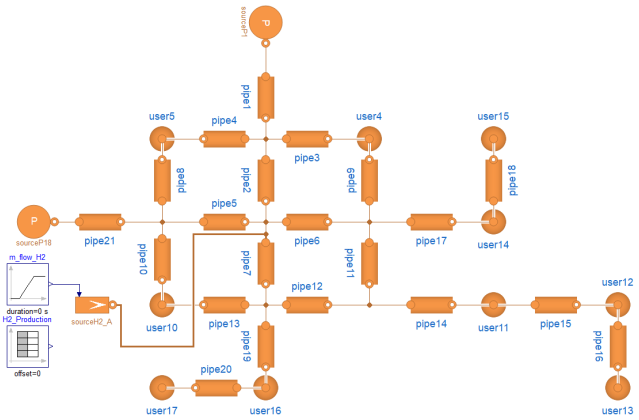


Figure 3. Modelica diagram view of medium-pressure gas network based on (Cheli et al. 2021).

In the modeling of extensive gas networks, it is crucial to carefully consider the initial values, especially for pressure. Although exact initialization is unnecessary, the starting pressure values should align with anticipations of network conditions. Poorly selected initial values may lead to initialization issues, potentially impacting the convergence of the model.

To determine feasible initial values, several approaches can be employed:

- Using real measured data from steady-state operation: When available, historical or real-time measurements from the gas network under steady-state conditions provide a reliable basis for initializing pressures and flows.
- Applying a trial-and-error method with iterative refinement: In the absence of reliable data, a manual approach may be adopted, starting with reasonable estimates followed by successive adjustments based on model feedback.
- Performing steady-state simulations using simplified frameworks: Employing less complex simulation tools (e.g., Pandapipes (Lohmeier et al. 2020)) to solve the network under steady-state assumptions can offer a practical way to generate plausible initial values.

2.2.5 Coupling with thermal energy vector

The `H2GasFacility` package can interface with the `DistrictHeatingNetwork` models by incorporating a gas connector into components such as gas boiler and CHP unit. This additional connector represents the fuel inlet (e.g., natural gas or hydrogen blends), enabling direct coupling between gas and thermal domains. This feature is important for conducting multi-vector energy simulations, allowing researchers to analyze the operational implications of hydrogen blending on both gas networks and thermal generation systems.

2.3 `TestFacility`

The `TestFacility` package contains models, systems, and full plant configurations that replicate the RSE's DHTF. It also includes the corresponding validation test cases. Because the DHTF is currently implemented while the GTF is still under development, most existing models focus on the district heating side of the library. Nonetheless, for demonstration purposes, a few gas-related study cases are already included. As soon as the GTF is operational, we plan to add experimental data and corresponding models to extend the library's scope on the gas/hydrogen side.

This package features more complex systems built from the basic models in the other two packages, forming subsystems and complete configurations of the DHTF. Users can find not only the validation tests within the DHTF environment, but also the experimental data employed in these validations. The sub-package `Resources` stores these datasets according to specific plant configurations. At present, most validations concentrate on centralized setups, but future work will incorporate diverse plant configurations to exploit the full capabilities of the DHTF.

Another important sub-package is `FMUExport`, which includes example models at both subsystem and full-plant scales. These classes enable the generation of FMUs for use in external environments like Simulink or Python, supporting advanced control strategies.

2.4 Availability & Software Compatibility

The MESL is developed as an open-source project and is publicly available at `https://github.com/RSE-TGM/multienergysystem`. The library has been primarily developed and tested in Dymola 2023, ensuring full compatibility with this Modelica environment. However, the

DOI
10.3384/ecp218871

`DistrictHeatingNetwork` package—and models using only its components—have been successfully tested in OpenModelica v1.24.3, demonstrating full functionality in this open-source platform.

On the other hand, the `H2GasFacility` package currently faces some limitations in OpenModelica. While basic gas models run correctly, advanced test cases involving multi-component gas flow present initialization issues. Future work will focus on improving OpenModelica compatibility to enhance accessibility and usability across multiple platforms.

3 Validation

Prior to this work, we had already validated the core district heating components in the `DistrictHeatingNetwork` package—such as boilers, heat exchangers and water pumps—using the RSE's DHTF. Over a series of experiments, we verified each model's accuracy under diverse operating conditions, measuring temperature, pressure, and flow rates. In the most comprehensive configuration (centralized layout), the models achieved a CVRMSE below 18% and an NMBE below 8%, in accordance with ASHRAE Guideline 14 (Muro Alvarado et al. 2024). These results confirm that the library reliably captures the dynamics in a district heating environment and can thus be extended to other energy vectors.

To enable the interconnection with gas networks, we updated specific models from the `DistrictHeatingNetwork` main package, by adding an additional gas inlet connector, drawn from `H2GasNetwork.Interfaces.Connectors` subpackage. In particular, two heat-generation models—the gas boiler and the CHP unit—were extended to calculate gas mass flow rates using Equation 1:

$$w_g = \frac{P_c}{HHV_g \eta_c} \tag{1}$$

where w_g is the gas mass flow (kg/s), P_c is the combustion power (W), HHV_g is the higher heating value of the gas (J/kg), and η_c is the combustion efficiency.

For the computation of the gas mass flow rate, the value of P_c depends on the specific component. In the gas boiler, the combustion power is entirely converted into thermal output, so P_c directly depends on the required thermal power. In the CHP model, P_c can be computed based on either the required thermal power or the required electrical power, depending on the selected control mode configuration.

This connector-based approach allows each gas-consuming component to be directly linked to a gas network model, thereby facilitating a coupling between thermal and gas domains.

To validate these hybrid models, we compared simulated results with experimental measurements from the DHTF. Figure 4 illustrates the test model diagram for a gas boiler configuration. The main boundary conditions—inlet temperature and water mass flow rate—were imposed using measured data (Figure 5.a). For evaluation purposes, we focused on two key variables: the outlet temperature of the boiler system, representing the thermal energy vector, and the fuel mass flow rate, representing the gas energy vector. Although Figure 5.b shows the outlet temperature trend, the performance metrics (CVRMSE & NMBE) were computed based on the temperature difference across the boiler (i.e., outlet minus inlet).

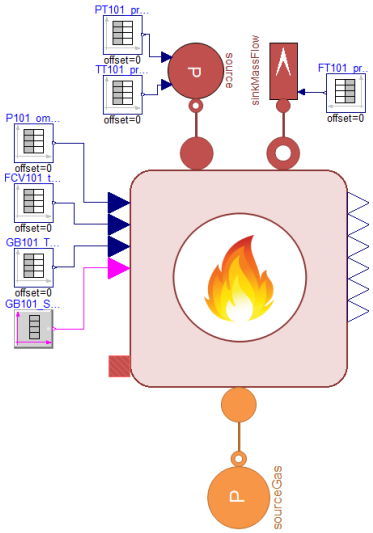
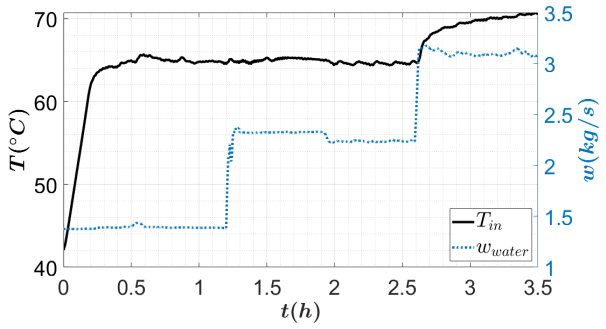


Figure 4. Validation test model for gas boiler subsystem.

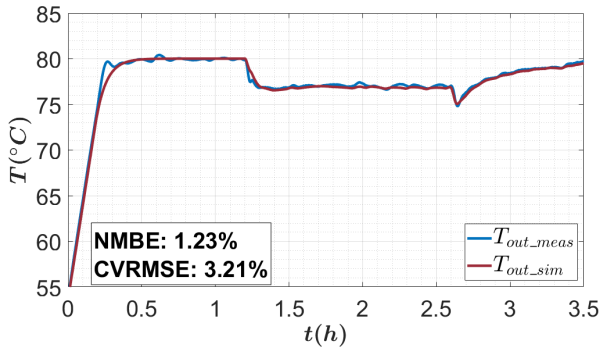
The validation results show good alignment between measured and simulated values: the temperature difference achieved an NMBE of 1.23% and a CVRMSE of 3.21%, while the fuel mass flow rate (Figure 5.c) resulted in an NMBE of -0.14% and a CVRMSE of 2.43%. These results demonstrate both strong accuracy and practical suitability of the model for integrated gas-thermal simulations.

Regarding gas network modeling, one of our primary goals is accurately capturing gas quality parameters (e.g., Wobbe Index) when hydrogen is blended into natural gas networks. We reference (Cheli et al. 2021) for such scenarios, which, although focused on quasi-steady operating conditions, provide a sufficient number of hourly data points to validate our model's compositional calculations. Figure 6 illustrates a representative comparison, showing minimal deviations from the reference data. While (Cheli et al. 2021) does not explore fast transients, our simulation framework allows for the generation of continuous hydrogen injection profiles. This feature becomes particularly useful for investigating faster ramping conditions and transient behaviors beyond the temporal resolution of the reference data.

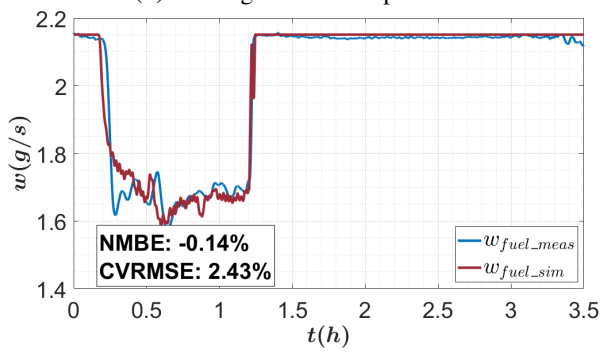
For more rapid transients (seconds to minutes), we have compared the library's performance with other validated tools—such as the Modelica Standard Library (Fritzson 2011)—yielding negligible discrepancies. Once the



(a) Main boundary conditions for the test



(b) Outlet gas boiler temperature.



(c) Fuel mass flow rate.

Figure 5. Measured and simulated profiles in a gas boiler subsystem validation test case.

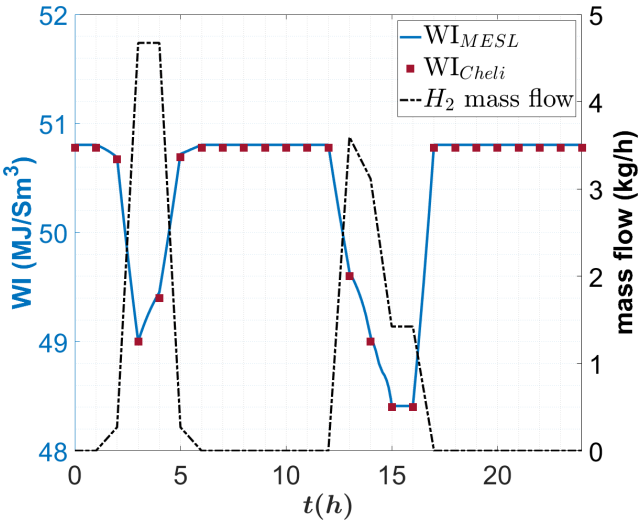


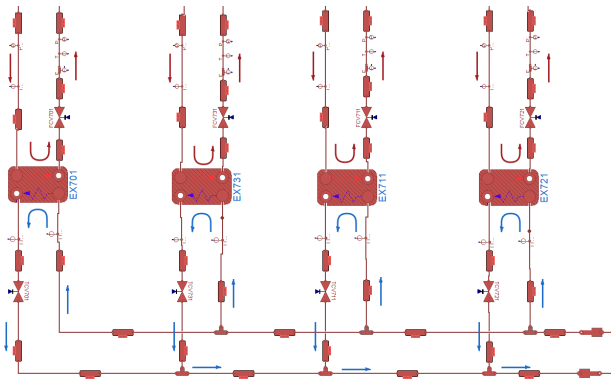
Figure 6. Comparison with (Cheli et al. 2021).

RSE's GTF is fully implemented, additional experimental validation will be performed to improve confidence in these models under dynamic hydrogen blending scenarios.

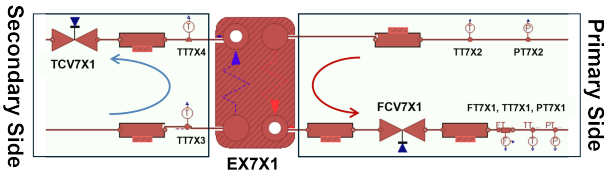
4 Case Studies

4.1 DHN application

One of the first applications of our library was the development of a decentralized control system for the load section of the DHTF. The load system consists of four heat exchangers (each representing a user in a DHN), connected on one side to a centralized heat distribution system (primary side) and on the other to a cooling system (secondary side) (Figure 7.a). Each user is equipped with pressure, temperature and flow sensors on both the primary and secondary sides, as well as two controllable valves: a flow control valve on the primary side and a temperature control valve on the secondary side (Figure 7.b). By adjusting these valves, the system emulates different heat demand profiles, closely replicating real DHN user behavior.



(a) Full Load system model.



(b) Single Load model.

Figure 7. DHTF Load System using MESL.

Traditionally, centralized DHN controllers regulate the secondary-side outlet temperature to ensure user comfort (Buffa et al. 2021). However, in this study, we adopt a different approach, prioritizing thermal power control at each heat exchanger. This strategy enables more precise energy demand regulation, allowing predefined load profiles. Moreover, it aligns with advanced control methodologies such as Model Predictive Control (MPC), where load thermal power profiles serve as key inputs for optimizing supply temperatures and mass flow rates from the heat generation side. A secondary objective is return temperature regulation on the primary side, ensuring compliance with the operational constraints of heat generators. Consequently, the primary variables of interest are

DOI
10.3384/ecp218871

the *thermal power exchanged* at each heat exchanger and the *return water temperature* on the primary side.

To achieve these objectives, we designed a decentralized control system. We used our library to develop an open-loop analysis of the load system in a centralized configuration and verified that (i) Primary-side valve control is best suited for thermal power regulation since mass flow directly governs power exchange, (ii) Secondary-side valve control is better for regulating primary-side return temperature and (iii) the system can be decomposed into independent Single-Input Single-Output (SISO) loops, allowing a decentralized control strategy. For simplicity, the detailed analytical steps of this analysis have been omitted from this paper.

The decentralized control system for the load system, developed using MESL, was validated both in simulations and experimentally in the DHTF. As shown in Figure 8, the final design consists of:

- Four independent PI controllers for return temperature control (secondary-side valves).
- Four cascade controllers for thermal power control (primary-side valves), with a PI controller in the primary loop and an integrator in the secondary loop.

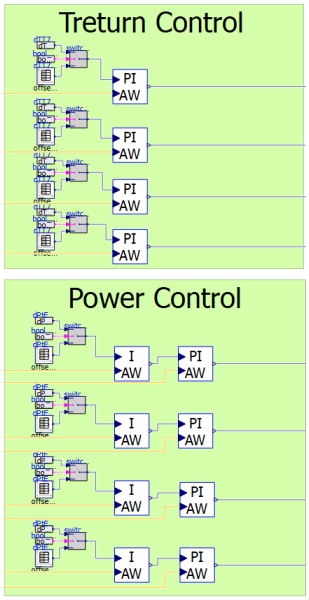
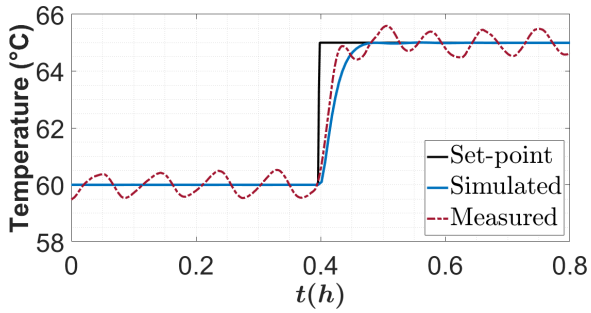
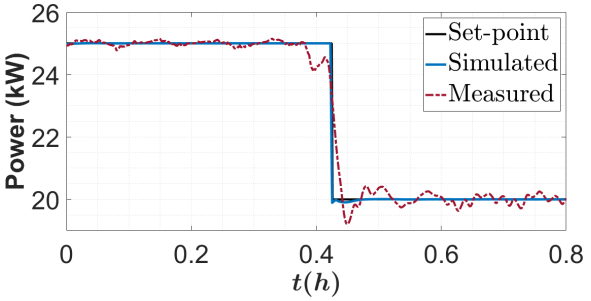


Figure 8. Controller diagram in Modelica.

A comparison between simulated and experimental results for a single load is shown in Figure 9. Trends for both return temperature (Figure 9.a) and thermal power (Figure 9.b) demonstrate the control system's ability to regulate desired variables effectively. While the overall response is well captured, some discrepancies appear, particularly in the return temperature dynamics. These deviations are primarily due to the cooling system's chiller: in real operation, the chiller follows an on/off control



(a) Return temperature.



(b) Thermal exchanged power.

Figure 9. Example of the controlled variables of one load.

approach, whereas in the simulation, an ideal temperature control assumption was used for simplicity. Despite these differences, the experimental results confirm that the control system successfully regulates the load dynamics, demonstrating the effectiveness of MESL in developing and validating customized control strategies for district heating applications.

Beyond the decentralized control implementation described above, the package *DistrictHeatingNetwork* has also been used in the design and experimental validation of a MPC strategy for district heating networks. In that study, MESL was employed to simulate the DER-TF environment and support controller tuning. For further details, see (Nigro et al. 2025). This application reinforces the suitability of the library not only for dynamic simulation but also to support advanced control design.

4.2 Integration of Thermal and Gas Domains

As mentioned previously, the gas models within our library have not yet been experimentally validated in the DER-TF. However, given the satisfactory simulation results obtained so far with academic results, we aim to showcase the potential of integrating district heating and gas networks using the developed tools. Specifically, we demonstrate how thermal and gas systems interact and how simulation-based approaches can help predict key network parameters in scenarios involving hydrogen injection.

To illustrate this, we constructed a study case that couples an existing system from the DHTF—the gas boiler system (previously introduced in Figure 4)—with the gas distribution network model described earlier (Figure 3).

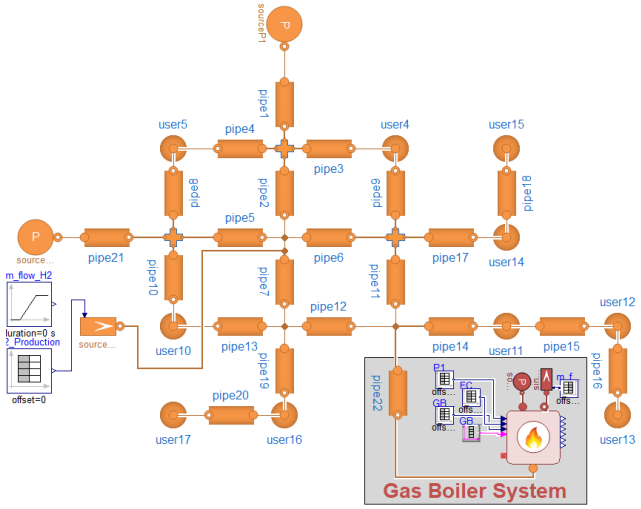
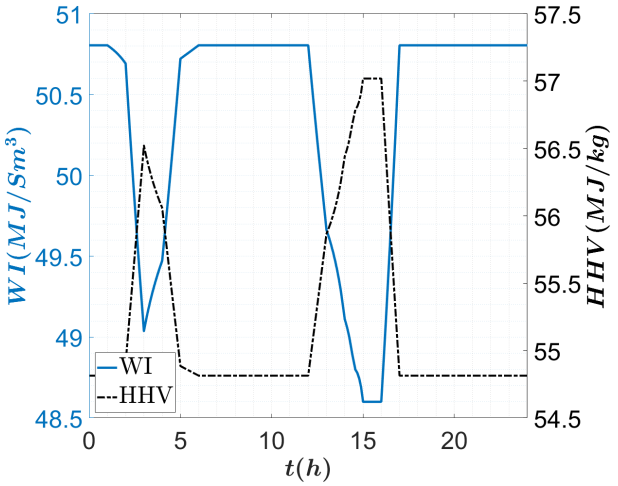
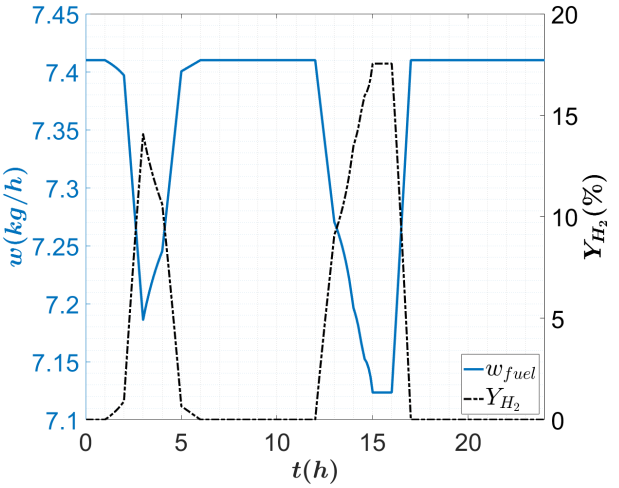


Figure 10. Diagram View of gas boiler system model from the DHTF, integrated in the case study based on (Cheli et al. 2021).



(a) Gas quality indexes.



(b) Fuel flow and hydrogen content.

Figure 11. Time evolution of key gas quality and fuel flow variables.

In this scenario, the gas boiler system is treated as an additional user in the gas network, alongside the existing users. The diagram of this configuration is described in Figure 10. The goal is to assess how the gas composition and flow characteristics evolve when the same hydrogen injection profile used in Figure 6 is applied.

The initial fuel composition consists of 97.20% CH_4, 1.80% C_2H_6, 0.30% C_3H_8, 0.10% C_4H_{10}, 0.10% CO_2, and 0.50% N_2 (vol%). The nominal fuel mass flow rate required by the gas boiler system, once connected to the network, is 7.41 kg/h. As hydrogen injection occurs throughout the day, this mass flow rate dynamically adjusts in response to the changing gas mixture composition, according to Equation 1.

Figure 11 illustrates the impact of hydrogen injection on fuel consumption. As hydrogen enters the system, the fuel mass flow rate decreases (Figure 11.b, blue line), which is expected since the HHV of the gas mixture increases (Figure 11.a, black line). At hour 15, the hydrogen volume fraction in the gas mixture reaches approximately 17.5% vol(Figure 11.b, black line), remaining within the manufacturer's specified limit of 20% vol. In this hypothetical scenario, the gas boiler operates safely within its design constraints despite the hydrogen injection.

This simulation not only provides insights into how hydrogen blending affects gas demand, but also highlights potential operational constraints—not only for the pipeline network but also for end-user equipment, which must be evaluated for hydrogen tolerance and performance limitations.

Complementary to this case, the *MultiEnergySystem* library has also been used in a separate study focused specifically on decentralized hydrogen injection in natural gas distribution networks (Sassaroli, Muro, and Casamassima 2025). That work leveraged MESL to develop a detailed dynamic model that captures transient effects in pressure, temperature, and gas mixing, including hydrogen and biomethane injection scenarios. The case study used a real-world Italian gas distribution network, further demonstrating the flexibility and applicability of the library to current and future hydrogen integration challenges.

5 Conclusion and Future Work

This paper presented the *MultiEnergySystem* Library, a Modelica-based library for modeling, simulating, and analyzing district heating and gas network. The library includes experimentally validated district heating models and a gas modeling approach capable of handling ideal (and potentially real) gas formulations for the study of hydrogen blending.

A key aspect of MESL is its ability to bridge district heating and gas domains, facilitating studies on hydrogen injection impacts and coupled energy networks. The library was successfully used in two case studies: (i) the design and validation of a decentralized load control sys-

DOI
10.3384/ecp218871

tem for the DHTF, and (ii) the integration of a gas boiler system into a hydrogen-injected gas network. These applications demonstrate MESL's versatility for both operational and control-oriented studies.

Despite these advancements, some limitations remain. While district heating models have been experimentally validated, the gas/hydrogen components are still pending validation in RSE's upcoming Gas Test Facility. Additionally, MESL currently lacks a dedicated electrical network modeling package, though electric power parameters can already be estimated for key components such as water pumps, CHP unit, and electric boiler.

Future work will focus on enhancing the library's capabilities. Once the Gas Test Facility is operational, we plan to experimentally validate the gas models, particularly for hydrogen blending effects. Moreover, we aim to expand MESL's electrical modeling, potentially by integrating it with existing Modelica libraries. Finally, the library will be further developed so the models can be used for advanced control applications, exploring co-simulation approaches with tools such as Python or Simulink.

By continuously improving MESL, we hope to provide a comprehensive and open-source simulation tool for multi-energy systems, fostering research and innovation in district heating, gas networks, and hydrogen integration.

Acknowledgements

This work has been financed by the European Union – NextGeneration EU from the Italian Ministry of Environment and Energy Security POR H2 AdP MMES/ENEA with involvement of CNR and RSE, PNRR - Mission 2, Component 2, Investment 3.5 "Ricerca e sviluppo sull'idrogeno".

References

Anderis, Claudio, Marcelo A. Muro Alvarado, and Riccardo Lazzari (2024). "Implementation of an Experimental Facility for District Heating Networks Flexibility Assessment". In: *2024 AEIT International Annual Conference (AEIT)*, pp. 1–6. DOI: 10.23919/AEIT63317.2024.10736713.

Buffa, Simone et al. (2021-01). "Advanced control and fault detection strategies for district heating and cooling systems—A review". In: *Appl. Sci. (Basel)* 11.1, p. 455. DOI: 10.3390/app11010455.

Casella, Francesco and Alberto Leva (2006). "Modelling of thermo-hydraulic power generation processes using Modelica". In: *Mathematical and Computer Modelling of Dynamical Systems* 12.1, pp. 19–33. DOI: 10.1080/13873950500071082.

Casella, Francesco and Alberto Leva (2024). "Modelling and Simulation of Thermal Power Generation Systems with Modelica: 20 Years of Experience with the ThermoPower Library". In: *IFAC-PapersOnLine* 58.13. 12th IFAC Symposium on Control of Power and Energy Systems - CPES 2024, pp. 227–234. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2024.07.487.

Cheli, Lapo et al. (2021). "Steady-state analysis of a natural gas distribution network with hydrogen injection to absorb excess renewable electricity". In: *International Journal of Hydrogen Energy* 46.50, pp. 25562–25577. ISSN: 0360-3199. DOI: 10.1016/j.ijhydene.2021.05.100.

Fritzson, P. (2011). "The Modelica Standard Library". In: *Introduction to Modeling and Simulation of Technical and Physical Systems with Modelica*. John Wiley & Sons, Ltd. Chap. 5, pp. 157–168. ISBN: 9781118094259. DOI: 10.1002/9781118094259.ch5.

Gandolfi, Chiara et al. (2022). "MISSION Project: Design of MV/LV hybrid AC/DC smart-energy grid". In: *2022 AEIT International Annual Conference (AEIT)*, pp. 1–6. DOI: 10.23919/AEIT56783.2022.9951813.

Giraud, Loic et al. (2015-09). "Presentation, Validation and Application of the *DistrictHeating* Modelica Library". In: *Proceedings of the 11th International Modelica Conference, Versailles, France*, pp. 79–88. DOI: 10.3384/ecp1511879.

GmbH, TLK-Thermo et al. (2025). *TILMedia: Thermophysical Property Library for Modelica*. https://github.com/ClaRaLibrary/TILMedia. Accessed: 2025-06-25.

Lohmeier, D. et al. (2020). "Pandapipes: An Open-Source Piping Grid Calculation Package for Multi-Energy Grid Simulations." In: *Sustainability*, vol. 12, p. 9899. DOI: 10.3390/su12239899.

Maier, Laura et al. (2024). "AixLib: an open-source Modelica library for compound building energy systems from component to district level with automated quality management". In: *Journal of Building Performance Simulation* 17.2, pp. 196–219. DOI: 10.1080/19401493.2023.2250521.

Muro Alvarado, Marcelo Andre et al. (2024). "Development and Experimental Validation of an Open-Source Model Library for District Heating Network Simulation". In: *2024 Open Source Modelling and Simulation of Energy Systems (OSMSES)*, pp. 1–6. DOI: 10.1109/OSMSES62085.2024.10668994.

Nigro, Lorenzo et al. (2025). "Experimental validation and design of nonlinear predictive control and moving horizon estimation for district heating". In: *Proceeding of IFAC Workshop on Smart Energy Systems for Efficient and Sustainable Smart Grids and Smart Cities*. to be published.

Papay, J. (1968). "ATermelestechnologiai Parameterek Valtozasa a Gazlelepk Muvelese Soran". In: *OGIL MUSZ*, pp. 267–273.

Sassaroli, Guido, Marcelo Muro, and Vincenzo Casamassima (2025). "Dynamic modeling and simulation of gas distribution networks with decentralized hydrogen injections". In: *Proceeding of IFAC Workshop on Smart Energy Systems for Efficient and Sustainable Smart Grids and Smart Cities*. to be published.

Senkel, A. et al. (2021). "Status of the TransiEnt Library: Transient Simulation of Complex Integrated Energy Systems". In: *Linköping Electronic Conference Proceedings*. DOI: 10.3384/ecp21181187.

DOI
10.3384/ecp218871

Calibration of a Chiller Modelica model with experimental data

Christophe Montsarrat[1] Pascal Borel[1] Ana Paez[1]

[1]EDF R&D, France, firstname.lastname@edf.fr

Abstract

The modelling and calibration of industrial chillers can be challenging, especially when few measurements with a large uncertainty are available. The use of an approximate calibration model is investigated to explore its potential in calibrating a chiller model, constructed in *Modelica*. Based on the results of a well-instrumented experimental rig of a chiller at the EDF laboratory recorded over five days, with a varying outdoor air temperature, an approximated steady analysis is carried out by calibrating the *Modelica* model with the experimental results (mass flow rate, temperatures, pressures). The results indicate a good match between the calibrated model and the experiments and a physical sensitivity in the parameters of the model, consistent with the operation of the machine (isentropic efficiency, compressor flow rate, evaporator overheat regulation). Further work can be done on the unsteady measurements recorded on the chiller to improve the model and its stochastic calibration.

Keywords: Chiller modelling, Approximate Bayesian Calibration, Modelica, R507A

1 Introduction

Chillers play a key role in various industries and notably within nuclear power: they enable to condition air that cools down electrical rooms, which are essentials for plant operation and safety. It is thus required to have numerical models that can predict, with required accuracy, the key quantities of these systems, such as the maximum available cooling power under sizing conditions or temperature of the chilled water produced.

Several approaches exist to model these complex systems. A first approach relies on analytical models, often based on a steady-state assumption for the system. It is used to get a quick evaluation of the system behavior, but accuracy is often poor and a large data-set is generally needed to calibrate the model as exhibited by (Cecchinato, Chiarello, and Corradi 2010).

Another model category is the zonal model which consists in establishing a set of equations for each main components of the chiller such as the evaporator, the condenser, the expansion valves and the compressor. Solving the obtained equations systems, either assuming steady state or studying dynamic response, enable to predict the detailed behavior of the system as proposed by (Mouneer, Aly, and Mina 2021), (Afzali and Mahalec 2017) or (Lissandrin et al. 2017).

Computational Fluid Dynamics (CFD) can be used to study the details of the flows within some parts of the chiller. However, this approach is generally not affordable at the integral scale.

Calibration of chiller models is a key aspect of the zonal or analytical as the model relies on parameters that generally cannot be directly deduced from geometry and associated input data of the system. On a design phase, correlations are generally used to compute the required coefficients for the model to run (e.g. heat exchange coefficients). Once the system is in operation, data assimilation techniques can be used to calibrate model parameters. Various techniques can be used to calibrate models: amongst most used methods are least square methods (linear or non-linear), to minimize the differences between observed data and model prediction. In a more general way, Bayesian calibration techniques can be set up to estimate the posterior distribution of the parameters to be calibrated on a given dataset as described by (Marin, Robert, et al. 2007). (Zhen, Niu, and Tian 2023) propose to calibrate a chiller analytical model from the Modelica Buildings library (Wetter et al. 2014) by optimizing the improved indicator of the mean normalized bias error of model predictions.

The primary objective of this study is to calibrate a zonal Modelica chiller model using experimental data obtained from experiments conducted at the EDF laboratory.

First, a description of the Modelica-based chiller models is provided, along with a justification for the approaches used to model the different components of the chiller. Next, the rationale behind the statistical calibration techniques employed is presented, followed by an overview of the experimental facility used for the experiments.

The results from the experiment, considered to be steady, are discussed and then compared with the model predictions resulting from the calibration process. Discussions on the significance of the measurement uncertainties are also addressed.

2 Theory

2.1 Nodal modelling of a chiller

The modelling of a chiller is based on three components developed through an in-house library in `Modelica`. The motivation is to use components as generic as possible that can be adapted to different configurations of chillers without relying too much on geometric characteristics. As

the motivation is to develop a methodology for calibrating chillers, reducing the number of coefficients for these three main components is essential. The main principles of the components used are quickly presented here:

- **Compressor** - it is modelled by four values: one volumetric flow rate Q_v and three efficiencies quantifying its performance. The three efficiencies quantify the different losses encountered on a compressor: the mechanical efficiency η_{mech} for the friction losses, the volumetric efficiency η_{vol} measuring how efficiently used is the volume sucked in (dead volume, impermeability, pressure loss) and the isentropic efficiency η_{is} quantifying the entropy increase due to the compression of the fluid. These four parameters are important to model the enthalpy increase resulting from the compression rate $\Pi = p_{hp}/p_{lp}$ applied:

$$\Delta h = f(\Pi, Q_v, \eta_{mech}, \eta_{vol}, \eta_{is}) \tag{1}$$

- **Heat exchanger** - the model consists of N discrete cells in which heat exchanges occur between a fluid 1, not changing state, and a fluid 2 that absorbs or releases heat both through sensible exchange and change of state. The exchangers modelled can be parallel or counter flow exchangers, with fluids moving in the same or opposite directions. The cell $i-1$ supplies the inlet conditions for the following cell i (phases, pressure, enthalpy, mass flow rate) and three heat transfer coefficients are used: K_{lat} for the latent heat exchange, $K_{sens,vap}$ and $K_{sens,cond}$, respectively for the sensible exchanges between the vaporized fluid 2 and fluid 1 and the condensed fluid 2 and fluid 1. In the modelling here considered, **these three coefficients are kept constant** and used as parameters for the calibration. For each cell i, the heat exchange q_i per area units is modelled with the following expression:

$$q_i = K\,(T_{1,i} - T_{2,i}) \tag{2}$$

  The heat flux applied to the cell updates the temperatures at the outlet of the cell i for each fluid (1 and 2) which are then used as the inlet temperatures of the following cell. Depending on the state of the flow (two-phase, vapor or liquid), the coefficients K_{lat}, $K_{sens,vap}$ or $K_{sens,cond}$ respectively apply in (2).

- **Expansion valve** - it is simply modelled by a relationship between the mass flow rate and the pressure difference across the valve. The relationship is the following, with p_{hp} the high-pressure at the inlet and p_{lp} the low pressure at the outlet:

$$\dot{m} = K_{valve}\sqrt{p_{hp} - p_{lp}} \tag{3}$$

  A dependency on the coefficient K_{valve} can be added to adjust the opening of the valve, for example using the overheat, i.e. the temperature difference between the outlet of the evaporator and the saturation temperature within it.

Figure 1 shows a generic chiller model as used in `OpenModelica` with four main components, representing a classical thermodynamic cycle for a chiller: compression, condensation, expansion and evaporation. The overheat regulation is indicated by the arrow starting at the outlet of the evaporator to regulate the opening of the expansion valve.

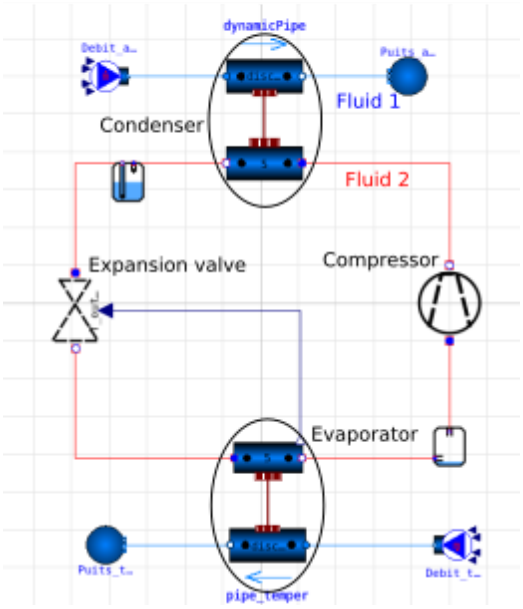


Figure 1. Illustration of a generic chiller model used in the `OpenModelica` GUI.

2.2 Approximate Bayesian Calibration

The purpose of calibration is to estimate the posterior mutltivariate law of the parameters θ to be calibrated of a model function denoted g. This model, $g(\theta, X)$, predicts observed quantities Y based the parameters to be calibrated and observed input parameters X.

$$Y = g(\theta, X) \tag{4}$$

As implied by the Bayes theorem recalled in Equation 5, the estimation of the posterior distribution of the parameters $\pi(\theta|Y)$ usually requires to be able to express the likelihood $\mathscr{L}$ of the observed variable Y, given the prediction of the model $\hat{Y} = g(\theta, X)$.

$$\pi(\theta|Y) = \frac{\mathscr{L}(Y|\theta)\pi(\theta)}{\pi(Y)} \tag{5}$$

A usual approach is to assume that the discrepancies between model prediction $\hat{Y}$ and the measured variable Y,

DOI 10.3384/ecp218881

denoted ε, follow a multivariate gaussian law with a null mean. If the errors are homoscedastic, non correlated and the model $g(\theta,X)$ is linear or locally linear using Taylor expansion, one can apply the Gauss-Markov theorem to obtain the ordinary least square (OLS) estimator $\hat{\theta}$ and its associated variance covariance matrix Σ. Assuming the likelihood function $\mathscr{L}(Y|\theta)$ is Gaussian, the posterior distribution can be expressed as:

$$\pi(\theta|Y) = \mathscr{N}(\hat{\theta},\Sigma) \tag{6}$$

In both linear and nonlinear Gaussian calibration approaches, it is further assumed that the prior distribution $\pi(\theta)$ is multivariate Gaussian. These methods rely on the Gaussian nature of the likelihood and require that the residuals—i.e., the differences between model predictions and observations—have zero mean. This implies that discrepancies are attributed solely to measurement uncertainties, rather than model bias or structural errors.

When the likelihood $\mathscr{L}(Y|\theta)$ has no analytical form, a solution is to use likelihood-free techniques such as the approximate bayesian calibration (ABC method), as suggested by (Turner and Zandt 2012) or (Cant and Evins 2023). This method mainly rely on the definition of a metrics to compare the simulated quantities $\hat{Y}$ to the same observed quantities Y. A threshold ε_0 is then associated to this metrics to consider a candidate point θ producing calibrated quantities. This is equivalent to considering that if the candidate point $\hat{\theta}$ produces predictions close enough to the observed values, then this point has a non-zero probability of being in the posterior parameter distribution $\pi(\theta|Y)$ as illustrated in Equation 7.

$$\pi(\theta|Y) \approx \pi(\theta|\rho(\hat{Y},Y) \leq \varepsilon_0) \tag{7}$$

This approach can also accommodate the presence of model error by specifying a probability density function f_ε to represent it. Using a fixed acceptance threshold, as described above, is equivalent to assuming that the model error follows a uniform distribution. In this case, all discrepancies within the threshold are considered equally likely, regardless of their magnitude. A common metrics used is the sum of the squared errors between the predictions and the observations.

$$RMSE_{Y_i} = \sqrt{\frac{1}{n_{obs}} \sum_{observation} (Y_i^{pred} - Y_i^{obs})^2} \tag{8}$$

$$MBE_{Y_i} = \frac{1}{n_{obs}} \sum_{observation} (Y_i^{pred} - Y_i^{obs}) \tag{9}$$

$$CvRMSE_{Y_i} = \frac{RMSE_{Y_i}}{\overline{Y_{obs}} - Y_{ref}} \tag{10}$$

$$NMBE_{Y_i} = \frac{MBE_{Y_i}}{\overline{Y_{obs}} - Y_{ref}} \tag{11}$$

Common threshold to retain a candidate point is to have $CvRMSE \leq 0.2$ and $NMBE \leq 0.05$ for each observed variable.

Using OpenTURNS (Baudin et al. 2017), a candidate set sample is generated from the multivariate distribution $\pi(\theta)$. Then, for each candidate point, the model g is evaluated allowing the defined metrics to be computed. The multivariate law of the posterior sample is then inferred using parametric or non parametric methods to define notable the most probable posterior values of $\pi(\theta|Y)$. If the posterior sample does not contain enough points regarding a targeted size previously defined, additional samples from the prior distributions can be considered. The above presented algorithm is usually called the basic ABC-rejection algorithm which can be improved using sequential MonteCarlo approach.

3 The R507 Chiller Experiment

3.1 Global Description

The test facility (Figure 2) used in this study operates with R507A, a hydrofluorocarbon (HFC) refrigerant (CoolProp n.d.). The test facility is well-equipped with pressure and temperature sensors installed at the inlet and outlet of each component, complemented by several Coriolis flow meters integrated into the test facility system.

The test facility consists of three main circuits. The first circuit, using Temper 55 as the secondary fluid in the evaporator, is designed to measure the cooling demand. This circuit includes electrical resistances and a pump that allow for the control of flow rate, temperature, and thermal power dissipation. A second air circuit interacts with the condenser to dissipate heat from the refrigerant. The third circuit utilizes R507A as the working fluid and forms the core of the experimental setup.

The refrigeration system consists primarily of a plate evaporator with a surface area of 44.4 m², a condenser of 1157.5 m² equipped with five variable-speed fans, an electronic expansion valve, and a Grasso rotary screw compressor. The compressor allows for variations in the internal swept volume. Depending on the applied compression ratio, the refrigerant flow rate can range from 23.1 m^3/h to 231 m^3/h.

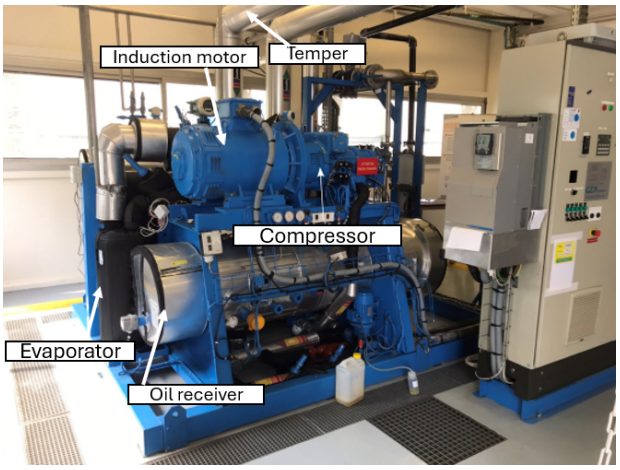


Figure 2. Photo of the test facility

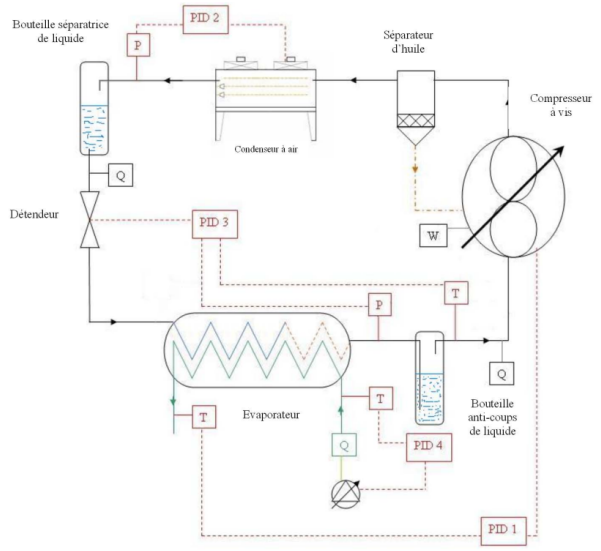


Figure 3. Experimental setup with control loop

Figure 3 shows the four controllers (PID1, PID2, PID3, and PID4) used to regulate the precision of the evaporator, high pressure, overheating, and the secondary pump flow rate of the evaporator circuit, respectively. These regulations will not be considered in the current model.

3.2 Uncertainty of measurements

The expanded uncertainty of each sensor considers the whole measuring chain, from the sensor to the automate errors. The uncertainty given corresponds to a confidence interval of 95%.

The mass flow rate, pressure, and temperature sensors have an accuracy at $\pm 0.1\%$, $\pm 3\%$, and $\pm 0.174°C$, respectively. It must however be noted that the uncertainties on the measurements on the air-side of the condenser are higher and more difficult to estimate.

When a calculation includes several variables, for example the calculation of the cooling capacity, a combined standard uncertainty U_c^2 should be calculated using the equation 12. The method is based on first-order Taylor series, and it is referred to as the law of propagation of uncertainty (Taylor, Barry N and Kuyatt 1994).

$$U_c^2(y) = \sum_{i=1}^{n} \left(\frac{\partial f}{\partial x_i} \right)^2 u^2(x_i) + \sum_i \sum_j \partial_i f \partial_j f r_{ij} u_i u_j \quad (12)$$

where $u(x_i)$ represents the uncertainty of each variable. In this article, the variables are assumed to be independent and with a linear behavior, which allows to neglect the last term of the previous equation that refers to the correlation between the variables.

3.3 Experimental conditions

The experimental results presented in this study were obtained under the following conditions:

- The flow rate of the Temper-55 fluid was maintained at 6 kg/s using the circuit pump. The evaporator pressure was kept constant at 7.5 bar through compressor regulation. To simulate a cooling demand, electrical resistances installed in the Temper-55 circuit provided a constant thermal load of 100 kW.
- The refrigerant overheat was regulated by the expansion valve, ensuring a stable overheat of 6°C. The five fans of the air-condenser operated continuously at maximum speed, allowing the refrigerant's high pressure to vary according to the ambient air temperature.
- Data were recorded every 10 seconds over a period of five days.

4 Results and discussions

4.1 Experimental results

The analysis of the results revealed significant oscillations primarily due to the ambient air temperature and the simultaneous regulation of four controllers, see section 3.1. While these oscillations could provide valuable insights in a future phase, particularly when incorporating regulation into our models for dynamic comparisons with experimental data, our current study focuses on a steady-state analysis.

In order to validate and calibrate the models in steady-state conditions, the experimental results were divided into multiple intervals where the ambient temperature variation remained below 1°C.

Figure 4 illustrates the fluctuations of air temperature during the recording test. To ensure a clear and reliable validation of the model, four representative points are selected on the following criterion: the temperature remains in a $\pm 1°C$ range for at least one hour. The different variables measured and their mean values over these intervals are presented in Table 1.

To validate the data collected from the test facility, an energy balance was performed for both the evaporator, (13) and the global system (14). The energy balance in the condenser was not calculated due to the imprecision of the air measurements.

$$\dot{m}_{\text{Temper}}\, c_{p\text{Temper}}\, \Delta T_{\text{Temper_evap}} = \dot{m}_{\text{R507A}}\, \Delta h_{\text{R507A_evap}} \quad (13)$$

$$\dot{Q}_{\text{evap}} + \dot{Q}_{\text{cp}} = \dot{Q}_{\text{cond}} \quad (14)$$

Figure 5 shows the statistical variations in thermal capacity and the energy balance results for point 0. The data is presented using box plots, as this format allows for the display of key information in a single figure, such as the mean and distribution data. The black boxes represent the data distribution, with the blue horizontal lines inside indicating the mean and the red lines representing the median.

DOI 10.3384/ecp218881

Point	T_air (°C)	Q_R507A (kg/s)	HP (bar)	BP (bar)	T_evap,out (°C)	T_comp,out (°C)	T_cond,out (°C)
0	19.06	0.841	12.32	7.55	11.09	48.20	20.82
1	16.50	0.828	11.88	7.55	10.88	48.43	19.30
2	15.42	0.815	11.43	7.54	10.69	48.31	17.82
3	19.89	0.847	12.72	7.56	11.06	48.37	22.03

Table 1. Experimental results under different steady-state conditions

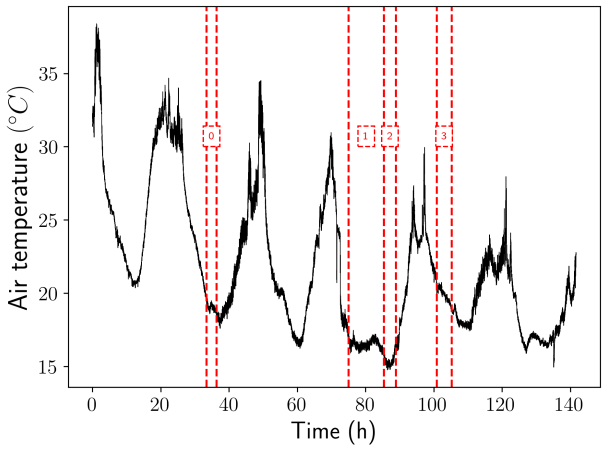


Figure 4. Evolution of the air temperature, measured at the air condenser. The four experimental periods corresponding to the four approximate steady conditions are indicated by the red dashed lines.

The ends of the boxes denote the first and third quartiles, while the black "whiskers" extend to the minimum and maximum values.

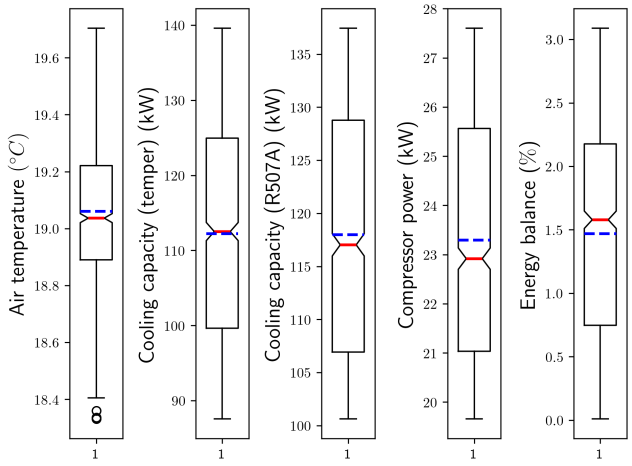


Figure 5. Box plot at point 0 comparing the thermal power exchanged at the evaporator and condenser. Energy balance of the system. Median: red bar, Mean: blue line.

The energy balance revealed a small discrepancy, less than 2.5 %, which validates the measurements taken in the test facility.

4.2 R507 model and calibration

4.2.1 Presentation of the R507 model

A `Modelica` model has been used to capture the behaviour of the chiller tested experimentally. As explained in 3.3, the aim is to model steady-like behaviors of the system. The assumption here made considers that the air temperature fluctuations are small enough to approximate the operations of points 0 to 3 to steady conditions. The chiller is modelled with four main components as shown in Figure 1 and presented in 2.1. Here, the different parameters to be calibrated are presented along with the setup used in `OpenModelica`.

Most of the modelling is based on an adapted version of the `Modelica Standard Libry (MSL)` except for the types of fluid used in the model. Indeed, the MSL does not integrate the possibility of utilising fluids like R507A and Temper-55. The two fluids can be introduced by using the open source library `CoolProp` (Bell et al. 2014), whose integration into the software `OpenModelica` has been done with the library `ExternalMedia` (Casella and Richter 2008).

Overall, twelve uncertainties and parameters can be calibrated in the model as shown in Table 3. They correspond to the uncertainties associated to the values measured experimentally and to the parameters of the components of the model (heat exchange coefficients, efficiency, overheating). The way they are calibrated is explained in the next section based on the methodology described in 2.2. Large uncertainties for the air temperature and mass flow rates were initially taken in the a priori distributions ($2°C$ and 1 kg/s resp.): the resulting calibrated point is consistent with this choice.

The prior distributions are chosen based on the knowledge of the uncertainties ($U_.$) and the parameters. For the uncertainties, the value is sampled into a normal distribution centred into 0 and with a standard deviation corresponding to the uncertainty of the probe (temperature or mass flow rate). This is added to the observed value experimentally as an input of the model. Regarding the rest of the parameters, their distribution is chosen uniform with bounds determined as much as possible from the experience of the user: the larger is the range, the more unphysical operating points for the model will be computed, leading to important errors between the model and the experiment or simply difficulties for the model to converge.

4.2.2 Calibration

Given a random sampling of all the uncertainties of measurements and the parameters to be calibrated in the model done N times according to prior distributions of probability, the principle of the ABC method is to compare the values obtained with the model to the experiment and then select the most relevant.

Min C_vRMSE	Min C_vRMSE	Min $NMBE$	Max $NMBE$
0	0.2	-0.2	0.2

Table 2. Range of acceptable errors for the selection of the parameters of the model.

The relevance is based on criteria defined by the user depending on the values observed and the tolerance accepted for the calibration in terms, for example, of the mean bias error (MBE) and the root mean square error (RMSE) with the experimental measurements. These two criteria are the ones retained for the selection of the points in the sample and can be written as follows for each value observed, as explained in 2.2.

The ranges for the selection of the parameters are indicated in Table 2. It means that, for one set of parameters from the sample, when comparing the values obtained with the model, the set is conserved if the RMSE and MBE are in the range indicated. The choice of 20% enables to keep sets of parameters with a reasonable error compared to the experiment.

The size of the design of experiments concerns 5000 simulations of approximately 30 seconds CPU, that have been carried out. With the RMSE and MBE criteria below 20%, only 6 of them have been selected. By making the criteria looser, for example with accepting results with a 30% error on the RMSE and MBE, 127 could have been retained. Here, the motivation was to show the principle of the method and the good possibilities for a calibration but this could be improved to get a larger sample for the posterior distributions. Also , out of the 5000 samples, some lead to unphysical behaviors and never converge in the model. These can be kept and analyzed to actually understand the potential problems of robustness of the model to the parameters. Only fifty out of the 5000 samples failed in this case, indicating quite a robust model, numerically.

The optimal point for the different parameters and uncertainties calibrated is shown in Table 3. These are determined with the resulting distributions built on the selected points of each parameter. The most probable value in each distribution obtained corresponds to the optimal value.

Figures 6 and 7 show the distributions of the isentropic effiency η_{is} and Q_v, prior and posterior to the approximate bayesian calibration. The distribution in green represents the distribution on which the parameters are drawn to simulate the behaviour of the chiller with the model and the orange one represents the points for which the selection criteria are fulfilled. The blue curve gives an approximation of the orange distribution based on a kernel smoothing

$U_{T_{air}}$	$U_{Q_{air}}$	$U_{T_{temper}}$	$U_{Q_{temper}}$	K_{lat}^{cond}	K_{sg}^{cond}
-1.61 °C	-0.78 kg/s	-0.14 °C	-0.32 kg/s	160.9	136
K_{sl}^{cond}	K_{lat}^{evap}	K_s^{evap}	$Q_v^{compressor}$	$\Delta T_{valve,nom}$	η_{is}
140.8	134.5	117.9	0.022 m^3/s	5.26 °C	0.398

Table 3. Table of the optimal values based on the calibration done with the R507 model.

Q_{R507}	p_{cond}	p_{evap}
Mass flow rate of R507	Pressure in the condenser	Pressure in the evaporator
$T_{out,evap}$	$T_{in,cond}$	$T_{out,cond}$
Temperature at the outlet of the evaporator	Temperature at the inlet of the condenser	Temperature at the outlet of the condenser
$P_{th,evap}$		
Thermal power exchanged at the evaporator		

Table 4. Description of the observed measurements used for the comparison with the model.

method (Baudin et al. 2017). A peak in the distribution can be noticed in both cases, meaning that the model is very sensitive to these two parameters for the calibration. Given a compression ratio, the temperature at the inlet of the condenser is strongly affected by the isentropic efficiency. Being too far from $\eta_{is} = 0.398$ therefore seems to lead to probable large errors on the calibration. The same can be inferred on Q_v, which controls the mass flow of the refrigerant. For three other parameters, the posterior distribution is quite different from the prior choice, which indicates a sensitivity of the model to these parameters to be consistent with the experimental observations (ΔT_{valve}, $U_{T_{air}}$ and K_s^{evap}).

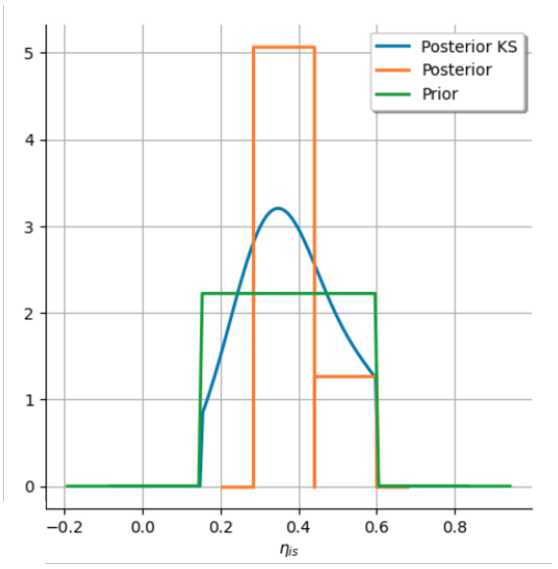


Figure 6. Distribution of the isentropic efficiency, prior and posterior to the application of the ABC method and the selection. The blue distribution approximates the distribution of orange values selected with a kernel smoothing.

It is now interesting to compare the predictions of the model with the optimal set of parameters determined with the ABC method to the actual observations measured experimentally as described in Table 4.

DOI 10.3384/ecp218881

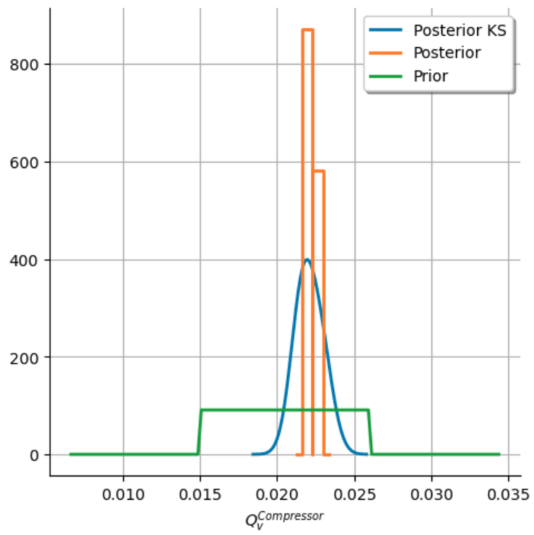


Figure 7. Distribution of the compressor volumetric flow, prior and posterior to the application of the ABC method and the selection. The blue distribution approximates the distribution of orange values selected with a kernel smoothing.

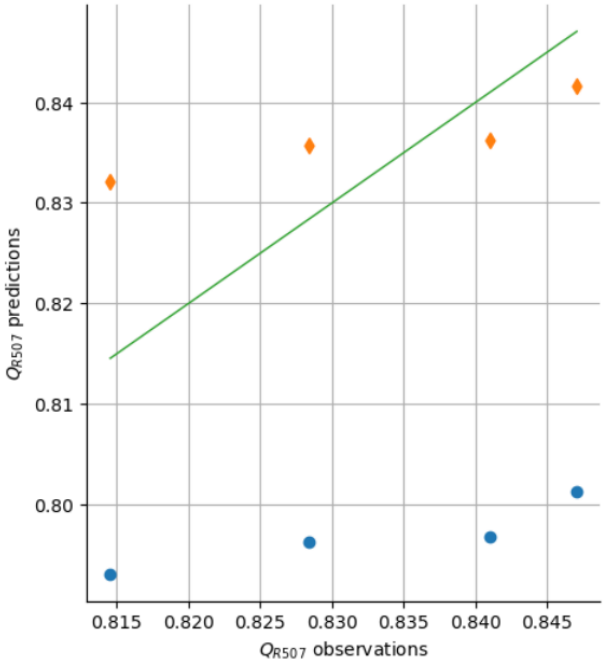


Figure 8. Distribution of the compressor volumetric flow, prior and posterior to the application of the ABC method and the selection. The blue distribution approximates the distribution of orange values selected with a kernel smoothing.

Figure 8 compares the predictions from the model with the experimental values measured on the mass flow rate of refrigerant. The blue dots show the initial predictions based on the most probable set of parameters with the prior distributions whilst the orange dots indicate the predictions from the posterior, built distributions from the selection. The green line is the first bisector corresponding to a perfect equality between predictions from the model and experiments for the different observations. Overall, the ABC method leads to a good calibration of the model, improving clearly the match between experiment and modelled results.

Another view on the comparison between the experiment and the model is shown in Figure 9. It compares the cycles for point 0 (cf. Table 1). What can be observed is the overestimate of the pressure at the condenser for four sets of parameters. The fifth set corresponds to the optimal point and matches quite well with the experimental cycle. Other differences concern the enthalpy at the outlet of the condenser (slight difference) and the specific enthalpy at the compressor outlet.

The advantages of the ABC calibration method over more traditional approaches are clearly demonstrated in this case. Due to the complexity of the model, certain parameter evaluations may lead to simulation failures—situations that ABC is uniquely equipped to handle, as it does not require explicit likelihood evaluations. As illustrated in Figure 6 and Figure 7, the resulting posterior distributions are approximately Gaussian. ABC enables the generation of a discrete sample from this posterior, providing a flexible and robust inference framework. Given the typically nonlinear nature of the models, conventional methods such as nonlinear least squares or nonlinear Gaussian calibration often suffer from poor exploration of the parameter space and high sensitivity to initial conditions. In contrast, ABC facilitates a more comprehensive exploration, especially when the likelihood is intractable or the model is prone to numerical instabilities. Moreover, due to limited prior knowledge about the model parameters, a uniform prior distribution is often the most

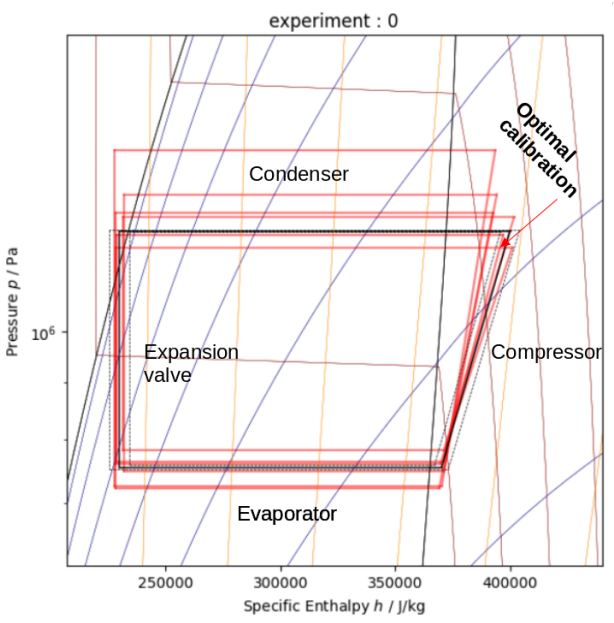


Figure 9. Representation of the experimental cycle for point 0 (black lines) and the ones obtained with the model after calibration with the ABC method. The point resulting from the optimum values in the calibration is shown by the arrow.

appropriate choice. Gaussian priors, by contrast, may introduce unwarranted assumptions and overly constrain the calibration process.

Generally, the calibration enables a quite satisfying fit of the model with the experiment for the four observations on the different values. Although a larger shift can be observed for the inlet temperature at the condenser (or outlet temperature of the compressor), the overall prediction is improved with the calibration. That shift could be explained by the choice made for modelling the compressor with constant efficiencies: a variation of the volumetric and isentropic efficiencies with the pressure ratio may improve these results (Gicquel 2001).

5 Conclusions and further work

This study was motivated by the use of an approximate calibration method for the modelling of a chiller. To the knowledge of the authors, this kind of methods for this type of applications has rarely been done before. Experimental results on a chiller test facility at EDF have been obtained and used to make a calibration of a model representing the chiller in *Modelica*.

With four experimental points approximated as being in a steady state, a calibration has been performed based on the approximate bayesian calibration method. Considering seven variables observed experimentally for these four points, a set of parameters is varied in the model following a design of experiments constructed on probabilistic distributions a priori. The calibration done has consisted in selecting the sets of parameters ensuring an acceptable error between the experiments and the variables modelled.

The results obtained with the calibration are satisfying as they enable to maintain the error below 20% for the sets of parameters kept. Given the initial number of parameters in the model, the ABC method also offers a very interesting way of priorizing the parameters regarding the sensitivity of the variables of interest in the model. In our case, consistently with what is physically modelled, the isentropic efficiency, the volumetric flow rate and the nominal overheating for the expansion valve are the most sensitive parameters.

It is interesting to comment on the use of such methods to analyze further experimental results. In this study, the choice was to start with approximating steady regimes for the chiller with the experimental results based on an assumption of low variations of the outdoor air temperature. The analysis here presented shows that it is possible to calibrate the model to these experimental conditions but it also shows that only a few points are kept in the end, out of the 5000 samples. Different improvements to the current application of the method can be listed here for further work:

- Treating the problem as an unsteady one by considering the variations of the variables over time. In practice, it means that for each time step in the sample considered, the variables observed correspond to an experimental point and the same method can be used with more experimental points. This would imply an improvement of the model to account for the regulation loops in the experiment but also an increase in the CPU cost, with more observations in the method.

- Improving the components of the model: the compressor efficiencies depend on the pressure ratio, which is not the case in the current model. Also, the air-cooled condenser is modelled with a counterflow heat exchanger whilst in reality, it is a cross-flow exchanger.

- With more points selected for the probabilistic distributions, it would then be interesting to transpose the posterior distributions to more extreme air temperature conditions, beyond $40°C$, to make an evaluation of the uncertainties, in particular on the cooling power of the chiller.

- The choice of the prior distributions is essential to avoid non-physical and unstable numerical points in the model. A careful investigation of that choice can significantly help the process of calibration.

Acknowledgements

This work has been supported by Electricité de France (EDF) and the collaboration of three departments: MFEE (Fluid Mechanics, Energy and Environment), TREE (Technologies and Research for Energy Efficiency) and PRISME (Performance, Industrial Risk and Monitoring for Maintenance and Operation).

References

Afzali, Sayyed Faridoddin and Vladimir Mahalec (2017). "Optimal design, operation and analytical criteria for determining optimal operating modes of a CCHP with fired HRSG, boiler, electric chiller and absorption chiller". In: *Energy* 139, pp. 1052–1065. ISSN: 0360-5442. DOI: https://doi.org/10.1016/j.energy.2017.08.029. URL: https://www.sciencedirect.com/science/article/pii/S0360544217314056.

Baudin, Michaël et al. (2017). "Openturns: An industrial software for uncertainty quantification in simulation". In: *Handbook of uncertainty quantification*. Springer, pp. 2001–2038.

Bell, I. et al. (2014). "Pure and Pseudo-pure Fluid Thermophysical Property Evaluation and the Open-Source Thermophysical Property Library CoolProp". In: *Industrial Engineering Chemistry Research*. URL: https://pubs.acs.org/doi/full/10.1021/ie4033999.

Cant, Kevin and Ralph Evins (2023). "Improved calibration of building models using approximate Bayesian calibration and neural networks". In: *Journal of Building Performance Simulation* 16.3. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/19401493.2022.2137236, pp. 291–307. DOI: 10.1080/19401493.2022.2137236. URL: https://doi.org/10.1080/19401493.2022.2137236.

Casella, F. and C. Richter (2008). "ExternalMedia: a Library for Easy Re-use of External Fluid Property code in Modelica". In: *The Modelica Association*.

Cecchinato, Luca, Manuel Chiarello, and Marco Corradi (2010). "A simplified method to evaluate the seasonal energy performance of water chillers". In: *International Journal of Thermal Sciences* 49.9, pp. 1776–1786. ISSN: 1290-0729. DOI: https://doi.org/10.1016/j.ijthermalsci.2010.04.010. URL: https://www.sciencedirect.com/science/article/pii/S1290072910001092.

CoolProp (n.d.). "R507A defined in Coolprop". In: (). URL: https://coolprop.org/fluid_properties/fluids/R507A.html.

Gicquel, Renaud (2001). *Compresseurs volumétriques*. Vol. Systèmes énergétiques, tome 1. Mines de Paris.

Lissandrin, Marco et al. (2017). "Optimal operational efficiency of chillers using oil-free centrifugal compressors". In: *International Journal of Refrigeration* 82, pp. 83–96. ISSN: 0140-7007. DOI: https://doi.org/10.1016/j.ijrefrig.2017.05.019. URL: https://www.sciencedirect.com/science/article/pii/S0140700717302104.

Marin, Jean-Michel, Christian P Robert, et al. (2007). *Bayesian core: a practical approach to computational Bayesian statistics*. Vol. 268. Springer.

Mouneer, Tarek A., Mohamed H. Aly, and Ehab M. Mina (2021). "Optimal design configuration and operating sequencing of hybrid chiller plant: A case study of hotel building in Cairo, Egypt". In: *Journal of Building Engineering* 42, p. 102796. ISSN: 2352-7102. DOI: https://doi.org/10.1016/j.jobe.2021.102796. URL: https://www.sciencedirect.com/science/article/pii/S2352710221006549.

Taylor, Barry N and Chris E Kuyatt (1994). "NIST Technical Note 1297 Guidelines for Evaluating and Expressing the Uncertainty of NIST Measurement Results". In: *Technology*.

Turner, Brandon M. and Trisha Van Zandt (2012). "A tutorial on approximate Bayesian computation". In: *Journal of Mathematical Psychology* 56.2, pp. 69–85. ISSN: 0022-2496. DOI: https://doi.org/10.1016/j.jmp.2012.02.005. URL: https://www.sciencedirect.com/science/article/pii/S0022249612000272.

Wetter, Michael et al. (2014). "Modelica Buildings library". In: *Journal of Building Performance Simulation* 7.4, pp. 253–270. DOI: 10.1080/19401493.2013.765506. URL: https://doi.org/10.1080/19401493.2013.765506.

Zhen, Cheng, Jide Niu, and Zhe Tian (2023). "Research on Model Calibration Method of Chiller Plants Based on Error Reverse Correction with Limited Data". In: *Energies* 16.2. ISSN: 1996-1073. DOI: 10.3390/en16020918. URL: https://www.mdpi.com/1996-1073/16/2/918.

DOI
10.3384/ecp218881

A Dynamic Simulation Model of Outdoor Swimming Pool with Thermal Energy Storage, Boiler and Solar Thermal Collectors

Pierre Clément Blaud[1] Imad Mourtaji[1]

[1]Akkodis Research, Akkodis, France, {firstname.surname}@akkodis.com

Abstract

Outdoor swimming pool facilities are significant energy consumers, primarily due to the need to maintain thermal comfort for swimmers, as the pool is exposed to wind and outdoor air. To address the challenge of reducing energy consumption, effective simulation of the dynamical system is essential to identify energy-saving solutions. This paper presents a comprehensive implementation of an outdoor swimming pool simulation model using the Modelica programming language. The simulation model also incorporates a thermal storage system, a boiler, and a solar thermal collector, allowing a detailed analysis of the system. In addition, an in-depth explanation of the physical equations governing the swimming pool is provided, along with the methodology used to construct the Modelica model. The results show that the model allows the simulation of the thermal behavior of an outdoor swimming pool, including various heat losses, solar heat gains, and heat production from the boiler and solar collectors.

Keywords: Swimming pool model, energy system, thermal storage, boiler, solar thermal collectors

1 Introduction

In late 2017, France had 4,135 public swimming pool facilities comprising 6,412 pools and covering a total area of 1.6 million square meters (Cour des comptes 2018). These facilities represent a significant financial burden for local communities, both due to the need to offer affordable access to the public and the high levels of energy consumption required for operation (Delgado Marín, Vera García, and García Cascales 2019). A large proportion of these facilities are aging, with more than 70% built before 1995 (Cour des comptes 2018). The recent energy crisis has further exacerbated the situation, leading to increased operational costs and in some cases, full or partial closures during the winter months (Cour des comptes 2018; Delgado Marín, Vera García, and García Cascales 2019). The high energy consumption associated with these facilities arises from the need to heat large volumes of water to maintaining temperatures between 25°C and 28°C as recommended by the World Aquatics Federation, as well as to power auxiliary systems such as showers, ventilation, lighting, and administrative areas. Modeling the dynamic behavior of public swimming pools and their associated energy systems is essential for analyzing energy use and developing advanced control strategies for heating systems (Li et al. 2021).

Numerous studies have focused on the modeling of indoor swimming pools (Yuan et al. 2021; Kühn et al. 2022), outdoor swimming pools (Buscemi et al. 2024; Bernhard et al. 2019), and the integration of swimming pools with heat production systems (Li et al. 2021). Several of these works use multi-domain modeling languages such as TRNSYS (Delgado Marín, Vera García, and García Cascales 2019; Starke et al. 2017; Liu, Kim, and Shin 2021) or Modelica (Kühn et al. 2022). In the study (Delgado Marín, Vera García, and García Cascales 2019), TRNSYS was used as a simulation platform for implementing predictive control strategies. Their results demonstrated that predictive control significantly reduces energy consumption compared to traditional proportional-integral-derivative (PID) controllers. In the study (Starke et al. 2017) conducted a technical and economic analysis of a pool equipped with solar thermal heating and a heat pump, concluding that this configuration improves energy efficiency. Similarly in (Liu, Kim, and Shin 2021), the authors used TRNSYS to model a swimming pool and adjacent buildings, examining the feasibility of using a fuel cell cogeneration unit for heat and power generation. Although the system was found to be economically viable, the payback period exceeded 10 years. It can be observed that detailed modeling of swimming pool systems with TRNSYS simulation tool is well established in the literature. However, concerning the Modelica modeling language, few studies have addressed swimming pool systems coupled with heat sources, to our knowledge, only one of these studies focused on an indoor swimming pool (Kühn et al. 2022), without considering an outdoor pool and several heat sources.

In the present work, we model a public outdoor swimming pool using the Modelica language including an external pool, consequences of weather conditions on the pool, a thermal energy storage unit and two heat sources: a boiler and a solar thermal collector. The main contributions of this work are:

- The implementation and simulation of a public outdoor swimming pool with consideration of external environment and thermal cover modeling using the Modelica language;
- The development of a complete energy system sim-

ulation that includes thermal energy storage (TES), and two heat sources, with a boiler, and a solar thermal collector (STC);

- And finally to propose a realistic comparative analysis of heat losses, and energy savings with and without the use of solar thermal collectors.

This paper is organized as follows: Section 2 introduces the simulation models; Section 3 depicts experimental setup; Section 4 presents the results of the work; and Section 5 concludes the work.

2 Simulation Model

2.1 Swimming Pool Model

The complete model of the swimming pool is shown in Figure 1. In the following sections, each submodel will be presented in detail.

2.1.1 Energy balance

Figure 2 depicts the energy balance without thermal cover. Assuming that the pool water is fully mixed, the temperature dynamics are governed by the following energy balance equation (Gauthier-Clerc et al. 2023; Buscemi et al. 2024) :

$$V_p \rho_w c_{p,w} \frac{dT_p}{dt} = \dot{Q}_{tot} \tag{1}$$

$$\dot{Q}_{tot} = \dot{Q}_{sun} + \dot{Q}_{heat} - \dot{Q}_{conv} - \dot{Q}_{evap} - \dot{Q}_{rad} - \dot{Q}_{cond} - \dot{Q}_{re} \tag{2}$$

with V_p the volume of the pool, ρ_w the density of water, $c_{p,w}$ the specific heat of water, $\dot{Q}_{sun}$ the heat gain from solar irradiance, $\dot{Q}_{heat}$ the heat power from the heating, $\dot{Q}_{conv}$ the heat loss from convection, $\dot{Q}_{evap}$ the heat loss from the evaporation, $\dot{Q}_{rad}$ the heat loss from the radiation, $\dot{Q}_{cond}$ the heat loss from conduction and $\dot{Q}_{re}$ the heat loss from water refilling and recycling. For the rest of the work, conduction losses are neglected as it contributes to less than 1% of total energy losses (Buscemi et al. 2024).

During the night, it is preferable to cover the pool to reduce heat loss, in this case the energy balance is depicted in Figure 3. The temperature dynamic is equal to (Buscemi et al. 2024):

$$V_p \rho_w c_{p,w} \frac{dT_p}{dt} = \dot{Q}_{heat} + \dot{Q}_{cover} - (\dot{Q}_{cond} + \dot{Q}_{re}) \tag{3}$$

$$\dot{Q}_c \simeq \dot{Q}_{sun} - (\dot{Q}_{conv} + \dot{Q}_{rad}) \tag{4}$$

with V_p the volume of the pool, ρ_w the density of water, $c_{p,w}$ the specific heat of water, T_p the pool water temperature, $\dot{Q}_{sun}$ the heat power from the solar irradiance, $\dot{Q}_{heat}$ the heat power from the heating, $\dot{Q}_{conv}$ the heat loss from convection, $\dot{Q}_{evap}$ the heat loss from the evaporation, $\dot{Q}_{rad}$ the heat loss from the radiation, $\dot{Q}_{cond}$ the heat loss from conduction, $\dot{Q}_{re}$ the heat loss from water refilling and recycling, and with $\dot{Q}_c$ the heat exchange from the pool to the cover.

2.1.2 Solar gain

Since solar irradiance can heat the pool water, the governing equation is equal to (Buscemi et al. 2024):

$$\dot{Q}_{sun} = G_h \alpha_p A_p \tag{5}$$

with G_h the global horizontal solar irradiance, α_p the solar absorptance of the pool and A_p the pool area. The solar absorptance of the pool depends on whether the pool is covered or not by the thermal cover. When the pool is not covered, the absorptance of the pool is equivalent to the absorptance of water (Buscemi et al. 2024). The Modelica graphical implementation is provided in Figure 4. An ideal prescribed heating power is implemented, and a switch controls the flow of covered or uncovered water. Both pool area and solar absorptance gains are incorporated.

2.1.3 Evaporation losses

Evaporation produces a transfer of mass and heat and occurs on the surface of a saturated liquid. This process is enhanced by both natural and forced convection, the latter being induced by wind in the case of outdoor pools (Buscemi et al. 2024). The heat losses due to evaporation are described as (Buscemi et al. 2024):

$$\dot{Q}_{evap} = \dot{E}_p \Delta H_w A_p \tag{6}$$

with $\dot{Q}_{evap}$ the heat loss from the evaporation, $\dot{E}_p$ the rate of evaporation of water, ΔH_w the latent heat of vaporization of water, A_p the pool area. When the thermal cover is unfolded, the evaporation does not occur, $\dot{Q}_{evap} = 0$. The rate of evaporation of water with forced convection and natural convection is equal to (Buscemi et al. 2024):

$$\dot{E}_{p,fc} = (0.28 + 0.784 v_{wi}) \frac{(p_{v,i} - p_{v,e})^{0.695}}{\Delta H_w} \tag{7}$$

$$\dot{E}_{p,nc} = h_m (\rho_{v,i} - \rho_{v,e}) \tag{8}$$

with $\dot{E}_{p,fc}$ the forced convection rate of evaporation, v_{wi} the wind velocity, $p_{v,i}$ the vapor pressure at the water surface depends to the saturated vapor pressure at pool temperature, $p_{v,i} = p_{vs}(T_p)$, $p_{v,e}$ the vapor pressure of the free stream depends to the saturated vapor pressure at air temperature $p_{v,e} = p_{vs}(T_a)(\frac{RH_e}{100})$, ΔH_w the latent heat of vaporization of water, $\dot{E}_{p,nc}$ the natural convection rate of evaporation, h_m the transfer coefficient for natural convection, $\rho_{v,i}$ the vapor density at the water surface and $\rho_{v,e}$ the free stream vapor density. The total rate of evaporation of the pool with the natural and forced convection is (Buscemi et al. 2024):

$$\dot{E}_p = F_A \left(\dot{E}_{p,nc}^{7/2} + \dot{E}_{p,fc}^{7/2} \right)^{2/7} \tag{9}$$

with $\dot{E}_p$ the pool rate of evaporation, $\dot{E}_{p,nc}$ the natural convection rate of evaporation, $\dot{E}_{p,fc}$ the forced convection rate of evaporation and F_A the evaporation surface

DOI 10.3384/ecp218891

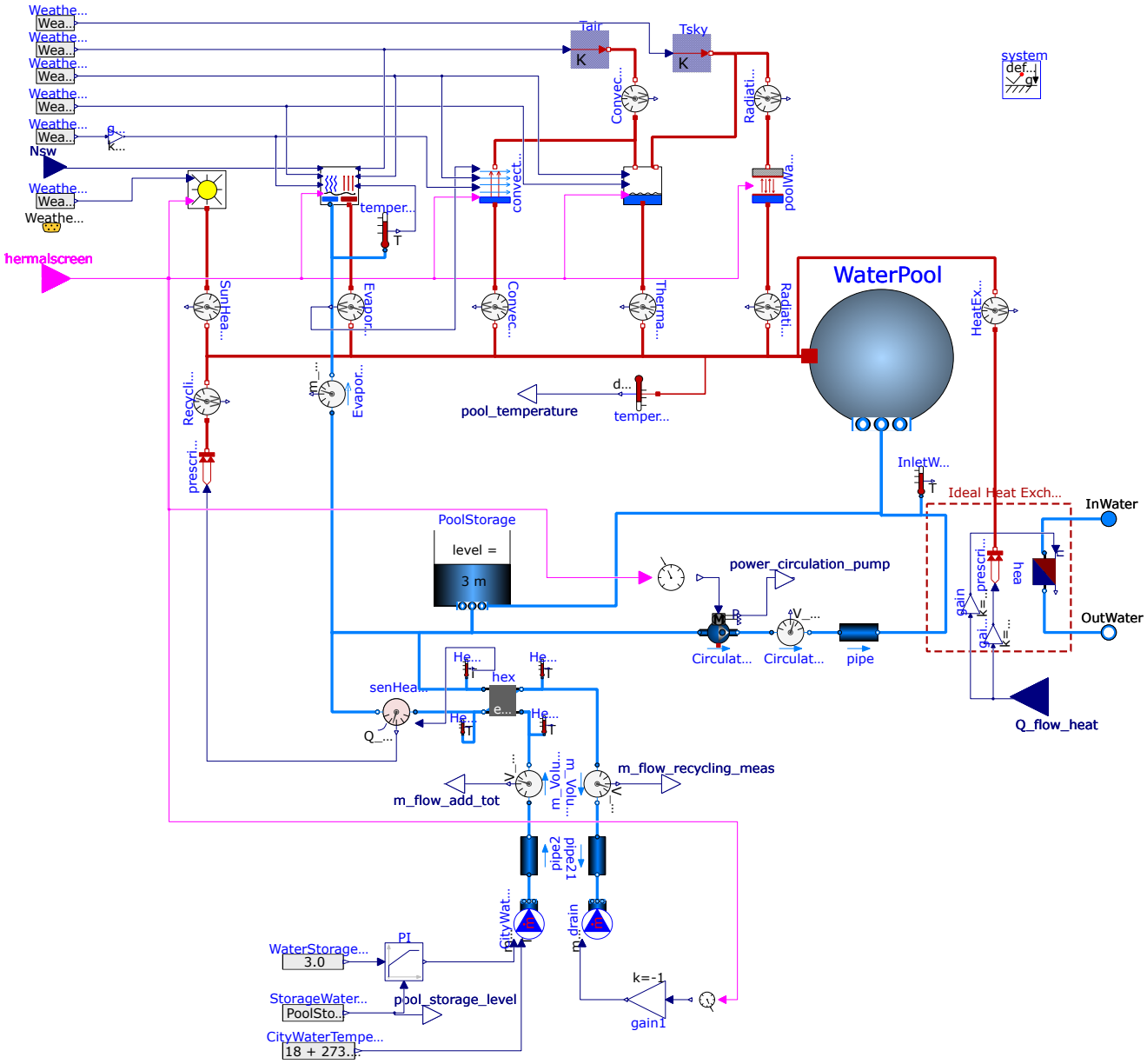


Figure 1. Model of the outdoor swimming pool.

enhancement which depends on the number of swimmer. The evaporation surface enhancement factor F_A is equal to (Buscemi et al. 2024):

$$F_A(F_u) = 1 + 2.3F_u \text{ if } F_u < 0.1 \quad (10)$$

$$F_A(F_u) = 1.2 + 0.3F_u \text{ if } 0.1 \leq F_u \leq 1 \quad (11)$$

$$F_A(F_u) = 1.5 \text{ if } F_u > 1 \quad (12)$$

with F_u the utilization factor of the outdoor swimming pool. The utilization factor is equal to (Buscemi et al. 2024):

$$F_u = \frac{A_{max}}{A_p} N_{sw} \quad (13)$$

with A_{max} the pool area per swimming pool when maximum occupancy, A_p the pool area and N_{sw} the number of swimmers inside the pool.

To calculate the transfer of mass and heat from the evaporation losses within the program, two controlled sources are implemented, one for the transfer of water and the second for heat transfer Figure 5. In addition, a program is written to control the controllable sources Listing 1. A numerical difficulty arises from the calculation of density and pressure differences, Equation 7 is no longer valid. A conditional evaluation was added and set to $\dot{E}_{p,nc}$ or $\dot{E}_{p,fc}$ when both differences become negative.

Listing 1. Sub-model program view of the modeling of the evaporation loss (heat and water loss).

```
...
equation
  Tf = (WaterTemperature + Tair)/2;
```

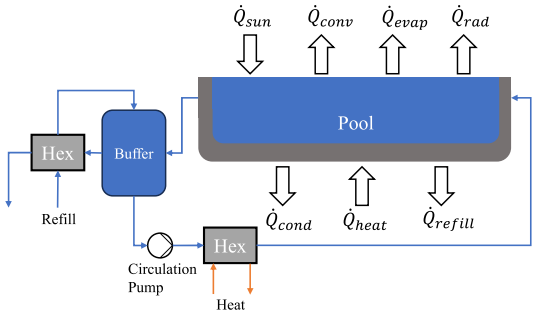


Figure 2. Energy balance of pool without thermal cover.

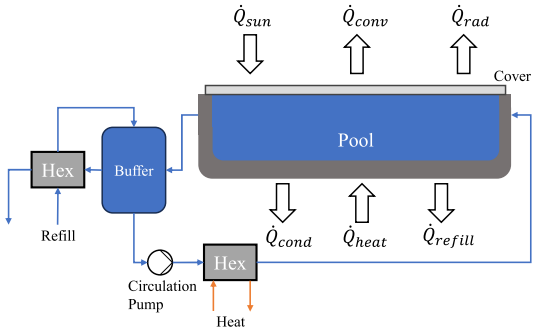


Figure 3. Energy balance of pool with thermal cover.

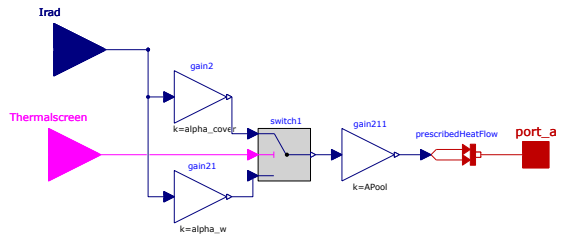


Figure 4. Sub-model of the modeling of the solar gain for the outdoor swimming pool.

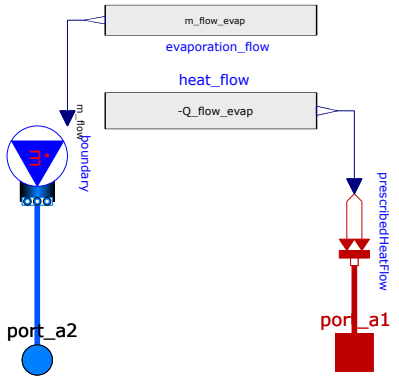


Figure 5. Sub-model graphical view of the modeling of the evaporation loss (for simplification only the equation part is presented).

```
  pvi = psat_TPool;
  pve = psat_TAir*(RHair/100);

  if ThermalScreen == true then
    Eunfc = 0.0;
    Eunnc = 0.0;
    Eun = 0.0;
    hm = 0.0;
    Ral = 0.0;
    Sh = 0.0;
    Fu = 0.0;
    Fa = 0.0;
    Dwaf = 0.0;
    m_flow_evap = 0.0;
    Q_flow_evap = 0.0;
    delta_rho=0.0;
    delta_p=0.0;
  else
    Fu = (Amax/Ap)*Nsw;
    if Fu < 0.1 then
      Fa = 1 + 2.3*Fu;
    elseif (0.1 <= Fu) and (Fu < 1) then
      Fa = 1.2 + 0.3*Fu;
    else
      Fa = 1.5;
    end if;
    Dwaf = (1.049*10^(-4)/pAtm)*Tf^1.774;
    Ral = (g_n*abs(rho_vi - rho_ve)*Lch^3)
        /(mudaf*Dwaf);
    if Ral < 10^7 then
      Sh = 0.54*Ral^(1/4);
    else
      Sh = 0.15*Ral^(1/3);
    end if;
    hm = Sh*Dwaf/Lch;

    delta_rho = (rho_vi - rho_ve);
    delta_p = (pvi - pve);
    if (delta_rho > 1e-4) and (delta_p > 1)
        then
      Eunnc = hm*(rho_vi - rho_ve);
      Eunfc = (0.28 + 0.784*vwind)*(((pvi
          - pve)^(0.695))/ hVap);
      Eun = (Eunfc^(7/2)+Eunnc^(7/2))
          ^(2/7);
    elseif (delta_rho <= 1e-4) and (delta_p
        > 1) then
      Eunnc = 0.0;
      Eunfc = (0.28 + 0.784*vwind)*(((pvi
          - pve)^(0.695))/ hVap);
      Eun = Eunfc;
    elseif (delta_rho > 1e-4) and (delta_p
        <= 1) then
      Eunnc = hm*(rho_vi - rho_ve);
      Eunfc = 0.0;
      Eun = Eunnc;
    else
      Eunnc = 0.0;
      Eunfc = 0.0;
      Eun = 0.0;
    end if;

    m_flow_evap = -(Fa*Eun*Ap);
    Q_flow_evap = Fa*Eun*Ap*hVap;
  end if;
...
```


DOI 10.3384/ecp218891

2.1.4 Convective losses

Convective losses are computed based on the Bowen approach (Buscemi et al. 2024):

$$\dot{Q}_{conv} = R_{Bo}\dot{Q}_{eva} \tag{14}$$

with $\dot{Q}_{conv}$ the heat loss from convection, R_{Bo} the Bowen ratio, and $\dot{Q}_{eva}$ the heat loss from evaporation. The Bowen ratio is equal to (Buscemi et al. 2024):

$$R_{Bo} = \left(\frac{c_{pa,e}p_{a,tot}}{0.622\Delta H_w}\right)\left(\frac{T_p - T_a}{p_{v,i} - p_{v,e}}\right) \tag{15}$$

with R_{Bo} the Bowen ratio, $c_{pa,e}$ the specific heat of air, p_a the air pressure, ΔH_w the latent heat of vaporization of water, T_p the pool water temperature, T_a the air temperature, $p_{v,i}$ the vapor pressure at the water surface and $p_{v,e}$ the vapor pressure of the free stream.

The implementation of convective losses is visible in the list 2. It includes an if-else condition regarding the thermal cover unfolded according to Equation 14 and Equation 15. In addition, there is a numerical issue regarding the Bowen ratio implementation, if vapor pressure at the water surface is equivalent to vapor pressure of the free stream, Equation 15 is no longer valid (division by zero). Especially since, according to (Woolley, Harrington, and Modera 2011), the Bowen ratio could have negative values and convection contributes to heat water. When the Bowen ratio approaches zero, the convective loss is zero (Woolley, Harrington, and Modera 2011), as such, the numerical implements a conditional evaluation with hysteresis with 500 Pascal and 600 Pascal for low and high threshold respectively, and is set to zero.

Listing 2. Sub-model program view of the modeling of the convective losses.

```
...
equation
  Tair = fluid.T;
  Twater = solid.T;
  pvi = psat_TPool;
  pve = psat_TAir*(RH/100);

  if thermalScreen == false then
    Q_flow = Rbo*Qevap;
    delta_p = (pvi - pve);
    if hysteresis.y == true then
      Rbo = (cp_TAir*pAtm/(0.622*hVap))*((
          solid.T - fluid.T)/(pvi - pve));
    else
      Rbo = 0.0;
    end if;
  else
    Q_flow = 0;
    Rbo = 0;
    delta_p = 0;
  end if;
  solid.Q_flow = Q_flow;
  fluid.Q_flow = -Q_flow;
...
```

2.1.5 Radiative losses

The Stefan-Boltzmann equation is used in this work to calculate the radiative heat flow losses between the water surface and the environment (Buscemi et al. 2024):

$$\dot{Q}_{rad} = \sigma\varepsilon_p A_p(T_p^4 - T_{sky}^4) \tag{16}$$

with σ the Stefan-Boltzmann constant, ε_p the long wave emissivity of the pool, A_p the pool surface area, T_p the pool water temperature and T_{sky} the sky temperature. The sky view factor is not considered as it is considered equal to one (Buscemi et al. 2024).

The implementation of the radiative heat flow is visible in Listing 3. It includes an if-else rule regarding the thermal cover unfols.

Listing 3. Sub-model program view of the modeling of the radiative losses.

```
...
equation
  if thermalScreen == false then
    Q_flow = epsilon_p*APool*Modelica.
        Constants.sigma*(port_a.T^4 -
        port_b.T^4);
  else
    Q_flow = 0.0;
  end if;
...
```

2.1.6 Thermal cover losses

When the thermal cover is unfolded, it is in direct contact with the pool water, the heat from the pool is released by conduction on the cover material then by convection and by radiation from the cover to the outside environment. The conduction heat flow of the thermal cover is equal to (Buscemi et al. 2024):

$$\dot{Q}_c = \frac{\lambda_c}{s_c}(T_c - T_p)A_p \tag{17}$$

with $\dot{Q}_c$ the heat flow of the thermal cover, λ_c the thermal conductivity of the cover, s_c the thickness of the cover, T_c the cover temperature and A_p the pool surface area. The convective heat loss of the thermal cover is defined as (Buscemi et al. 2024):

$$\dot{Q}_{conv} = h_{conv}A_p(T_c - T_a) \tag{18}$$

with $\dot{Q}_{conv}$ the convective heat flow of the thermal cover, h_{conv} the variable convective coefficient, T_c the cover temperature and T_a the air temperature. The radiative heat flow losses between the cover surface and the environment is equal to (Buscemi et al. 2024):

$$\dot{Q}_{rad} = \sigma\varepsilon_c A_p(T_c^4 - T_{sky}^4) \tag{19}$$

with σ the Stefan-Boltzmann equation, ε_c the long wave emissivity of the cover, A_p the pool surface area, T_c the cover temperature and T_{sky} the sky temperature.

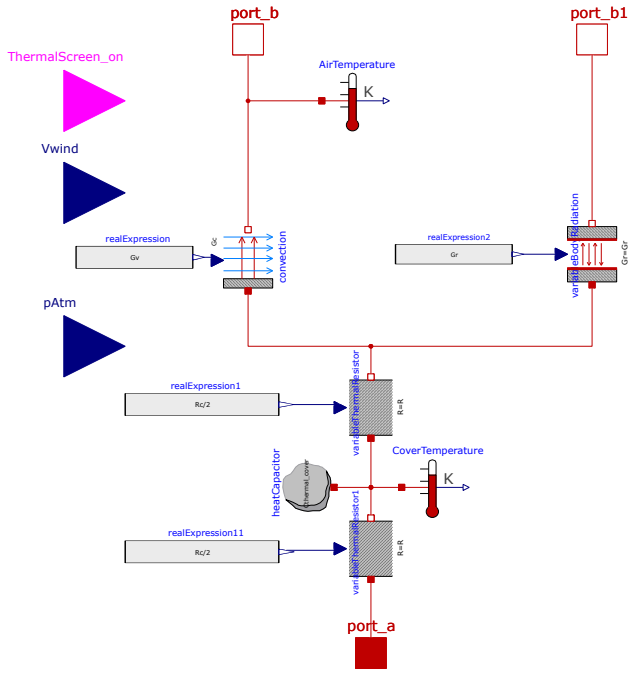


Figure 6. Modeling of the heat flow of the thermal cover.

Thermal cover model model consists of a graphical component, as shown in Figure 6, along with a text-based program designed to calculate the varying thermal coefficients, as detailed in Listing 4.

Listing 4. Sub-model program view of the modeling of the thermal cover heat flow.

```
...
equation
  Tf = (Tthermal_screen + Tair)/2;
  if ThermalScreen_on == true then
    Rc = Rts;
    Gr = epsilon_c*APool;
    Ral = (g_n*abs(Tthermal_screen - Tair)
        *(Lch^3)*Pr)/(Tf*eta_air_Tf^2);
    if Tthermal_screen > Tair then
      if Ral < 10^7 then
        Nul = 0.54*Ral^(1/4);
      else
        Nul = 0.15*Ral^(1/3);
      end if;
    else
      Nul = 0.52*Ral^(1/5);
    end if;
    hnc = ((lambda_air_Tf*Nul)/Lch);
    hfc = (5.7*Vwind^(0.8)*Lch^(-0.2));
    Gv = (hnc^(7/2) + hfc^(7/2))^(2/7) *
        APool;
  else
    Ral = 0.0;
    Nul = 0.0;
    hnc = 0.0;
    hfc = 0.0;
    Gr = 0.0;
    Rc = Rts;
    Gv = 0.0;
  end if;
...
```

2.1.7 Heating power

The heat source Q_{heat} for outdoor swimming pools incorporates a water-to-water heat exchanger (Buscemi et al. 2024). An ideal heat exchanger is utilized within the Modelica model, featuring an external circuit connected to the swimming pool to draw energy from the network. This system is coupled with an ideal heat source to supply heat to the water pool model, as illustrated in Figure 1.

2.1.8 Water buffer and recycling

The pool includes a water buffer volume designed to store excess water when swimmers are present, as well as to facilitate the addition of fresh water (Gauthier-Clerc et al. 2023). This fresh water is introduced through a heat exchanger that exchanges heat with gray water. Within Modelica implementation, a controlled water source removes gray water, while a second water source, controlled by a PI controller, maintains a constant water level in the buffer volume. When water volume decreases due to evaporation or removal, the controlled water source adds fresh water Figure 1.

2.2 Energy Systems Model

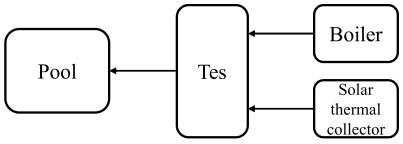


Figure 7. Block diagram of the considered energy system, constituted of pool, TES, boiler and STC.

The block diagram in Figure 7 illustrates the energy system used in the simulation. The swimming pool is modeled as a heat-consuming unit, while TES unit can either store or release heat to the pool. Two heat sources are integrated into the system: a boiler and an STC. The heat generated by the boiler can be directly controlled, whereas the heat produced by the STC depends on solar radiation and cannot be controlled actively. All three components originate from the AixLib Modelica library (Maier et al. 2024), and they are interconnected via pumps and fluid interfaces, as shown in Figure 8. The pool temperature is regulated using a proportional integral controller, while the boiler is managed by a hysteresis control system: the boiler operates at full capacity when the TES temperature falls below a defined minimum, and it turns off when the temperature reaches a specified upper limit. The STC is not actively controlled; instead, its heat output is directly delivered to the TES.

3 Experimental Setup

The swimming pool and associated energy systems were modeled using OpenModelica (Fritzson 2011) version

DOI 10.3384/ecp218891

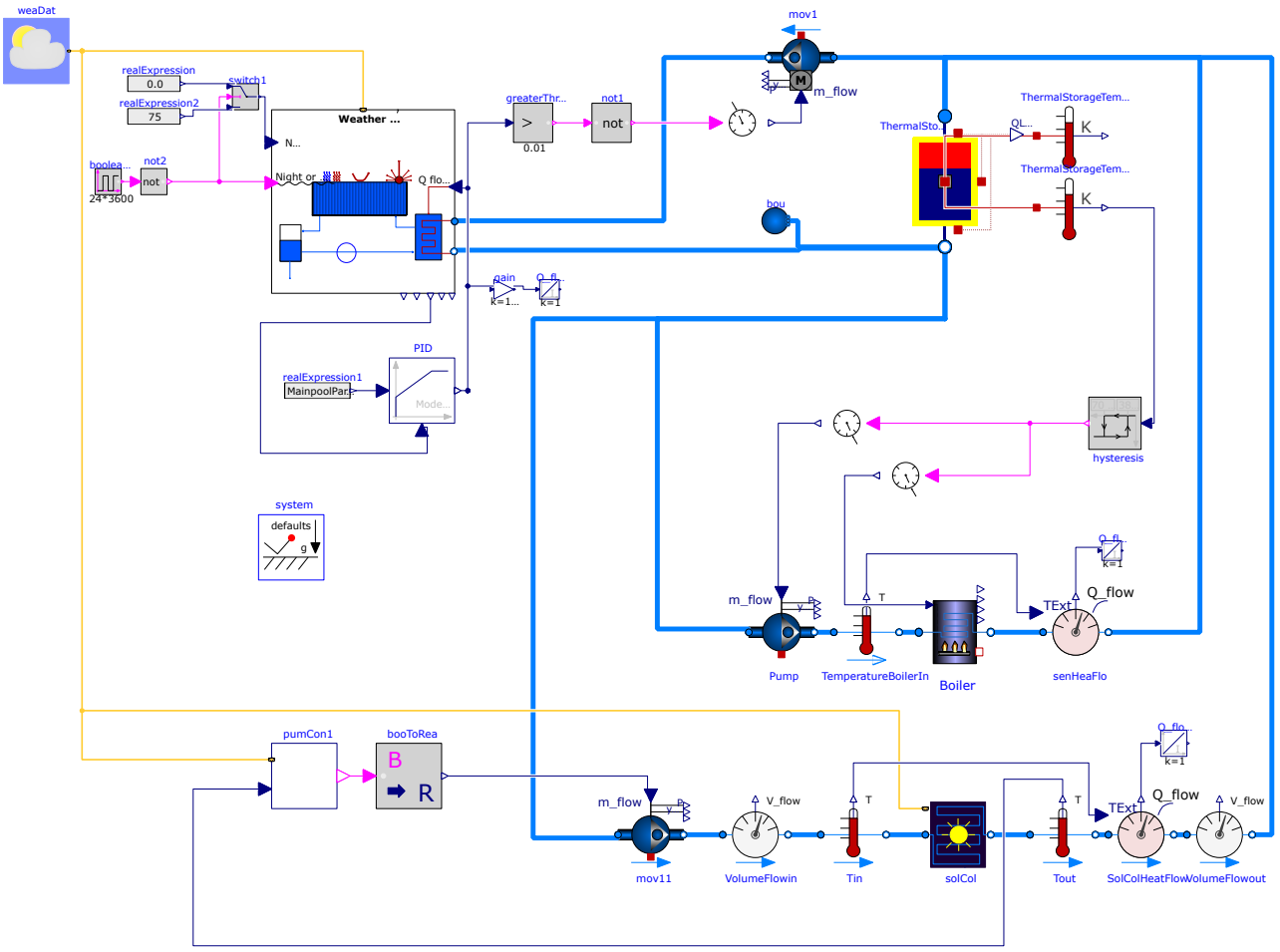


Figure 8. Model of the modeling of the energy system considering the swimming pool, the TES, the boiler and the STC.

1.23.1. Simulations were executed using the Dassl solver (Petzold 1982) on a laptop running Windows, equipped with an AMD Ryzen 7 PRO 7730U CPU and 32 GB of RAM. The system parameters are summarized as follows: Swimming pool: Table 1; Boiler, TES and STC: Table 2. Weather data were synthesized using Typical Meteorological Year (TMY) files for the city of Montpellier, France (*Repository of Building Simulation Climate Data* 2025). These data enabled integration of the AixLib meteorological model (Maier et al. 2024). To adapt the wind speed from the weather station to the pool surface and Equation 7, a 0.35 coefficient wind adjustment was used. One simulation scenario was performed, in which the pool water temperature was regulated at $28°C$. The boiler operation was managed by a hysteresis controller applied to the TES: the boiler was activated when the TES temperature dropped below the lower threshold and deactivated when the temperature exceeded the upper threshold.

4 Results

Figure 9 presents the simulation results, including the thermal power delivered by the heat exchanger, as well as the air, sky, and pool water temperatures. The pool water temperature is successfully maintained at $28°C$ throughout the simulation period. Air temperature ranges from a minimum of $5°C$ to a maximum of $39°C$, while sky temperature varies from $-12.5°C$ to $37.5°C$. The PI controller regulates the heating power between 0 and $5{,}000kW$ to maintain the target water temperature. Notably, between $4{,}000h$ and $6{,}000h$, the heating demand exceeds $2{,}000kW$, despite elevated air and sky temperatures approaching $28°C$. A reduction in heating demand was expected during this summer period, yet evaporation losses contribute significantly to increased thermal requirements.

Figure 10 shows the evolution of the different thermal losses: radiative (Figure 10.a), thermal cover (Figure 10.b), convective (Figure 10.c), evaporative (Figure 10.d) and fresh water heating (Figure 10.c). Radiative losses range from $0kW$ to $250kW$ and are more pronounced during winter ($0h$-$2{,}000h$ and $6{,}000h$-end), exceeding $150kW$. During summer ($4{,}000h$ –$6{,}000h$), they remain below $100kW$, Figure 10.a. Thermal cover losses (Figure 10.b) range from $0kW$ to $400kW$, with noticeable reductions during summer due to more favorable ambient conditions. Convective losses can be either positive

Table 1. Outdoor swimming pool parameters.

Parameters	*Descriptions*	*Value*	*Unit*	*References*
V_p	Volume	3,750	m^3	(World Aquatics 2024)
A_p	Area	1,250	m^2	(World Aquatics 2024)
L_{ch_p}	Parametric length	8.33	m	(World Aquatics 2024)
D_p	Depth	3.0	m	(World Aquatics 2024)
V_{sto}	Buffer volume	30.0	m^3	(Gauthier-Clerc et al. 2023)
T_p	Reference temperature	28.0	$°C$	(World Aquatics 2021)
$\dot{V}_{flow}$	Circulation flow	250	m^3/h	(World Aquatics 2021)
$\dot{V}_{recy}$	Recycling flow	$5\%V_p$	m^3/day	(Delgado Marín, Vera García, and García Cascales 2019)
ε_c	Cover emissivity	0.5	–	(Li et al. 2021)
ε_w	Water emissivity	0.95	–	(Buscemi et al. 2024)
α_c	Cover absorptance	0.65	–	Author's estimate from grey material
α_w	Water absorptance	0.85	–	(Buscemi et al. 2024)
λ_c	Cover conductivity	0.65	$W/(m^2K)$	(Li et al. 2021)
N_{sw}	Number of swimmers	75	–	Author's estimate
-	Opening hour	12	$hours/day$	Author's estimate

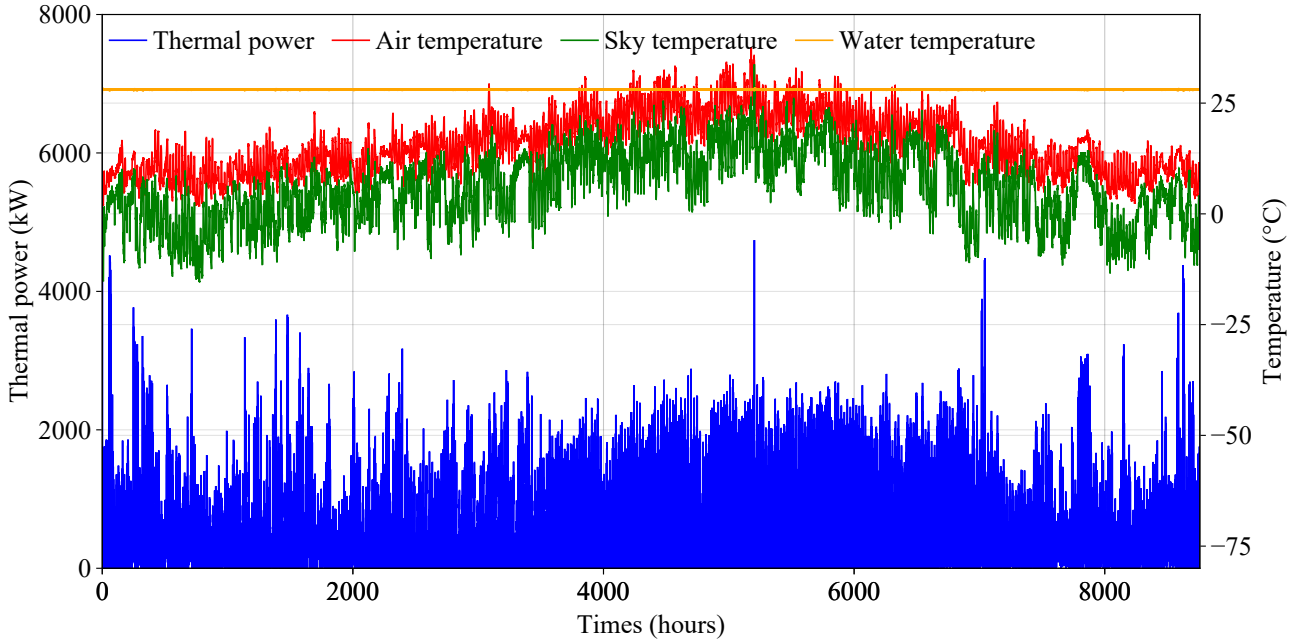


Figure 9. Temporal variation of heat exchanger thermal power, air temperature, sky temperature and pool water temperature.

DOI
10.3384/ecp218891

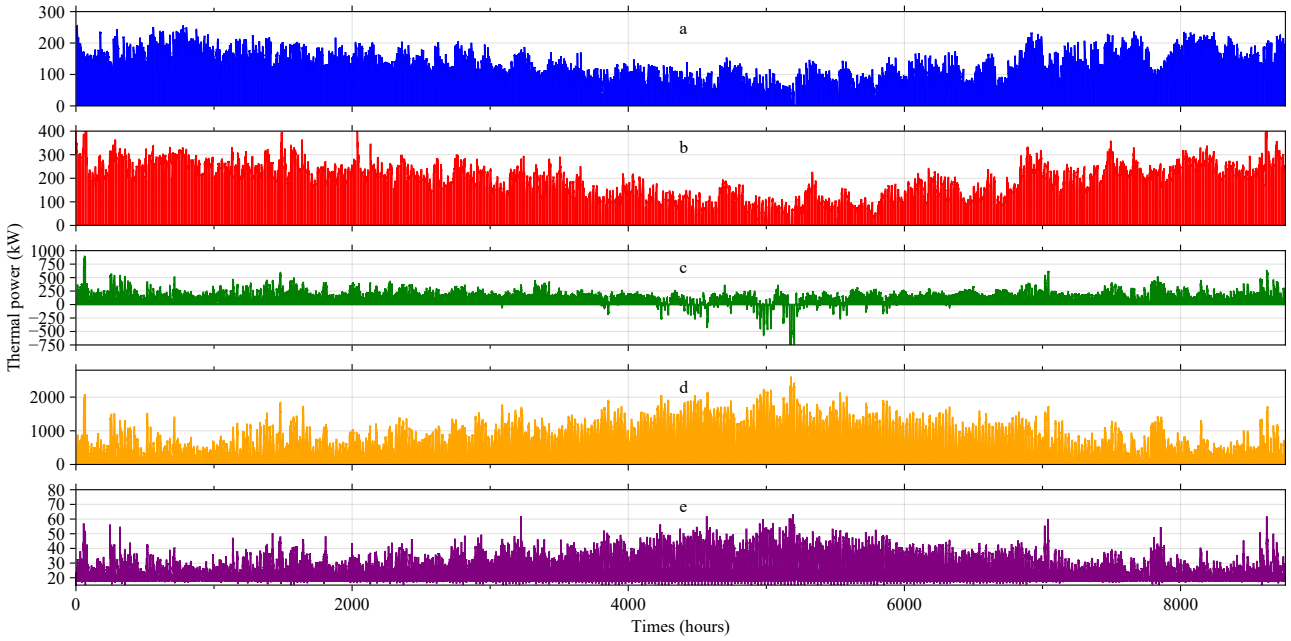


Figure 10. Temporal variation of pool thermal losses. (a) Depicts the radiative losses. (b) Depicts the thermal cover losses. (c) Depicts the convective losses. (d) Depicts the evaporation losses. (e) Depicts the fresh water losses.

Table 2. Boiler, TES and STC parameters.

System	*Parameter*	*Value*	*Unit*
Boiler	Nominal power	6	MW
Boiler	Minimum load	30	$\%$
Boiler	Maximum load	100	$\%$
Boiler	Low threshold hysteresis	40	$°C$
Boiler	High threshold hysteresis	70	$°C$
TES	Volume	800	m^3
STC	Azimuth angle	30	$°$
STC	Tilt angle	-180	$°$
STC	Ground reflectance	20	$\%$
STC	Total area	900	m^2
STC	Collector system	Tube	-

or negative, depending on the Bowen ratio. Notable, at $4,500h$, $5,000h$ and $5,100h$, the pool gains heat from the surrounding air through convection, with power values reaching up to $750kW$. In other periods, convective heat losses reach peaks of up to $875kW$. Evaporative losses (Figure 10.d) are null when the thermal cover is in place but can reach up to $2,500kW$ during uncovered periods, especially in the summer. These losses largely explain the increase in heating demand observed from $4,000h$ to $6,000h$. Fresh water heat losses (Figure 10.e) range from 20 to $60kW$, with lower values observed at night when the pool is covered. These losses correspond to the energy required to heat the replacement water that compensates for evaporation.

Figure 11 illustrates the overall distribution of thermal losses and heat sources. Evaporation accounts for the largest share of thermal losses (63.3%), followed by thermal cover (12.97%), convection (11.33%), radiation (8.88%) and fresh water heating (3.5%). Regarding heat inputs (Figure 11.b), solar heat contributes 25.14% of the total, while the remaining 74.85% is provided by the heat exchanger connected to the boiler and thermal storage system.

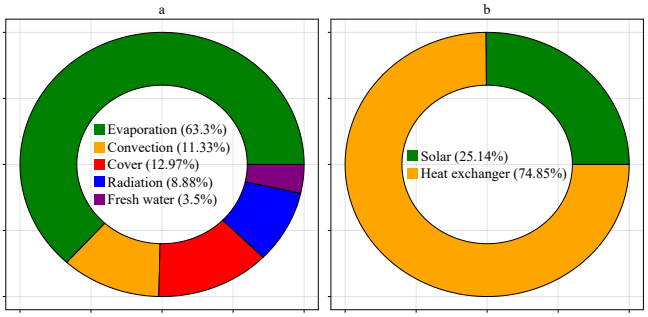


Figure 11. Distribution of the thermal power losses and heating. (a) Depicts the losses distribution with evaporation, convection, thermal cover, radiation and the fresh water added to the swimming pool. (b) Depicts the heating distribution with solar gain and heat added by the heat exchanger.

Table 3 summarizes the annual energy balance. The total heating demand of the pool reaches 1.8×10^{13} J, of which 1.67×10^{13} J is supplied by the boiler, and 0.13×10^{13} J (7.2%) by the solar thermal collectors.

5 Conclusion

This paper has presented a comprehensive dynamic

Table 3. Annual heat demand and production of the energy system.

Pool demand	*Boiler production*	*STC production*
$1.8 \times 10^{13}\ J$	$1.67 \times 10^{13}\ J$	$0.13 \times 10^{13}\ J$

simulation model of a public outdoor swimming pool, integrated with a multi-source energy system comprising a boiler, thermal energy storage (TES), and solar thermal collectors (STC). The model, implemented in Modelica, is based on fine-scale physical representations of the pool's thermal behavior, and has been used to simulate one full year of operation. It accounts for all major heat transfer phenomena, including evaporation, convection, radiation, and interactions with thermal covers. The simulation results demonstrate the model's ability to capture the nonlinear dynamics of the system and provide detailed insights into energy consumption and heat loss distributions. In particular, evaporation is identified as the dominant source of thermal losses, while the solar contribution remains modest in the current configuration.

Future work will focus on coupling this physical model with an Economic Model Predictive Control (EMPC) strategy (Blaud et al. 2023). This advanced control approach will enable the optimization of energy usage and cost by dynamically coordinating the different heat sources. The ultimate objective is to reduce operational expenditures while ensuring thermal comfort and efficient energy use.

Acknowledgements

This work was supported by Akkodis Research, France. The authors' contributions are as follows: Pierre Clément Blaud: conceptualization, methodology, software, simulation, visualization, and writing of the original draft. Imad Mourtaji: supervision and funding acquisition. The authors wish to thank Ludivine Armstrong and Daniel Longworth for their support in defining the pool parameters, and Mounina Tounkara for her valuable help in revising the language and structure of this article.

References

Bernhard, Maïté et al. (2019). "Sensitivity analysis of an outdoor swimming pool under dynamic conditions". In: *Procedia Manufacturing* 35. The 2nd International Conference on Sustainable Materials Processing and Manufacturing, SMPM 2019, 8-10 March 2019, Sun City, South Africa, pp. 124–129. DOI: 10.1016/j.promfg.2019.05.014.

Blaud, Pierre Clément et al. (2023). "From multi-physics models to neural network for predictive control synthesis". In: *Optimal Control Applications and Methods* 44.3, pp. 1394–1411. DOI: 10.1002/oca.2845.

Buscemi, Alessandro et al. (2024). "A novel model to assess the energy demand of outdoor swimming pools". In: *Energy Conversion and Management* 302, p. 118152. DOI: 10.1016/j.enconman.2024.118152.

Cour des comptes (2018). *Piscines et centres aquatiques publics: un modèle obsolète*. Tech. rep. Cour des comptes.

Delgado Marín, J.P., F. Vera García, and J.R. García Cascales (2019). "Use of a predictive control to improve the energy efficiency in indoor swimming pools using solar thermal energy". In: *Solar Energy* 179, pp. 380–390. DOI: 10.1016/j.solener.2019.01.004.

Fritzson, Peter (2011). "Modelica — A cyber-physical modeling language and the OpenModelica environment". In: *2011 7th International Wireless Communications and Mobile Computing Conference*, pp. 1648–1653. DOI: 10.1109/IWCMC.2011.5982782.

Gauthier-Clerc, François et al. (2023). "Operating data of a specific Aquatic Center as a Benchmark for dynamic model learning: search for a valid prediction model over an 8-hour horizon". In: *2023 European Control Conference (ECC)*, pp. 1–8. DOI: 10.23919/ECC57647.2023.10178287.

Kühn, Larissa et al. (2022-09). "Indoor swimming pools: a dynamic simulation model for energy analyses". In: *Proceedings of BauSim Conference 2022: 9th Conference of IBPSA-Germany and Austria*. Vol. 9. BauSim Conference. Weimar, Germany: IBPSA-Germany and Austria. DOI: 10.26868/29761662.2022.47.

Li, Yantong et al. (2021). "Swimming pool heating technology: A state-of-the-art review". In: *Building simulation*. Vol. 14, pp. 421–440. DOI: 10.1007/s12273-020-0669-3.

Liu, Jie, Sung-Chul Kim, and Ki-Yeol Shin (2021). "Feasibility Study and Economic Analysis of a Fuel-Cell-Based CHP System for a Comprehensive Sports Center with an Indoor Swimming Pool". In: *Energies* 14.20. DOI: 10.3390/en14206625.

Maier, Laura et al. (2024). "AixLib: an open-source Modelica library for compound building energy systems from component to district level with automated quality management". In: *Journal of Building Performance Simulation* 17.2, pp. 196–219. DOI: 10.1080/19401493.2023.2250521.

Petzold, Linda R (1982). *Description of DASSL: a differential/algebraic system solver*. Tech. rep. Sandia National Labs., Livermore, CA (USA).

Repository of Building Simulation Climate Data (2025). https://climate.onebuilding.org/. Accessed: 2025-02-15.

Starke, Allan R et al. (2017). "Thermal analysis of solar-assisted heat pumps for swimming pool heating". In: *Journal of the Brazilian Society of Mechanical Sciences and Engineering* 39.6, pp. 2289–2306. DOI: 10.1007/s40430-016-0671-y.

Woolley, Jonathan, Curtis Harrington, and Mark Modera (2011). "Swimming pools as heat sinks for air conditioners: Model design and experimental validation for natural thermal behavior of the pool". In: *Building and Environment* 46.1, pp. 187–195. DOI: 10.1016/j.buildenv.2010.07.014.

World Aquatics (2021). *FINA Facilities Rules*. Tech. rep. World Aquatics.

World Aquatics (2024). *Swimming Pool Certification*. Tech. rep. World Aquatics.

Yuan, Xiaolei et al. (2021). "Heating energy-saving potentials in HVAC system of swimming halls: A review". In: *Building and Environment* 205, p. 108189. DOI: 10.1016/j.buildenv.2021.108189.

Railway Marketplace for Data, Know-How and Services

Stefan H. Reiterer[1] Alexander Meierhofer[1] Ivan Vidovic[2] Marco Forberger[2] Benjamin Stuntner[3] Jochen Nowotny[3]

[1]Virtual Vehicle Research, Austria, {stefan.reiterer,alexander.meierhofer}@v2c2.at
[2]voestalpine Digital Track Management GmbH, Austria, ivan.vidovic@voestalpine.com, m.forberger@vadtm.com
[3]Track Machines Connected Gesellschaft m.b.H., Austria, {benjamin.stuntner,jochen.nowotny}@tmconnected.com

Abstract

Like any other industry the railway sector undergoes great transformations. To make transportation by rail more sustainable and affordable infrastructure, maintenance is key. However, this needs a managed and collective effort of many players in order to make this efficient and easily accessible to experts of many different fields: Measurement experts, Simulation experts and Engineers who perform the maintenance. In this work, we present a market place which leverages modern technologies and standards like GAIA-X, Functional Mock-up Interface (FMI), Distributed Co-Simulation Protocol (DCP) or System Structure and Parameterization (SSP) to provide accessible and easy to use services and retrieve crucial data from other providers, while protecting sensitive data and know-how. This market place is developed under the Umbrella of the The European Rail Joint Undertaking (ERJU) of the European Union. This work is an update of our publication in Eisenbahn Ingenieur Kompendiumm 2024 (**EIK2024**) and contains several additions regarding the implementation and the usage of Modelica standards. A working prototype will be available in 2026.

Keywords: Data Spaces, Railway, FMI, SSP

1 Introduction

The railway industry is currently experiencing a unprecedented boom, which causes countless challenges. While almost all parties involved are working on innovative solutions, promoting, driving and accelerating innovation to face these challenges, it has become clear that a new form of cooperation and exchange is necessary. The time between an idea, prototype and the approval of new products is still extremely long and hence lacks the necessary speed to solve the tasks at hand. One of the most important issues, as it turns out, is the regulated exchange of knowledge, information, data, mathematical models, simulations, etc. in a controlled manner with defined rules and conditions. It is not enough to simply provide data and but also sharing it anonymously between parties. It should be possible for a third party to obtain an existing model, in return for monetary compensation (for example), enrich it with new data in order to possibly develop a better model, while protecting the know-how. On the one hand, this saves a huge amount of resources of all kinds. On the other hand, and this is the much more important aspect, it makes it possible to build on existing knowledge, refine it and thus develop new models, acquire new findings, etc. much faster and bring them timely to market. It also would open up the possibility of collaborative co-simulations. There are many different scenarios like track-vehicle interaction, interaction between vehicles and infrastructure with the environment of many standardized building blocks which could benefit of leveraging co-simulations for cooperation of different parties from manufacturers to maintainers for vehicles and infrastructure.

However, the reality is very different. Not only is it challenging from a technical point of view. There is also no consensus on how such tasks should be handled, how knowledge can be exchanged or obtained and under what framework conditions this is even possible. The legal, financial and security aspects play a decisive role here.

There are also technical challenges. While nowadays data space solutions exist (see Section 2 below) they are hard to configure and setup. This is why they are not as widely used as they could. It is therefore crucial to provide easy to use solutions which need serious development effort.

1.1 The European Rail Joint Undertaking

The European Rail Joint Undertaking (ERJU) is the direct successor to Shift2Rail, the European initiative to promote innovation, technology and cooperation in the rail sector. ERJU is unique in terms of its scale, both financially and in terms of the number of partners involved. The project volume amounts to around €1 billion, the European Union is funding the initiatives with a further €500 million and several hundred companies, scientific partners and rail operators are involved in the program. (*ERJU* 2023). The latter include Deutsche Bahn, the Austrian Federal Railways (ÖBB), ADIF, SNCF and many more. voestalpine Railway Systems (vaRS) is not only the only system solution provider represented in the consortium, it is also one of the 26 founding members of the ERJU. This is partly due to the excellent results from Shift2Rail.

The focus of ERJU is less on basic research or inno-

vations (which are of course welcome). Rather, the program strives to bring solutions, innovations and technologies developed in recent years to the next level (TRL 5 and higher) by demonstrating and ultimately approving the use of these new products under real conditions. The objectives are:

1) Test, approve and launch products more quickly or make them marketable, 2) tighten the cooperation between rail, industry and science and improve on joint solutions and approaches, 3) strengthen the resilience of rail to current and future challenges and thus ultimately achieve the goals of Shift2Rail and ERJU, 4) become the number one most sustainable form of transportation for people and goods.

ERJU runs until 2029 and will be advertised and processed in two calls, although only the first call has currently been published, which will now run until the end of 2026. The next one will follow in the near future.

2 Data Spaces

In this section, a proper definition of data spaces is provided and it is discussed why they are crucial to the future of the railway industry. Also, a short review of existing projects from automotive industries is given to show some examples of how a rail data space might potentially look like.

2.1 Data Space Market Places

In order to simplify cooperation between individual partners in the rail industry, it is important to make the exchange of data and services as easy as possible. Similar efforts and projects already exist in the automotive sector. Examples include Catena-X (catena-x.net) for the supplier sector and the ENVITED Marketplace (www.envited.market) which is used to exchange geodata and maps to aid the development of driver assistance systems. A marketplace for the exchange of geodata of infrastructure and other data for the various players in the European rail sector is currently developed to make essential data accessible to developers and service providers. This often poses a challenge, as IPs and know-how have to be protected accordingly. This problem should now be solved. One particular aspect that needs to be considered within this framework is the provision of simulations as a service, as implemented in the ÖBB Rail4Future project (`rail4future.com`), among others. It should be possible to purchase the necessary data sources to be able to carry out complex simulations by oneself or through partners. These services can then be purchased or offered on the planned marketplace. Further possible use cases of the marketplace will be discussed in detail in the course of the article. One of the most important considerations is that centralized approaches should be avoided: So, instead of a central location where all data and services are collected, a federated system should be established where all players can offer their data and services and potential customers can use the services locally. The advantage of this is that large amounts of data are not duplicated many times and data waste is avoided. In addition, data protection is guaranteed and the integrity and trustworthiness of the data offered is ensured. In order to meet and take account of these requirements, it is necessary to set up a "Federated Data Space". This term is discussed in more detail in the next section, which also looks at other important technical requirements.

2.2 Federated Data Spaces

A federated data space can essentially be characterized by four features:

1. Unlike databases, data spaces do not store data, but only provide links to the actual data.
2. They do not force the data provided into a standardized schema, but can be made available as they are.
3. Redundancy of data is permitted.
4. They can overlap and a participant can be part of several data spaces (see (Otto 2022), (Franklin, Halevy, and Maier 2005)).

GAIA-X (gaia-x.eu) serves as the basis for this project. It is a European initiative to create a secure and trustworthy digital ecosystem for data and services. This initiative aims to strengthen Europe's sovereignty in data processing and digital infrastructure, particularly in the areas of cloud computing and data management. Federated data spaces (in the sense of GAIA-X) also offer so-called "federation services" that function across the boundaries of several data spaces (see (Franklin, Halevy, and Maier 2005)). The architecture of GAIA-X is based on the IDS ("international data space") architecture (see (Siska, Karagiannis, Drobics, et al. 2023), (Otto et al. 2018)), which provides for the connection of the individual participants via so-called IDS connectors (see (Siska, Karagiannis, Drobics, et al. 2023) and Figure 1 for a graphical representation). These "connectors" help to convert the data correctly and enable any payment transactions between the individual parties.

GAIA-X also stipulates that data transfers should be secured by corresponding policies and certificates. The safeguarding and certification of simulation data and results poses an additional challenge, as the corresponding services that are supposed to perform this safeguarding automatically must be co-developed by the providers due to the fact that this assessment is anything but trivial. Such aspects were considered in the UPSIM project (`upsim-project.eu`), among others, and corresponding processes, standards and automated toolkits were proposed to determine the credibility of data and results in the context of simulations (see (Ahmann et al. 2022)). In order to be able to offer simulations as a distributed service, the aspect that these services can possibly be connected in series must also be taken into account. This aspect is already being considered in Rail4Future. For this purpose,

DOI 10.3384/ecp218901

so-called co-simulation process graphs are used (see (Reiterer, Schiffer, and Benedikt 2022)), which represent a description of the process and dependencies on data sources and can be managed within graph databases. As the name suggests, these special graphs represent processes and co-simulations. By definition, a co-simulation is a collection of individual simulations that exchange their results with each other (see (Gomes et al. 2018)). With the help of these "co-simulation process graphs", instructions for automation software (e.g. Jenkins) can be generated and executed using a service. The special graph structure also makes it possible to represent the occurring co-simulations between different simulation participants and to generate a configuration of the co-simulation that can be received by different co-simulation protocols in order to enable the orchestration of the co-simulation and the data exchange between the individual participants (see (Reiterer, Schiffer, and Benedikt 2022) for a simple example). This can also be used to optimize process flows and ultimately use valuable resources, such as computing capacity and licenses, more efficiently (see (Reiterer and Kalab 2021)). This fairly general representation not only facilitates automation, it also enables the standardization and exchange of processes, as these can be stored and passed on like recipes and individual process steps and data sources can be exchanged or expanded. This simplifies collaboration between individual participants enormously when it comes to offering and exchanging services. For example, it is now possible to offer not only individual simple simulations as a service, but also complex co-simulations that consist of many partial simulations and are fed from a wide variety of data sources, and not necessarily just in one central location, but distributed. As co-simulation process graphs only store links and instructions instead of entire data sets or simulations, they form a good basis for fully implementing the federated data space principle in the area of simulation.

It may be noted that data space technologies are in a constant development and changes to the core technology can happen any time. However, even with the advent of new versions, technologies and standards as well as the fundamental concepts stay the same as in GAIA-X.

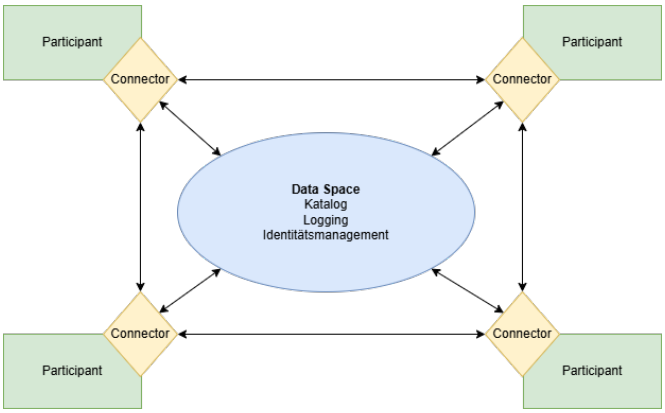


Figure 1. Structure of the Federated Data Space

3 The Railway Market Place

In the course of the ERJU cooperation, as described in Section 1.1, a demonstrator was defined along which the concept of a marketplace for railroad and railroad maintenance data is to be tested. The aim of this ERJU marketplace demonstrator is to provide the basic functionality for customer interaction and interaction with external platforms and federated data spaces. The ERJU Marketplace Demonstrator is used to gather customer experience during a user-friendly test phase. The concept is also the basis to present our innovation to a broad field of potential customers and partners - to get them on board by using our marketplace in the first version as their point of sale for the rail market on the one hand and as consumers on the other. Both customers and partners are spread across the entire rail industry. Railroad companies, manufacturers but also operators of machines, infrastructure and measurement technology. Every single company that wants to use data and services relating to rail transport and infrastructure should be able to benefit from the railroad marketplace.

The targeted customers do not have a broad overview of existing data and services in this area. Concepts for segmenting and structuring the data are therefore of high value, as most customers are specifically interested in finding the information that is valuable for the region or rail network for which they are responsible. Since there is currently no well-organized overview and data exchange facility across the rail industry, the marketplace helps to reduce the effort of creating the same data multiple times by different companies. With faster and wider availability of data, decisions can be made based on data and objectively, ultimately reducing the negative impact on infrastructure availability. In addition, any energy that can be saved is also of great value to our environment as it helps to reduce CO2 pollution and other negative side effects.

Potential partners are also spread across the industry and offer data and/or services relevant to the rail and infrastructure management business. They can use the marketplace as an additional sales channel to become more visible in this market area. This often results in new opportunities from previously unknown interested parties for the data for their own special use cases. The exchange of data should also be made easier for the partners through clearly defined format and quality criteria that should apply to both parties of possible transactions. A mockup of the marketplace view is shown in Figure 2.

It is also important to highlight that usage of data space technologies is complicated, hence, it is important to provide ways like simple web interfaces and services to hide the burden of configuration from the end users. This is one of the key aspects for the design of the market place we propose here and the main reason such data space solutions are not broadly adopted as there is a lot of configuration effort necessary to allow safe data transfers and provide services in a consistent manner.

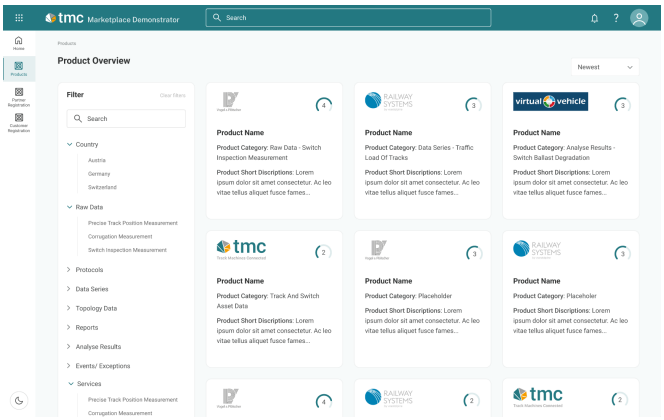


Figure 2. Marketplace View

3.1 Architecture & Services

An important milestone in the ERJU project and the implementation of the marketplace demonstrator is the definition of the architecture. The focus here is on state-of-the-art software technologies and microservice concepts, as these have proven to be extremely powerful in meeting the requirements for flexibility, scalability and maintainability in the fast-moving world of digitization. The basic idea behind microservice architecture is to divide a large and complex software application into smaller, independent services called microservices. Each microservice is responsible for a specific function or task and can be developed, deployed and scaled autonomously.

A detailed overview of the architecture can be seen in Figures 3 and 4. Figure 3 shows the architecture of the marketplace at its core, while Figure 4 shows how external services can be integrated.

The marketplace consists of the core components described in the following sections.

3.1.1 Data Catalog

For the marketplace, the data catalog plays a crucial role in the provision of information and services. This catalog, which is part of the service repositories and the data catalog repository, is at the heart of this state-of-the-art rail data marketplace. The service repository is a carefully managed collection of service self-descriptions. These self-descriptions contain a wealth of information that enables potential customers to understand exactly what type of service is being offered. The details of the data types are described in the data catalog repository. This tells customers what content is available and in what form it can be provided. This clear overview enables customers to search specifically for the data they need. Both data and services are presented as products on the marketplace interface. This creates a user-friendly environment in which customers can easily search for the information they need. Not only provides the data catalog overview pages, but also the product detail pages use these repositories to present the products and product information in an appealing way on the marketplace GUIs.

3.1.2 TrackDB: Railway Infrastructure Topology

To filter data and services as accurately and efficiently as possible, it is essential to not only locate the data country-specifically and geographically, but also to map it within the topology of a railroad infrastructure operator. Since this network model is not static but constantly changing, TrackDB proves to be a crucial tool for mastering the dynamics of the railroad topology network. TrackDB is an innovative service and part of the marketplace that makes it possible to create and manage the rail topology network based on various sources. This service goes far beyond traditional static network modeling and continuously adapts to the requirements and changes in the rail infrastructure by opening up freely accessible sources but also accessing partly (partially) closed data based on partnership with infrastructure operators.

The key to TrackDB's effectiveness lies in its ability to dynamically capture and update information about track sections, switches, signals and other elements of the rail network. This is done in real time, allowing users to access up-to-date data at all times. This means that the content provided by service partners and data providers can also be clearly assigned to the network, allowing this information to be filtered down to sections of specific tracks or even individual components such as switches.

3.1.3 Services

Another way to expand the product range is to host third-party services and ensure the smooth operation of these services. This aspect significantly expands the functionality of the marketplace and offers service providers the opportunity to integrate their specialized solutions directly into the platform. The "Service and Data Center" marketplace acts as a central hub for hosted services from third-party providers. Providers have the opportunity to host their services in this environment in order to make them accessible to a broader target group. This opens up new business opportunities and promotes innovation in the rail industry.

3.2 Simulation Services and Co-Simulation Standards

A special key feature of the market place is to provide simulation services for predictive maintenance.

To enable the seamless integration of these services, service providers must offer "black box" services that can communicate via standardized interfaces. These interfaces serve as templates and enable interaction between the hosted services and the data available on the platform. This ensures that the hosted services can work effectively with the required data inputs.

If a service is filled with data and executed, the next step is to output results and feed them back into the marketplace. Hosted services should be able to provide the results in a standardized file format. This facilitates integration and data exchange between the various services and users on the platform. The ability to integrate hosted

DOI
10.3384/ecp218901

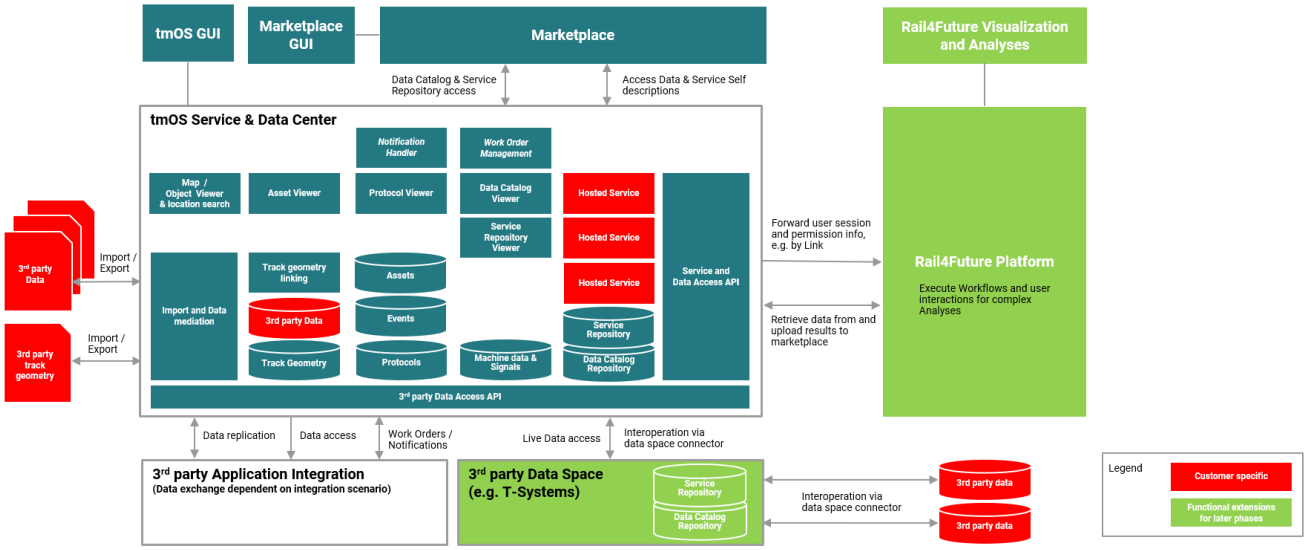


Figure 3. High level System Architecture

services into the platform opens up a world of potential. It allows customers to access a wide range of specialized services, from data analytics to predictive maintenance, without having to leave the platform and without having to worry about running their own platform.

3.2.1 The Role of Co-Simulation Standards in the Context of the Market Place

Co-Simulation standards like the Functional Mockup Interface (FMI) (FMI-Working-Group 2020) and the Distributed Co-Simulation Protocol (DCP) (Modelica Association Project DCP 2019) play a crucial role as they are leveraged to enable easy integration of new simulation services. First, mappings between input data and simulations can be created with ease and tools for reading out inputs and writing outputs can be provided. Second, they provide a standardized interface developers and engineers can rely on to provide a unified way to read and write data from and to simulations. This standardization is crucial as simulations are very diverse and rely on very different forms of input. The metadata file an FMU or DCP participant provides ways to auto generate forms for data provision as other participants know what they have to provide in order to make the simulation run and what to expect as output. We show an example in Section 3.2.2. Since the FMU defines the proper in and outputs this can be used to tell a potential user which data they need to provide (for example as CSV file) and which names the fields have. Another important benefit of co-simulation standards like FMI is that FMUs can be stored as standalone files that can be shared between different parties. This allows of easy exchange of models.

The only real drawback FMUs have is that it is not always easy to guarantee proper runtimes of the underlying models. So proper set-up sandbox environments are needed as well. A data space can help with that as the models can run with an enclosed environment (e.g. on site with the owning party) and just input and output data is exchanged, which only works for non real-time computations of course. However, solutions to this problem also could be provided in the future.

Furthermore, the System Structure and Parameterization (SSP) standard (see e.g. (Hällqvist et al. 2021)) can be used to link different simulation services together. This enables cooperation between different simulation domains and helps with automatic service execution as many tools support these open Modelica standards. As already mentioned, this can be also supported by graph based process descriptions (Stefan H. Reiterer and Clemens Schiffer 2021)(M. Stefan H. Reiterer and Clemens Schiffer 2023) which allow to make reproducible computation processes by auto generating and orchestrating execution pipelines. The viability of this approach in the context of the rail industry was already demonstrated (Ozan Kugu 2023).

Additionally, the data space architecture enables us to use data space connectors to safely upload sensitive data which is necessary for a multitude of simulations. This helps to protect know how of third parties and adds an additional layer of security.

3.2.2 An Example Use Case: Ballast Degradation

As an example to illustrate the diverse interaction possibilities on the marketplace, a use case was chosen called "Monitoring of Track Settlement and Identification of Single-Faults at Switches". One of the goals there involves the monitoring and prediction of ballast degradation in the switch-area, especially below the frog. In order to successfully execute this use case, both the service and all required relevant information sources must be defined in the service repository and the data catalog repository. It should be noted that we want to outline the process to give a better understanding of how this works, rather than explaining the model in detail, because if a simulation is offered as a service the inner workings of a simulation

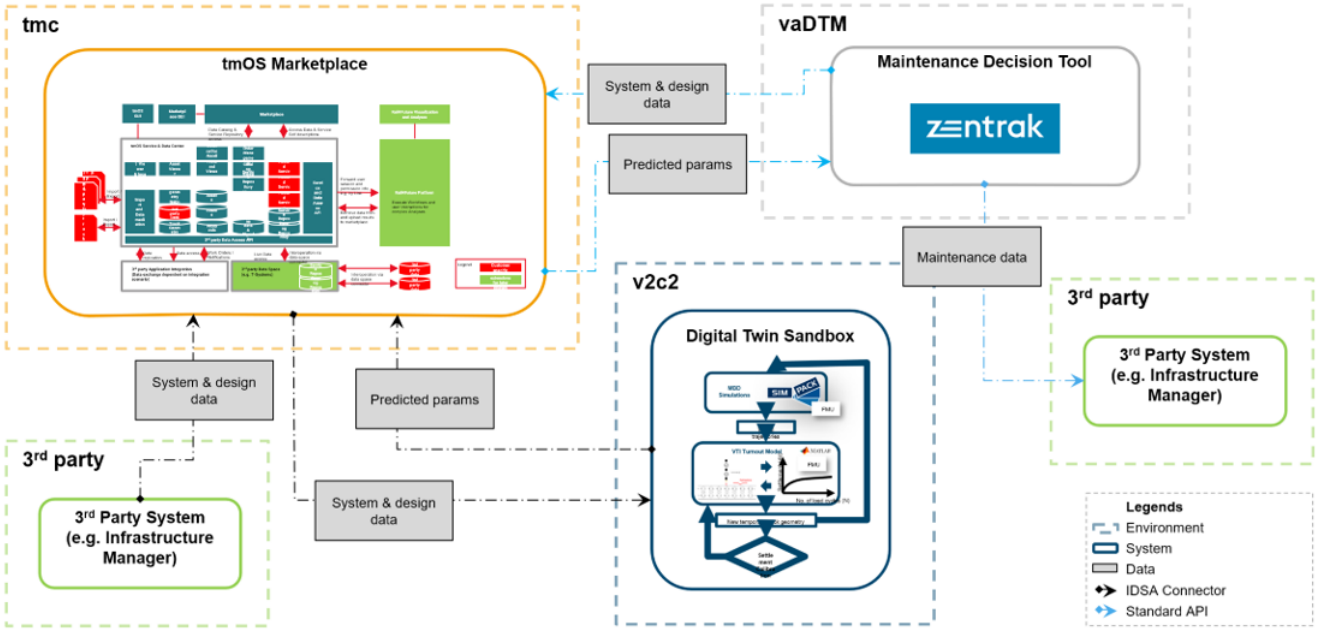


Figure 4. High level System Architecture with Integrated Services

have to be abstracted away within an FMU beforehand to make it usable. However, this heavily depends on the tools in use and would go far beyond of the scope of this work. The execution of this service offering requires several steps on the product offering side and is intended to demonstrate the possibilities for querying multiple, different data from different providers for a specific track section. Several interactions with menu selections and the interactive map function are highlighted. A key aspect is also to show how quickly the customer can filter the area of interest.

Here, the consumer not only buys the needed "raw data" for the simulation from data providers on the market place but also the (automated) execution of a complex simulation for calculating ballast degradation, which requires this "raw data" as input data. The algorithm itself does not run directly in the marketplace but in a separate Rail4Future simulation environment developed by Virtual Vehicle (`www.rail4future.com`).

The simulation itself runs within a Functional Mockup Unit. A generated pipeline is triggered which starts the FMI master. The master reads the data and executes the simulation and then writes out the simulation results.

After the customer has completed the purchase, the marketplace will exchange the relevant data with this environment, initiate the simulation there and finally make the processed data available to the customer again on the marketplace. See Figure 5 for a mock up view on the service.

Nevertheless, there are many more examples of simulations ranging from mechanical simulations (for example vehicle-track interaction), thermal simulations (behavior of infrastructure under changing temperature) and many more. Many tools which are used (for example Matlab or Simpack) provide export capabilities for Modelica standards like FMI or SSP which makes it an obvious choice regarding exchange and co-simulations of models.

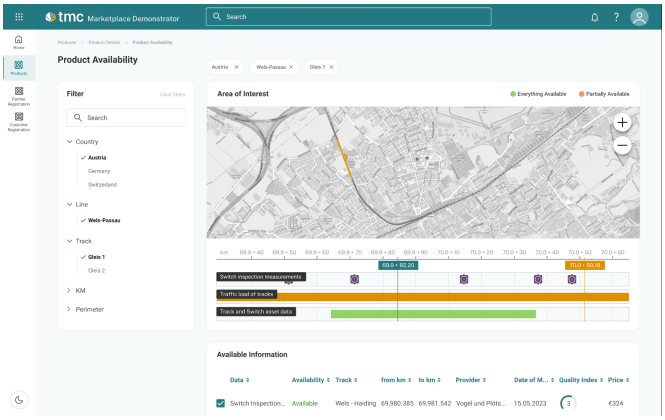


Figure 5. GUI View of the Ballast degradation Use Case

4 Conclusion

Managing Railway infrastructure needs a lot of know-how data and coordinated joint effort. Modern standards and initiatives like GAIA-X, FMI, DCP and SSP provide the foundation to enable the railway industry for a modern, comfortable and standardized workflow, and provides non-experts with a place where they can easily buy services which can be used without much effort.

In this work we demonstrated how such a market place can look like and a practical example how this can work. A demonstration shall be ready in 2026 and is open for collaboration. The ERJU offers the ideal platform for the joint development of further features and the promotion and on-boarding of additional partners. Infrastructure op-

DOI
10.3384/ecp218901

erators and rail transport companies in particular are invited to participate and get involved. Ultimately, they are the ones who benefit from accelerated developments, reduced costs and higher availability and can ultimately pass this on to their end customers.

Acknowledgements

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or Europe's Rail Joint Undertaking Neither the European Union nor the granting authority can be held responsible for them. The publication was written at Virtual Vehicle Research GmbH in Graz and partially funded within the COMET K2 Competence Centers for Excellent Technologies from the Austrian Federal Ministry for Innovation, Mobility and Infrastructure (BMIMI), Austrian Federal Ministry for Economy, Energy and Tourism (BMWET), the Province of Styria (Dept. 12) and the Styrian Business Promotion Agency (SFG). The Austrian Research Promotion Agency (FFG) has been authorised for the programme management.

References

Ahmann, Maurizio et al. (2022). "Towards continuous simulation credibility assessment". In: *Modelica Conferences*, pp. 171–182.

ERJU (2023). Web Site. Accessed: https://rail-research.europa.eu/about-europes-rail/europes-rail-reference-documents/ 2025-04-18.

FMI-Working-Group (2020). *Functional Mock-up Interface for Model Exchange and Co-Simulation*. https://github.com/modelica/fmi-standard/releases/download/v2.0.2/FMI-Specification-2.0.2.pdf. Accessed: 2021-04-12, V 2.0.2.

Franklin, Michael, Alon Halevy, and David Maier (2005). "From databases to dataspaces: a new abstraction for information management". In: *ACM Sigmod Record* 34.4, pp. 27–33.

Gomes, Cláudio et al. (2018). "Co-simulation: a survey". In: *ACM Computing Surveys (CSUR)* 51.3, pp. 1–33.

Hällqvist, Robert et al. (2021). "Engineering domain interoperability using the system structure and parameterization (ssp) standard". In: *Modelica Conferences*, pp. 37–48.

Modelica Association Project DCP (2019). *DCP Specification Document, Version 1.0*. Linköping, Sweden: Modelica Association. URL: http://www.dcp-standard.org.

Otto, Boris (2022). "A federated infrastructure for European data spaces". In: *Communications of the ACM* 65.4, pp. 44–45.

Otto, Boris et al. (2018). "Ids reference architecture model. industrial data space. version 2.0". In.

Ozan Kugu Rebecca Nowak, Stefan H. Reiterer et. al. (2023-10). "An FMI- and SSP-based Model Integration Methodology for a Digital Twin Platform of a Holistic Railway Infrastructure System". en. In: *15th Modelica Conference 2023*, pp. 717–726. DOI: 10.3384/ecp204. URL: https://ecp.ep.liu.se/index.php/modelica/issue/view/83/86 (visited on 2024-01-18).

Reiterer, Stefan H and Michael Kalab (2021). "Modelling deployment pipelines for co-simulations with graph-based metadata". In: *International Journal of Simulation and Process Modelling* 16.4, pp. 333–342.

Reiterer, Stefan H, Clemens Schiffer, and Martin Benedikt (2022). "A Graph-Based Metadata Model for DevOps in Simulation-Driven Development and Generation of DCP Configurations". In: *Electronics* 11.20, p. 3325.

Siska, Veronika, Vasileios Karagiannis, Mario Drobics, et al. (2023). "Building a dataspace: Technical overview". In: *Gaia-X Hub Austria*.

Stefan H. Reiterer and Clemens Schiffer (2021-09). "A Graph-Based Meta-Data Model for DevOps in Simulation-Driven Development and Generation of DCP Configurations". en. In: *14th Modelica Conference 2021*, pp. 411–417. DOI: 10.3384/ecp21181411. URL: https://ecp.ep.liu.se/index.php/modelica/article/view/219 (visited on 2022-07-26).

Stefan H. Reiterer, Mario Schwaiger and Clemens Schiffer (2023-10). "Graph-Based Meta-Data Model for DevOps: Extensions to SSP and SysML2 and a Review on the DCP standard". en. In: *15th Modelica Conference 2023*, pp. 159–166. DOI: 10.3384/ecp204. URL: https://ecp.ep.liu.se/index.php/modelica/issue/view/83/86 (visited on 2024-01-18).

DOI
10.3384/ecp218901

Dynamic Simulation of Off-Grid Energy Island with Wind-PV-Storage Hydrogen Production

Li Zuo[1] Yuanhui Dong[1] Shubin Zhang[1] Yuxin Li[1] Haiming Zhang[1] Ji Ding[1] Qi Liu[1] Fanli Zhou[1] Liping Chen[2]

[1]Suzhou Tongyuan Software & Control Technology Co., Ltd, China, {zuoli, dongyuanhui, zhangsb, liyuxin, zhanghm, dingj, liuq, zhoufl}@tongyuan.cc
[2]Huazhong University of Science and Technology, Wuhan, China, chenlp@hust.edu.cn

Abstract

This study proposes a novel discrete-solving algorithm implemented on the MWORKS.Sysplorer platform for power electronics simulation, addressing dynamic stability challenges in multi-energy integrated systems with high renewable energy penetration. For an offshore energy island, detailed component models including direct-drive wind turbines, photovoltaics, battery energy storage, and alkaline electrolyzers are developed and integrated into a standalone DC system. The energy storage system controls bus voltage to enable constant-power operation of electrolyzers. The implemented discrete algorithm provides critical multi-timescale simulation capabilities, advancing the planning and construction of offshore energy island projects.

Keywords: modelica, discrete algorithm, offshore energy island, integrated energy systems, electrolyzer

1 Introduction

The global energy structure has been undergoing rapid decarbonization and sustainable transformation recently. As an emerging paradigm for large-scale renewable energy integration, offshore energy islands have garnered significant attention due to their ability to synergistically couple diverse energy forms, including offshore wind, solar photovoltaics (PV), energy storage, and hydrogen production (wind-solar-storage-hydrogen) into a unified system. Offshore wind power demonstrates superior energy generation efficiency and economic viability compared to its onshore counterparts (Bilgili, Yasar, and Simsek 2011), attributed to its higher wind energy density and more stable operating conditions. Unlike conventional offshore wind platforms, offshore energy islands not only improve renewable energy utilization and reduce long-distance transmission losses but also enable dynamic multi-energy coordination through integrated "electricity-hydrogen" solutions, thus delivering stable and clean energy supplies to coastal and island regions (Igbinenikaro, Adekoya, and Etukudoh 2024).

However, the high proportion of renewable energy sources poses significant challenges to the transient stability and operational security of energy island systems. Under islanded or weak grid conditions, power electronic converter-dominated energy conversion substantially reduce system equivalent inertia, leading to frequent fluctuations in voltage and power, diminished stability margins, and potential risks of equipment overload or operational failure—critical threats to system reliability. (Kundur 2007)(Salmador Gonzalez 2016). To mitigate the stochastic fluctuations of renewable generation, water electrolysis for hydrogen production has emerged as a key flexibility resource for offshore energy islands, using its high energy density and large-scale regulation capacity. By converting surplus wind/PV power into storable hydrogen, this technology enables rapid energy release during peak demand or power deficits, providing essential power buffering and dynamic support to enhance system stability (Arsad et al. 2022)(Xing et al. 2018)(Sterner 2009).

Current research predominantly focuses on optimizing integrated energy dispatch strategies to address renewable intermittency, particularly through adaptive operational mode adjustments for hydrogen production systems. Coordinated multi-electrolyzer switching strategies (Fang and Liang 2019) have been proposed to balance fluctuating wind/PV power. Subsequent studies have refined these approaches to minimize electrolyzer cycling, such as rotational coordination (Xiaojun, Congying, and Hong 2021), duty-cycling optimization (Shen et al. 2018), and start-stop frequency reduction (Tiejiang et al. 2022). Nevertheless, these dispatch-oriented optimizations often overlook critical equipment operational characteristics, limiting their applicability to real-world implementations. Consequently, high-fidelity modeling and simulation of actual energy island components are imperative to bridge this gap between theoretical strategies and engineering practices.

Modelica exhibits unique advantages in multi-physics unified modeling, facilitating detailed equipment-level simulations. Previous studies (Webster and Bode 2019) have developed proton exchange membrane (PEM) electrolyzer models using Modelica, achieving close alignment with experimental results. To further investigate electrochemical dynamics, dedicated libraries like the Electrochemical Reactors Library (Pyschik et al. 2025) have been established. However, existing research pre-

dominantly focuses on standalone electrolyzer modeling, lacking co-simulation frameworks integrating renewable generation sources. Moreover, when interfacing electrolyzers with power grids, most Modelica-based tools face inherent limitations in handling complex dynamic processes. First, in nanosecond-to-microsecond timescale simulations, frequent power electronic switching induces high-frequency dynamics that may trigger numerical oscillations and poor real-time performance. Second, the absence of specialized high-frequency model libraries for power electronic systems within the Modelica ecosystem necessitates manual development of fundamental components, increasing modeling complexity and compromising simulation efficiency.

This study leverages the MWORKS. Sysplorer platform—a Modelica-based system modeling and simulation tool developed by Suzhou Tongyuan Soft Control Technology Co., Ltd.—to construct a high-fidelity offshore energy island model. The platform's advanced electromagnetic transient solver and domain-specific libraries enable precise modeling of key components, including wind turbines, PV arrays, lithium-ion battery storage, and alkaline electrolyzers. The proposed system supports multi-timescale simulations of power system dynamics, providing robust technical validation for parameter optimization and control strategy development in offshore energy island projects.

2 Model Components

2.1 Direct-Drive Wind Turbine

As shown in Figure 1, the topology of a conventional direct-drive wind turbine system typically consists of three key components: wind turbine blades, a permanent magnet synchronous generator (PMSG), and a power converter along with its controller.

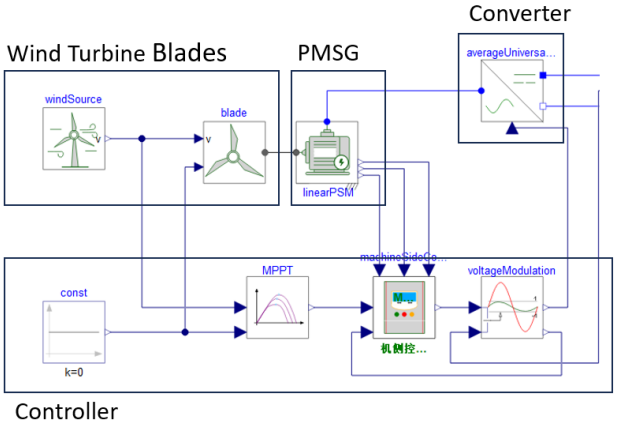


Figure 1. Topology of a conventional direct-drive wind turbine system

Based on aerodynamic theory, the mechanical power output of the wind turbine P_m can be calculated as:

$$P_m = \frac{1}{2} C_p(\lambda, \beta) \rho \pi R^2 v^3 \tag{1}$$

where, C_p is power coefficient, representing wind energy conversion efficiency; λ is the tip-speed ratio; β is blade pitch angle; ρ is the air density; R is rotor radius; v is the wind speed.

The power coefficient C_p, a critical parameter characterizing turbine efficiency, is expressed as a function of β and λ:

$$C_p = c_1 \left(\frac{c_2}{\lambda_i} - c_3 \beta - c_4 \beta^{c_5} - c_6 \right) e^{-\frac{c_7}{\lambda_i}} \tag{2}$$

$$\frac{1}{\lambda_i} = \frac{1}{\lambda + c_8 \beta} - \frac{c_9}{\beta^3 + 1} \tag{3}$$

where $c_1 - c_9$ are curve-fitting parameters that vary slightly across different turbine designs.

The drive train is modeled using a single-mass formulation:

$$H \frac{\mathrm{d}\omega_r}{\mathrm{d}t} = T_m - T_e \tag{4}$$

where, H is the total rotational inertia of the direct-drive system; T_m is the mechanical torque; T_e is the electromagnetic torque.

In the synchronous rotating dq-reference frame, the voltage equations are:

$$\begin{cases} L_s \dfrac{\mathrm{d}i_d}{\mathrm{d}t} = u_d - R_s i_d + \omega_s L_s i_q \\ L_s \dfrac{\mathrm{d}i_q}{\mathrm{d}t} = u_q - R_s i_q - \omega_s L_s i_d + \omega_s \psi_f \end{cases} \tag{5}$$

where, L_s is the stator inductance; R_s is the stator resistance; u_d, u_q are d- and q-axis stator voltages; i_d, i_q are d- and q-axis stator currents; ψ_f is the permanent magnet flux linkage; ω_s is the synchronous rotational speed.

The electromagnetic torque T_e is governed by:

$$T_e = \frac{3}{2} n_p \left[\psi_f i_q + (L_d - L_q) i_d i_q \right] \tag{6}$$

where n_p is the Number of pole pairs.

The operating range of a wind turbine generator can generally be divided into four regions: Pre-cut-in standby phase, maximum power point tracking (MPPT) phase, constant speed transition phase, and rated power regulation phase. In the maximum power point tracking phase, the rotational speed of the wind turbine is less than the maximum allowable speed. An optimal torque control mechanism, known as the Maximum Power Point Tracking (MPPT) system, is implemented to ensure that the wind turbine operates at the maximum wind energy utilization coefficient.

DOI
10.3384/ecp218909

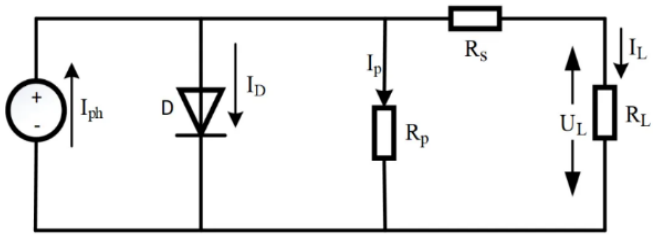


Figure 2. Photovoltaic equivalent circuit schematic

2.2 Photovoltaic

Photovoltaic power can be equated to a photovoltaic current source, a parallel diode, a series resistor R_s, and a parallel resistor R_p, as shown in Figure 2.

The output current I and photogenerated current I_{ph} relationships in photovoltaic systems are given by:

$$I = I_{ph} - I_s\left(e^{\frac{(V+IR_s)}{NV_t}} - 1\right) - \frac{V+IR_s}{R_p} \tag{7}$$

$$I_{ph} = I_{ph0}\frac{I_r}{I_{r0}} \tag{8}$$

where, I_r is the photogenerated current under actual irradiation; I_{ph0} is the reference photogenerated current at standard test conditions; I_s is the diode reverse saturation current; N is the diode ideality factor; V is the output voltage.

In practical PV systems, photovoltaic arrays are constructed through series-parallel configurations of multiple PV modules to meet diverse load power requirements. The system employs a Perturbation and Observation method (P&O) MPPT control strategy (Figure 3). The P&O method periodically perturbs the operating voltage slightly, then compares power before and after perturbation: continuing the direction if power increases, reversing it if power decreases. This iteration adapts to irradiance and temperature changes that shift the maximum power point, enabling stable tracking through the "perturb-and-observe" cycle. It keeps PV modules operating optimally, maintaining maximum power extraction and enhancing system efficiency under fluctuating conditions.

2.3 Energy Storage Battery

The equivalent circuit of the storage battery consists of a signal voltage source, a parallel self-discharge resistor R_{SD}, a charging dynamic model, and a series resistor R_0. The model's base voltage-charge state equation is described by the following equation:

$$V = V_0\left(\frac{SOC}{1-\beta(1-SOC)}\right) \tag{9}$$

where SOC is the ratio of the current charge to the rated capacity; V_0 is the voltage of a fully charged battery with no load; β is a constant that ensures the voltage equals V_1 when the charge is Ah_1. V_1 and Ah_1 can be given

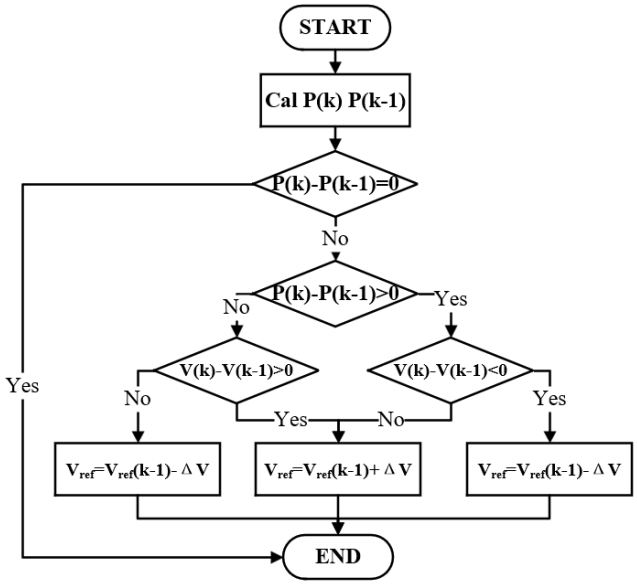


Figure 3. Perturbation and Observation method

through parameter setting, Ah_1 is the charging capacity/charge quantity corresponding to the voltage V_1 under no load, and V_1 is less than the rated voltage.

In energy storage control, the energy storage battery adopts voltage and current dual-loop control, and switches the charging and discharging states of the energy storage battery by outputting control signals. The voltage loop serves as the outer loop, taking the system's DC bus voltage as the control target. It compares the actual voltage with a preset reference voltage, and after processing the deviation through a PI regulator, outputs a current reference value. The current loop functions as the inner loop, which takes the current reference value output by the voltage loop as its target and eliminates the deviation between the actual current and the reference current via PI regulation, thereby generating precise control signals to regulate the charging and discharging processes.

2.4 Alkaline Electrolyzer

During electrolysis, DC power is applied across two electrodes in the electrolyte:

- Anode (Oxidation):

$$4OH^- \rightarrow O_2\uparrow + 2H_2O + 4e^-$$

- Cathode (Reduction):

$$2H_2O + 2e^- \rightarrow H_2\uparrow + 2OH^-$$

The rate of an electrolytic cell reaction is closely related to the current in the cell. According to Faraday's law (Ulleberg 2003), the rate of production of oxygen and hydrogen is directly proportional to the current I passing

through the electrolyzer, according to the following equation:

$$\begin{cases} \dot{N}_{\mathrm{O_2}} = \dfrac{I}{4F} & \text{(Oxygen production rate)} \\ \dot{N}_{\mathrm{H_2}} = \dfrac{I}{2F} & \text{(Hydrogen production rate)} \end{cases} \tag{10}$$

where, $\dot{N}_{\mathrm{O_2}}$ is the molar production rate of oxygen; $\dot{N}_{\mathrm{H_2}}$ is the molar production rate of hydrogen; I is the applied current; F is Faraday's constant (96,485 C/mol).

During the electrolysis of water, the anode and cathode follow the principle of mass conservation, respectively, which is used to describe the dynamic equilibrium between reactants and products:

$$\frac{dN_{H_2O}}{dt} = N_{H_2O}^{in} - N_{H_2O}^{out} + N_{H_2O}^{gn} \tag{11}$$

$$\frac{dN_{O_2}}{dt} = N_{O_2}^{in} - N_{O_2}^{out} + N_{O_2}^{gn} \tag{12}$$

$$\frac{dN_{H_2}}{dt} = N_{H_2}^{in} - N_{H_2}^{out} + N_{H_2}^{gn} \tag{13}$$

where, superscript *in* represents the inflow of substances; superscript *out* represents the outflow of substances; superscript *gn* represents the net generation of substances due to electrochemical reactions (a negative value indicates consumption in the reaction); suffix H_2O, O_2, H_2, respectively, represent water, oxygen, and hydrogen.

The total cell voltage consists of three components:

$$V_{\mathrm{cell}} = V_{\mathrm{rev}} + V_{\mathrm{act}} + V_{\mathrm{ohm}} \tag{14}$$

where, V_{rev} is the reversible voltage (thermodynamic equilibrium); V_{act} is the activation overpotential; V_{ohm}: Ohmic overpotential.

The reversible voltage is determined by thermodynamic equilibrium and is calculated by the Nernst equation (Bard, Faulkner, and White 2022), where the standard reversible voltage is empirically correlated (Roy, Watson, and Infield 2006), reflecting the theoretical minimum decomposition voltage:

$$V_{\mathrm{rev}} = V^0 + \frac{RT}{2F} \ln\left(\frac{p_{\mathrm{H_2}}^{1/2} p_{\mathrm{O_2}}^{1/4}}{a_{\mathrm{H_2O}}}\right) \tag{15}$$

where, V_{rev} is the reversible voltage, affected by temperature; $p_{\mathrm{H_2}}$ and $p_{\mathrm{O_2}}$ are the partial pressures of hydrogen and oxygen, respectively; $a_{\mathrm{H_2O}}$ is the activity of water.

The standard reversible voltage V_{rev}^0 is temperature-dependent:

$$\begin{aligned} V_{\mathrm{rev}}^0 = {} & 1.5148 - 1.5421 \times 10^{-3} T \\ & + 9.23 \times 10^{-5} T \ln T + 9.84 \times 10^{-8} T^2 \end{aligned} \tag{16}$$

The activation overpotential V_{act}, which overcomes the activation energy barrier for the electrochemical reaction at the electrode surface, is caused by electrochemical kinetics and is described by the Butler-Volmer equation (Falcão and Pinto 2020). Typically, the overpotential at the anode is higher than at the cathode:

$$V_{\mathrm{act}} = \frac{RT}{\alpha F} \ln\left(\frac{i}{i_0(1-\Theta)}\right) \tag{17}$$

where, i_0 is the exchange current density; i is the current density; Θ is the electrode surface gas coverage; α is the charge transfer coefficient.

The ohmic overpotential V_{ohm} arises from the internal resistance of the electrolyzer and follows Ohm's law:

$$V_{\mathrm{ohm}} = I \cdot R_{\mathrm{cell}} \tag{18}$$

where R_{cell} is the total resistance of the electrolyzer, including the electrolyte, electrodes, separator, and connecting components.

3 Modelica-Based Power Electronics Simulation

With the widespread integration of high-frequency switching power semiconductor devices in emerging power systems, the system exhibits multi-timescale dynamic behaviors, spanning from nanoseconds (switching transients) to seconds (system-level energy flow and control responses). Traditional Modelica-based power system modeling(Chieh, Panciatici, and Picard 2011) and simulation frameworks are increasingly challenged by the dual bottlenecks of modeling fidelity and computational efficiency(Masoom et al. 2021).

At the modeling level, idealized switch models fail to capture the nonlinear dynamic characteristics of switching transients, as they neglect critical device-level physical phenomena, such as junction capacitance, charge storage, and parasitic inductance. In contrast, physics-based detailed models (e.g., incorporating charge dynamics, temperature dependence, and parasitic networks) offer higher accuracy but are computationally prohibitive for large-scale system simulations.

At the simulation level, conventional variable-step integration algorithms rely on local error estimates to adjust step sizes. However, their gradual step-size reduction mechanisms struggle to effectively resolve both the abrupt transitions at switching events and the slow-varying dynamics of energy storage and thermal inertia components. Moreover, the presence of inherent parasitic parameters—such as stray inductances, device capacitances, and snubber networks—introduces significant numerical stiffness. This stiffness, characteristic of power electronic systems, exacerbates numerical oscillations, increases solver iteration counts, and severely impacts overall simulation speed and stability (Sterner 2009)(Li et al. 2017).

To address these challenges, Suzhou Tongyuan has focused on advancing solution algorithms for power electronic systems, leveraging fixed-step implicit methods as a

DOI 10.3384/ecp218909

foundation. Building on fixed-step implicit Euler and implicit trapezoidal algorithms, discrete forms of these methods have been developed to enhance simulation efficiency and resolve issues specific to discrete simulation, as illustrated in Figure 4.

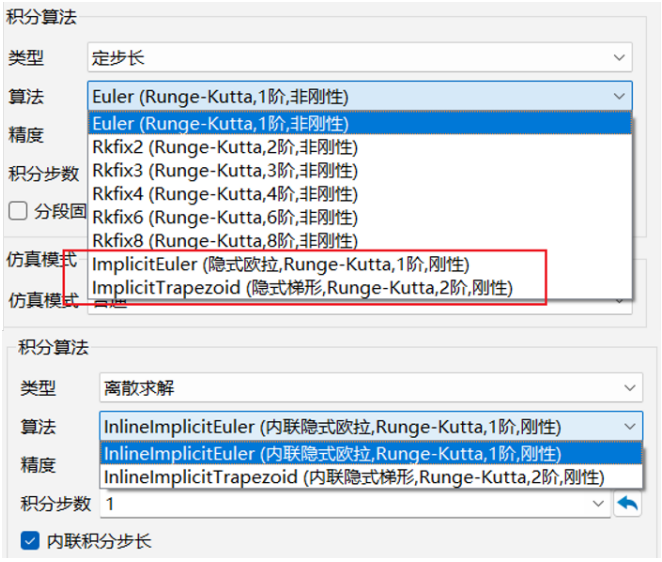


Figure 4. continuous and discrete solving algorithms

Specifically, the implicit Euler method and implicit trapezoidal rule, as core fixed-step implicit algorithms, possess unique properties that underpin their application in power electronic simulations:

The implicit Euler method is based on backward difference approximation, deriving the next-time-step state from the current-time-step state through the formula:

$$\frac{x_{k+1}-x_k}{h} = f(t_{k+1}, x_{k+1}) \tag{19}$$

It features first-order accuracy with strong numerical damping, which can effectively suppress high-frequency oscillations—an advantage when dealing with stiff systems induced by parasitic parameters. While its computational load is relatively low, its accuracy is limited.

The implicit trapezoidal rule combines derivative information from both the current and next time steps, employing trapezoidal integration approximation described by:

$$x_{k+1} = x_k + \frac{h}{2}\left[f(t_k, x_k) + f(t_{k+1}, x_{k+1})\right] \tag{20}$$

It offers second-order accuracy with weak numerical damping, enabling more accurate preservation of the system's inherent dynamic characteristics (e.g., transient oscillations in energy storage components). Though it provides higher accuracy, its computational load per iteration is slightly greater.

Sysplorer constructs a general mathematical model for power electronic system simulation based on the branching method and branching constitutive equations, supporting both continuous and discrete fixed-step algorithms. For discrete fixed-step schemes, the following key design strategies address numerical challenges:

1. Leveraging Piecewise Linear Time-Invariance
   During switching transients, the segmented analytical linearization technique decomposes the nonlinear switching process into multiple linear intervals. Within each interval, system dynamics are described by a linear state equation:

$$\dot{X} = A(t)X + B(t)U \tag{21}$$

   For such linear systems, the implicit Euler method is employed for discretization. Using the backward difference approximation for the time derivative, the discrete update rule is derived as:

$$\frac{X_{k+1}-X_k}{\Delta t} = A_{k+1}X_{k+1} + B_{k+1}U_{k+1} \tag{22}$$

   The core methodology for model discretization incorporates proprietary techniques protected by intellectual property rights and thus will not be elaborated in detail.

   Rearranging terms to solve for X_{k+1}, we obtain:

$$X_{k+1} = (I - \Delta t \cdot A_{k+1})^{-1} \cdot (X_k + \Delta t \cdot B_{k+1}U_{k+1}) \tag{23}$$

   Here, I denotes the identity matrix. This approach transforms state updates into a matrix inversion operation, exploiting the linear structure of the model to avoid nonlinear iterations. When solving the original model directly using the implicit Euler method or other implicit numerical integration algorithms, implicit methods require solving a nonlinear system of equations (the size of which is determined by the number of state variables). Efficiency issues may arise when the scale of this nonlinear system becomes excessively large. In contrast, this discretization method can leverage the inherent linear relationships of the model to further reduce the system equations, thereby improving simulation efficiency. For power electronic systems with inherent numerical stiffness (e.g., induced by parasitic capacitances or inductances), this strategy effectively mitigates numerical oscillations and convergence issues, enhancing the efficiency of fixed-step simulations.

2. **Accurate Switching Event Positioning**
   The discrete algorithms support precise detection of switching events (e.g., device turn-on/turn-off timing). By synchronizing state updates with event triggers, the simulation accuracy of fixed-step algorithms is further improved.

In Figure 5, the average model is based on the state averaging method to establish continuous time-domain

equations, and focuses on the low-frequency dynamic characteristics of the converter by filtering out the switching frequency ripples, which significantly reduces the order of the model to meet the demand for real-time simulation. For the high-frequency characterization of nanosecond switching transients, the detailed switching transient model adopts the segmented analytical linearization technique, which decomposes the switching process into multiple linearization intervals, and accurately characterizes the nonlinear behaviour of the device during the on/off process through the dynamic switching between analytical solving and event triggering. The three types of models are decoupled by time scale and interface variables to form a multi-level simulation framework from microsecond transient to second system response. Switching models are suitable for short-term high-precision analysis, while the averaging model is suitable for long-term or multi-component system simulations (Reindl et al. 2023).

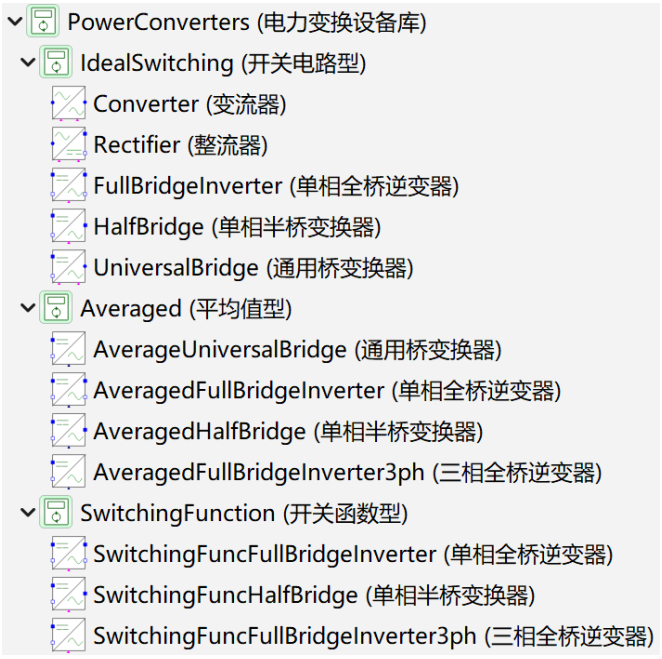


Figure 5. Power Electronic Converter Model

Based on the MWORKS.Sysplorer electromagnetic transient solving algorithm, the DC grid-connected model of the PV system in Figure 6 is calculated, and the simulation time is set to 0.5 s, the step size of the detailed switching model is 1×10^{-5} seconds, and the step size of the average value model is 5×10^{-5} seconds. The simulation results are consistent in terms of key performance variables, and a comparison of simulation times is presented in Table 1. The discrete algorithm incurs longer simulation time than the continuous counterpart. This stems from the model's frequent nonlinear switching events: although segmented linearization decomposes switching transients into linear intervals, repeated matrix inversions across these intervals accumulate computational overhead.

Considering the application scenario of off-grid energy islands, which integrate multiple distributed energy

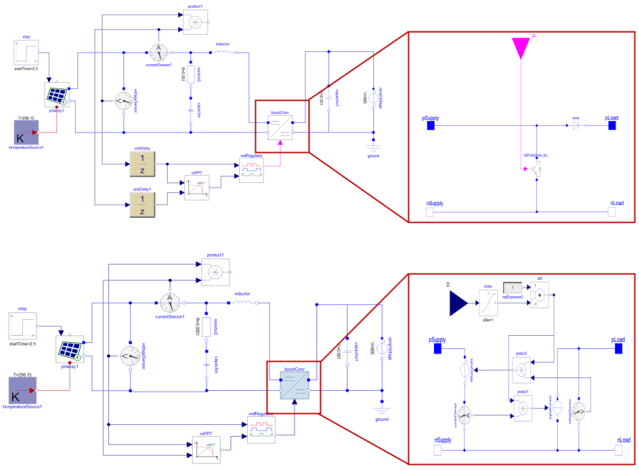


Figure 6. Test PV system for algorithm validation

sources, energy storage systems, and diverse loads with strong system coupling, long-term simulation is often required to analyze energy scheduling strategies, operational stability, and dynamic responses under fluctuating renewable energy inputs. In addition, considering issues such as system coupling and simulation rate, the average value model can meet the requirements of simulation accuracy. Therefore, the converter models in this paper all adopt the average value model.

4 System Modeling and Simulation

4.1 System Configuration

This chapter models an islanded DC microgrid integrating a photovoltaic (PV) system, wind power generation, electrochemical energy storage and an alkaline electrolyzer for hydrogen production. The system employs maximum power point tracking (MPPT) control to maximize renewable energy harvesting from PV and wind power, while a dual-loop adaptive control is implemented in the energy storage system to keep the DC bus voltage stable and smoothen the hydrogen production process. The alkaline electrolyzer is operated under constant power control to ensure optimal hydrogen production efficiency.

Figure 7 illustrates the operating flow of this renewable energy-hydrogen hybrid system. When the renewable energy generation exceeds the power demand of the electrolyzer, the excess energy is absorbed by the energy storage device. Conversely, when the load demand exceeds the renewable energy generation, the energy storage device discharges to compensate for the power deficit, thus maintaining the system in dynamic balance.

Component-level modeling takes a systematic approach: after initial independent validation of the wind, photovoltaic, and hydrogen generation subsystems, a full system simulation is performed with or without integrated energy storage. All simulations were performed using the discrete InlineImplicitEuler algorithm with a fixed time step of 5×10^{-5} seconds.

DOI
10.3384/ecp218909

Table 1. Comparison of Simulation Time for Different Converter Types and Algorithms

Converter Type	Simulation Time	
	Continuous Algorithm	Discrete Algorithm
Detailed Switching (step size: 1×10^{-5})	0.477	3.223
Average Model (step size: 5×10^{-5})	0.058	0.362

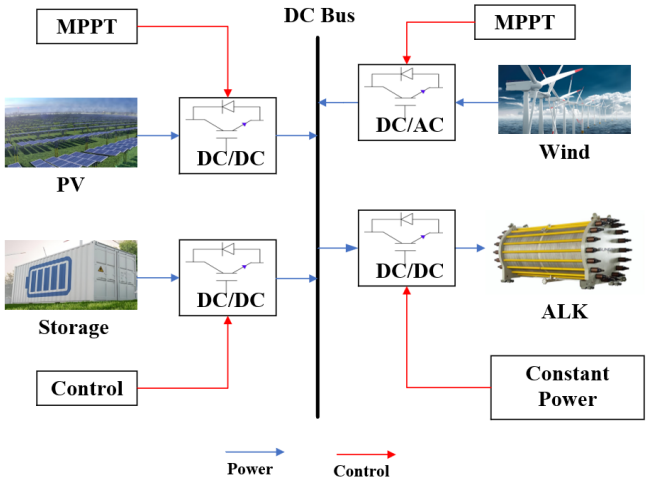


Figure 7. Schematic diagram of the structure of a Wind-PV-Storage Integrated Hydrogen Production system

4.2 Subsystem Validation

4.2.1 Wind Power System

According to Equation 1 – Equation 6 and the control method to establish the wind electronic system, as shown in Figure 8, the wind electronic system consists of wind speed model, blades, permanent magnet synchronous generator, averaged universal bridge converter, MPPT controller, voltage modulation model, machine-side controller and so on, and the main parameters of the wind power system are shown in Table 2.

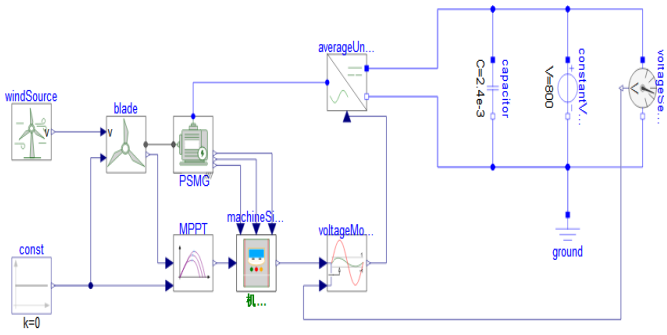


Figure 8. Model diagram of wind electronic system

The system employs a combined wind speed simulation to model the variation of wind speed over time. The wind speed starts from an initial value of 3 m/s and gradually increases to 9 m/s, taking into account both gradual wind changes and random wind fluctuations. The simulation duration is set to 500 seconds.

Table 2. Wind Turbine Parameters

Parameter	Value/Unit
Rated Power P	2 MW
Rated Speed ω	1.93 rad/s
Blade Radius R	40 m
Blade Pitch Angle β	0 °
Air Density ρ	1.293 kg/m^3
Wind Energy Calculation Coefficients	0.70, 150, 0.58, 0.002, 2.12, 13, -18, -0.02, 0.003

The results of the blade wind energy and mechanical power calculations are shown in Figure 9. It can be observed that the wind turbine operates in the constant speed region of the power tracking curve. The pitch angle β is maintained at 0 degrees, indicating normal turbine operation. However, the mechanical power has not reached the rated value, implying that a significant portion of the wind energy is not fully captured. The pitch angle remains unchanged, and the power output is primarily determined by the wind speed and the aerodynamic performance of the blades.

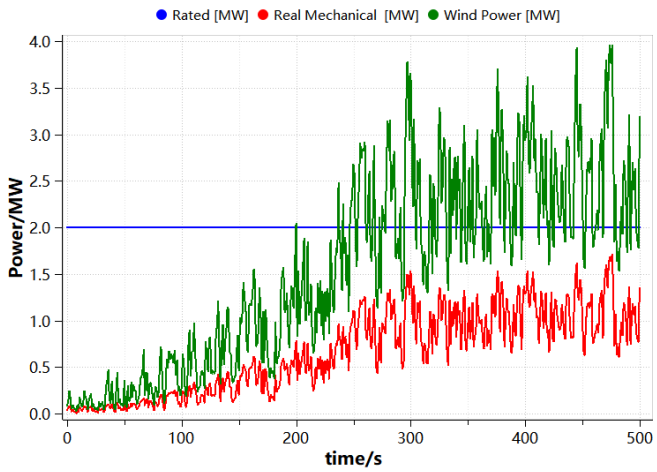


Figure 9. Wind and mechanical power curves

As shown in Figure 10, the tip-speed ratio and power coefficient curve indicate that the system has successfully

implemented Maximum Power Point Tracking (MPPT). The power coefficient has reached its maximum value, and the actual tip-speed ratio of the wind turbine closely approximates the optimal tip-speed ratio corresponding to the maximum power coefficient. The deviation of the tip-speed ratio from its optimal value is attributed to the fact that wind speed is not directly measured. Consequently, the reference signal for the controller (i.e., the rotational speed of the wind turbine) cannot directly respond to wind speed variations.

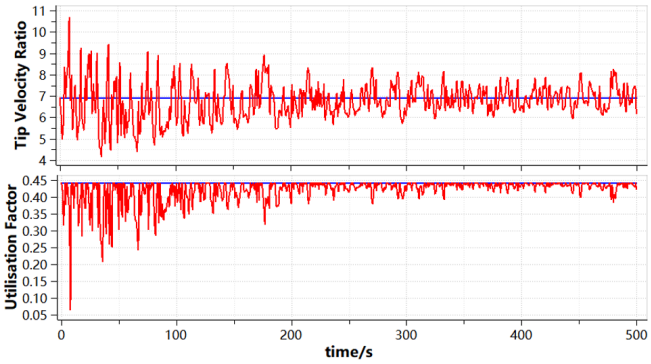


Figure 10. Leaf tip speed ratio curve and wind energy utilisation curve

4.2.2 Photovoltaic System

The photovoltaic (PV) system is established based on Equation 7−Equation 8 and the MPPT control method, as shown in Figure 11. The system comprises environmental models, PV panels, a Boost converter with an MPPT controller, etc. The Boost converter employs an average model and steps up the voltage based on the duty cycle signal output from the voltage regulator.

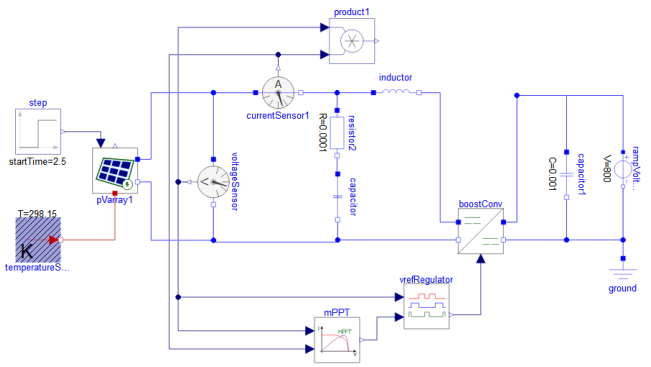


Figure 11. Photovoltaic power generation system

The solar irradiance is set to 1,000 W/m² and steps up to 1,200 W/m² at 2.5 s. The temperature is maintained at 25 °C. Each PV panel has a rated power of 213.15 W, with 10 panels connected in series and 47 panels connected in parallel. The main parameters of the single PV panel are listed in Table 3.

The photovoltaic (PV) MPPT control system can track environmental changes and adjust the output voltage of the PV array to maximize the output power.

Table 3. Single PV Panel Parameters

Parameter	Value/Unit
Maximum Power P_{max}	213.15 W
Number of Cells N_{cell}	60
Open-Circuit Voltage V_{oc_user}	36.3 V
Operating Voltage V_{mp_user}	29 V
Short-Circuit Current I_{sc_user}	7.84 A
Maximum Power Point Current I_{L_ref}	-0.36099 A
Voltage Temperature Coefficient β_{v_user}	0.102 V/K
Current Temperature Coefficient β_{i_user}	0.102 A/K
Series Resistance R_{s0}	0.39381 Ω
Shunt Resistance R_{sh0}	313.0553 Ω

As shown in Figure 12, the output power of the PV array varies with solar irradiance. The simulation results indicate that when the solar irradiance increases from 1,000 W/m² to 1,200 W/m², the output power of the PV panel, controlled by the MPPT controller, increases from 102,169 W to 121,362 W. This demonstrates that the MPPT controller effectively tracks environmental changes and stabilizes the output power at the maximum power point.

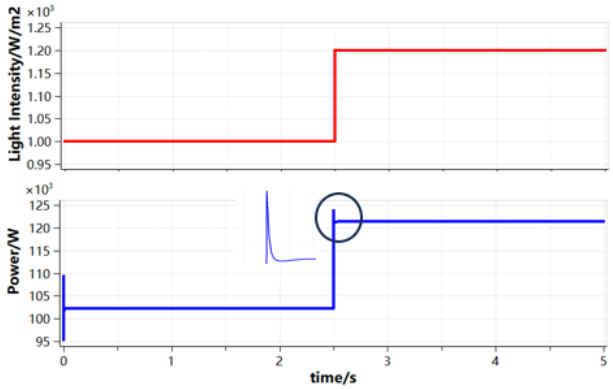


Figure 12. Variation curve of PV array output power with solar irradiance

4.3 Alkaline Electrolyzer

The alkaline electrolyzer model is established based on Equation 10−Equation 18. Considering the high reusability of the electrolyzer model, separate modeling is performed for the cathode, anode, Anolyte, Catholyte and separator within the alkaline electrolyzer. Based on the principles of alkaline electrolysis and considering current flow and material movement, the components are connected via electrolyzer ports and fluid interfaces.

The internal pressure of the electrolyzer is set to 1 bar, with the temperature maintained at 353 K, and temperature variations are considered negligible. A fixed voltage of 2 V is applied to a single electrolyzer cell for electrol-

ysis, with a current density of 1470 A/m^2. The potential differences between the components are listed in Table 4.

Table 4. Potential Differences Between Electrolyzer Components

Component	Potential Difference
Anode	0.57V
Anolyte	0.25V
Separator	0.02V
Catholyte	0.17V
Cathode	0.99V

The hydrogen and oxygen production rates of the alkaline electrolyzer are depicted in Figure 13. The production rates stabilize after 3 seconds. The balance time is influenced by the initial volume of the solution in the electrolyzer and the flow rate of the solution entering the electrolyzer. The hydrogen production rate per cell is 0.1678 Nm^3/h, while the oxygen production rate is 0.0838 Nm^3/h.

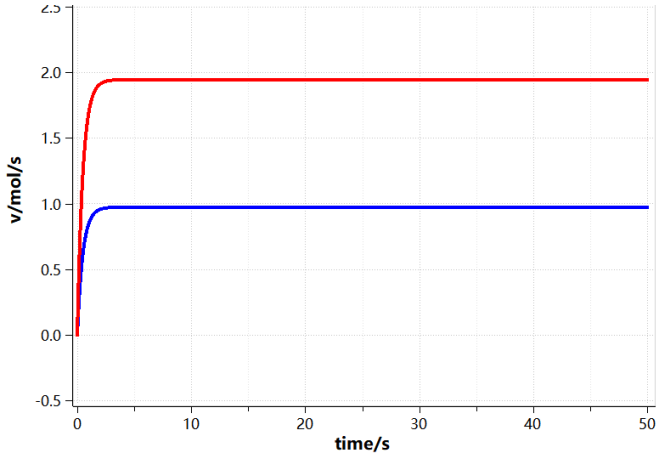


Figure 13. Hydrogen and oxygen production rates

4.4 Wind Power Storage Hybrid System Simulation

To validate the effectiveness of the hybrid wind–photovoltaic–storage hydrogen production system model and its control strategy for an offshore energy island scenario, and to analyze the impact of the wind and PV maximum power point tracking (MPPT) control strategy and the energy storage unit on system reliability and stability, simulations of the wind–PV hydrogen production system were conducted in two scenarios: with and without an energy storage unit.

4.5 Wind-Photovoltaic Hydrogen Production System Simulation

Based on the aforementioned wind power, photovoltaic, and alkaline electrolyzer models and their corresponding control strategies, a hybrid wind–PV hydrogen production system was established. The system comprises 100 series-connected alkaline electrolyzers, with a total rated power of 100 kW. The wind power and PV parameters were appropriately adjusted to match the rated power with the electrolyzer's hydrogen production load. Wind and PV power, as the source side, exhibit significant complementary characteristics in terms of temporal and spatial properties. Variations in wind speed and solar irradiance were simulated to verify the performance of the hybrid wind–PV power generation system under different operating conditions. The system configuration is illustrated in Figure 14.

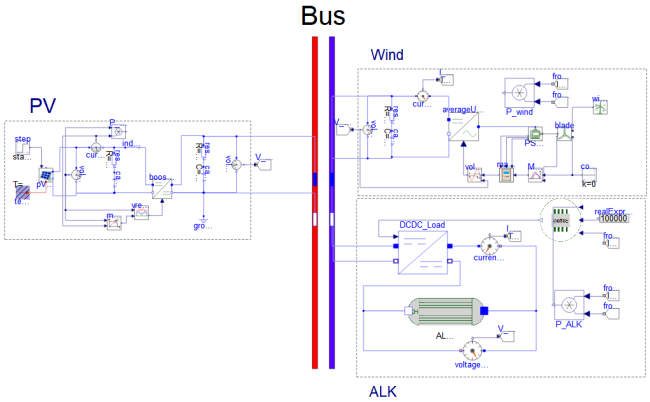


Figure 14. Hybrid Wind-PV Hydrogen Production System Topology

By adjusting the input of solar irradiance and wind speed, the initial solar irradiance was set to 800 W/m^2 and increased to 1000 W/m^2 at 100 s. The wind speed profile incorporated gusts, gradual changes, and random fluctuations.

As shown in Figure 15, the PV MPPT controller rapidly stabilizes the power output at the maximum power point. In the early simulation phase (first 100 s), the alkaline electrolyzers operates at low power due to insufficient wind–PV power generation. After the solar irradiance increase at 100 s, the electrolyzer's power gradually approaches 100 kW. However, as the wind power output further increases during the maximum power point tracking phase, the electrolyzer's power exceeds its rated capacity, posing potential safety concerns.

The fluctuations in wind and PV power generation significantly impact the safe operation of the alkaline electrolyzer. Due to the instability of wind and PV power, the input power to the electrolyzer may experience significant fluctuations, making it challenging to configure the electrolyzer capacity for specific scenarios.

4.6 Wind-PV-Storage Hydrogen Production System Simulation

To address the issues identified in the previous section, an energy storage system is coupled with the hybrid wind–wind-photovoltaic hydrogen production system. The energy storage system adaptively balances the power output

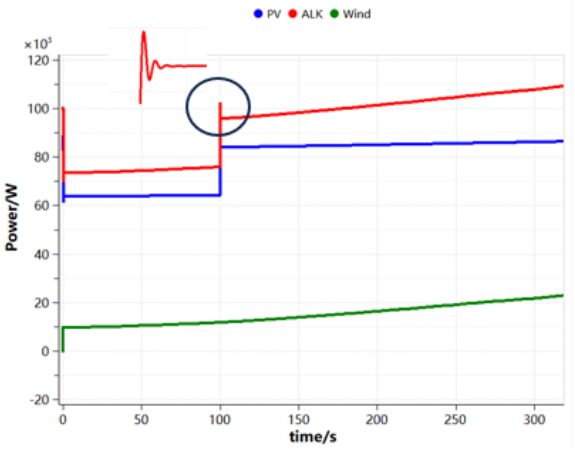


Figure 15. Wind-PV and Electrolyzer Power

from the wind and photovoltaic sources, suppressing fluctuations and stabilizing the DC bus voltage. This ensures that the electrolyzer operates stably at its rated power. The simulation model of the system is shown in Figure 16. The energy storage system is configured with a capacity of 38.4 kWh and an initial state of charge (SOC) of 0.5, and the simulation time is set to 1000 seconds.

The adaptive coordinated control of the energy storage system is based on the DC bus voltage as the power control signal, employing dual-loop voltage and current control. Photovoltaic and wind power, as renewable energy sources, have priority in power output. When the output power is higher or lower than the rated power of the alkaline electrolyzer load, the energy storage system achieves charging and discharging functions through an adaptive coordinated control method. This maintains the balance of the DC bus voltage and ensures the stable and safe operation of the electrolyzer.

In a 1000-second simulation period, the irradiance random perturbation condition was set (irradiance range: 500 - 2000 W/m², step interval: 100 seconds). Figure 17 shows that when an energy storage unit system adaptive coordinated control is activated to maintain the bus voltage at 800 V, small fluctuations in bus voltage occur with each irradiance fluctuation. The magnitude of these voltage fluctuations is related to the extent of irradiance changes. Despite these small fluctuations, the bus voltage remains relatively stable.

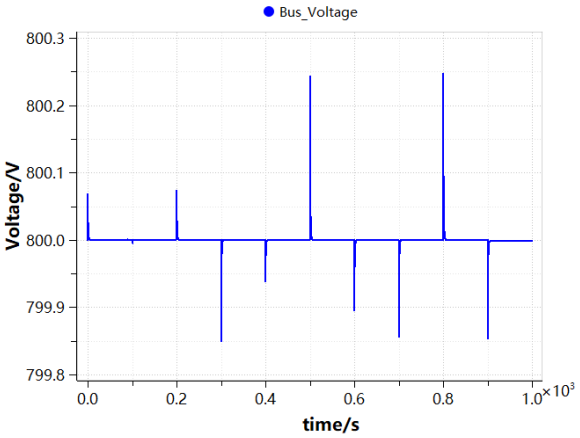


Figure 17. DC bus Voltage

The stable bus voltage enables the alkaline electrolyzer's constant-power controller to operate stably and continuously. The fluctuations in wind and solar power are both smoothed out by the energy storage system, as shown in Figure 18. Within the simulation period, the wind power output increases slowly but remains relatively low, so the system is mainly powered by solar energy. When the solar power output changes abruptly, the en-

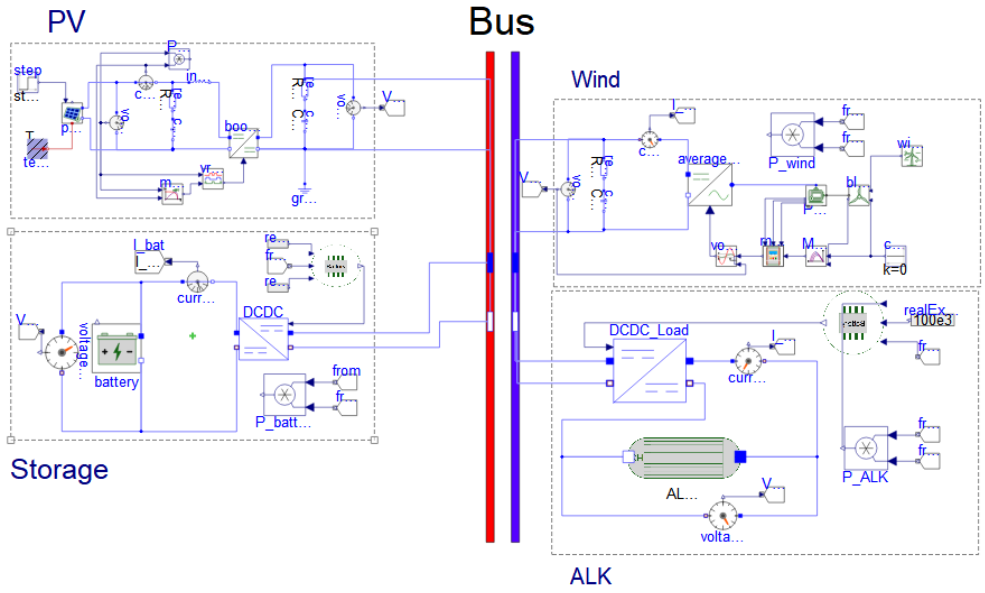


Figure 16. Hybrid Wind-PV-Storage Hydrogen Production System Topology

DOI
10.3384/ecp218909

ergy storage system responds by stabilizing it. The energy storage system's charge or discharge state is determined by the power difference between renewable outputs and the electrolyzer's 100 kW set power. The corresponding state-of-charge (SOC) variation is presented in Figure 19

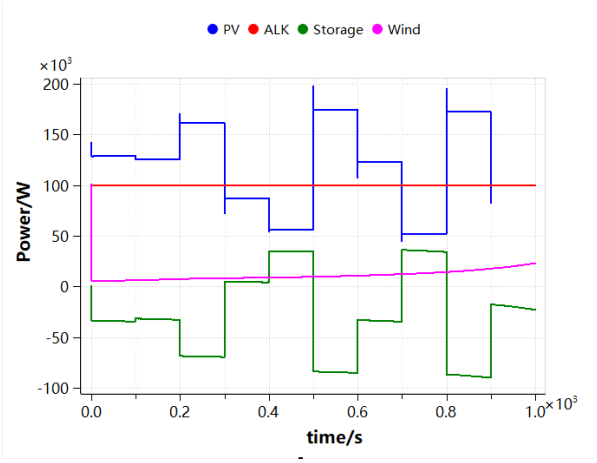


Figure 18. Wind-PV-Storage and Electrolyzer Power

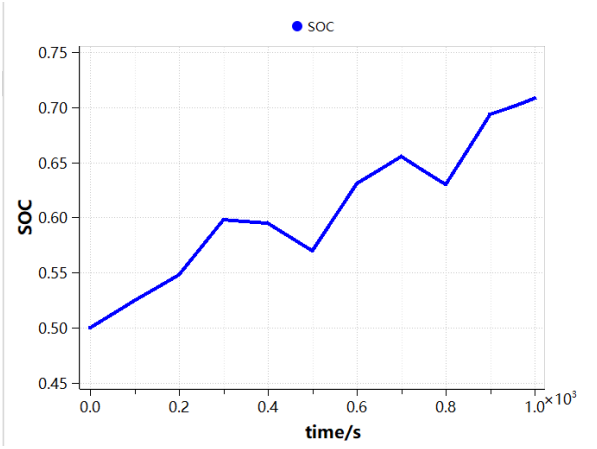


Figure 19. Energy storage SOC

The hydrogen and oxygen production rates of 100 series-connected alkaline electrolyzers with a rated power of 100 kW are shown in Figure 20. Under a current density of 1916 A/m^2 and a cell voltage of 2.087 V, the hydrogen and oxygen production rates are 20 Nm^3/h and 10 Nm^3/h, respectively.

5 Conclusions and Outlook

This paper investigates the dynamic characteristics and control strategies of an offshore energy island system in a hybrid wind–photovoltaic–energy storage hydrogen production scenario through detailed modeling and simulation based on Modelica. The key findings are summarized as follows:

Three modeling methods with different granularities are proposed for the multi-time-scale dynamic response problem of power electronic systems, and continuous and

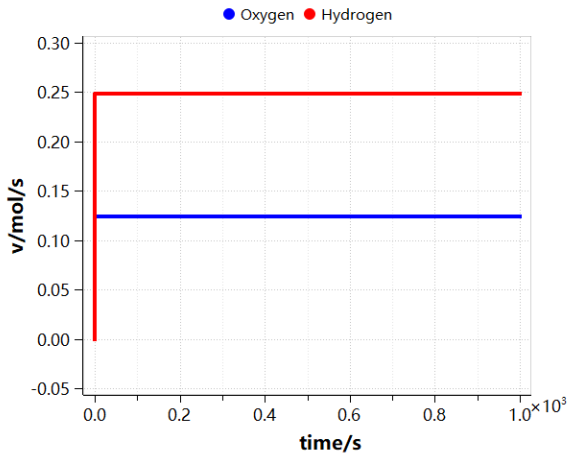


Figure 20. Hydrogen and oxygen production rates

discrete solution algorithms are developed. These approaches significantly enhance simulation efficiency and accuracy.

Subsystem verification shows that the wind power system achieves effective wind energy capture through maximum power point tracking (MPPT) control, and the photovoltaic system.

The energy storage system effectively mitigates the volatility of wind and photovoltaic power generation, maintains DC bus voltage stability, and ensures the safe operation of the electrolyzer.

The research results provide a high-precision simulation support for system parameter optimization and control strategy verification of offshore energy islands, offering significant reference value for practical project implementation. Future work could further explore the system's dynamic behavior across multiple time scales and complex electromagnetic transient processes to enhance the overall performance of offshore energy islands. Additionally, the proposed framework can be extended to include detailed models for hydrogen storage and ammonia/methanol production, bridging the gap between power systems and chemical processes.

References

Arsad, A Zaini et al. (2022). "Hydrogen energy storage integrated hybrid renewable energy systems: A review analysis for future research directions". In: *International Journal of Hydrogen Energy* 47.39, pp. 17285–17312.

Bard, Allen J, Larry R Faulkner, and Henry S White (2022). *Electrochemical methods: fundamentals and applications*. John Wiley & Sons.

Bilgili, Mehmet, Abdulkadir Yasar, and Erdogan Simsek (2011). "Offshore wind power development in Europe and its comparison with onshore counterpart". In: *Renewable and Sustainable Energy Reviews* 15.2, pp. 905–915. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2010.11.006. URL: https://www.sciencedirect.com/science/article/pii/S1364032110003758.

Chieh, Angela S., Patrick Panciatici, and Jérôme Picard (2011). "Power system modeling in Modelica for time-domain simulation". In: pp. 1–8. DOI: 10.1109/PTC.2011.6019400.

Falcão, DS and AMFR Pinto (2020). "A review on PEM electrolyzer modelling: Guidelines for beginners". In: *Journal of cleaner production* 261, p. 121184.

Fang, Ruiming and Yin Liang (2019). "Control strategy of electrolyzer in a wind-hydrogen system considering the constraints of switching times". In: *International journal of hydrogen energy* 44.46, pp. 25104–25111.

Igbinenikaro, Osayi Philip, Oladipo Olugbenga Adekoya, and Emmanuel Augustine Etukudoh (2024). "Conceptualizing sustainable offshore operations: Integration of renewable energy systems". In: *International Journal of Frontiers in Science and Technology Research* 6.02, pp. 031–043.

Kundur, Prabha (2007). "Power system stability". In: *Power system stability and control* 10.1, pp. 7–1.

Li, Boyang et al. (2017). "A novel simulation method for power electronics: Discrete state event driven method". In: *CES Transactions on Electrical Machines and Systems* 1.3, pp. 273–282.

Masoom, A. et al. (2021). "Modelica-based simulation of electromagnetic transients using Dynao: Current status and perspectives". In: *Electric Power Systems Research* 197, p. 107340. ISSN: 0378-7796. DOI: https://doi.org/10.1016/j.epsr.2021.107340. URL: https://www.sciencedirect.com/science/article/pii/S0378779621003217.

Pyschik, Jan F et al. (2025). "eCherry: A Modelica Library for Modular Dynamic Modelling of Electrochemical Reactors". In: *Electrochemical Science Advances*, e202400030.

Reindl, Andrea et al. (2023). "Switching and Averaging Models of a Bidirectional, Half-Bridge Based DC-DC converter with Load Distribution". In: pp. 683–692.

Roy, Amitava, Simon Watson, and David Infield (2006). "Comparison of electrical energy efficiency of atmospheric and high-pressure electrolysers". In: *International Journal of Hydrogen Energy* 31.14, pp. 1964–1979.

Salmador Gonzalez, Juan (2016). "Frequency support by variable speed wind turbines in the power system of Gran Canaria". In.

Shen, Xiaojun et al. (2018). "Experimental study on the external electrical thermal and dynamic power characteristics of alkaline water electrolyzer". In: *International journal of energy research* 42.10, pp. 3244–3257.

Sterner, Michael (2009). *Bioenergy and Renewable Power Methane in Integrated 100% Renewable Energy Systems. Limiting Global Warming by Transforming Energy Systems: Limiting Global Warming by Transforming Energy Systems*. Vol. 14. kassel university press GmbH.

Tiejiang, Yuan et al. (2022). "The day-ahead output plan of hydrogen production system considering the start-stop characteristics of electrolytic cell". In: *Electric power* 55.1, p. 101.

Ulleberg, Øystein (2003). "Modeling of advanced alkaline electrolyzers: a system simulation approach". In: *International Journal of Hydrogen Energy* 28.1, pp. 21–33. ISSN: 0360-3199. DOI: https://doi.org/10.1016/S0360-3199(02)00033-2. URL: https://www.sciencedirect.com/science/article/pii/S0360319902000332.

Webster, John and Carsten Bode (2019). "Implementation of a non-discretized multiphysics PEM electrolyzer model in modelica". In: *13th International Modelica Conference 2019*. Linköping University Electronic Press, pp. 833–840.

Xiaojun, Shen, Nie Congying, and Lü Hong (2021). "Coordination control strategy of wind power-hydrogen alkaline electrolyzer bank considering electrothermal characteristics". In: *Transactions of China Electrotechnical Society* 36.3, pp. 463–472.

Xing, Xuetao et al. (2018). "Modeling and operation of the power-to-gas system for renewables integration: a review". In: *CSEE Journal of Power and Energy Systems* 4.2, pp. 168–178.

From Simulation to Reality: Deployment of Reinforcement Learning-Based Neural Network Controllers Trained with Modelica Models

Joshua Brun[1] Thomas Sergi[1] Sylvan Mutter[1] Tim Arnold[1] Ulf Christian Müller[1]

[1]IME, Lucerne School of Engineering and Architecture, Switzerland,
{joshua.brun, ulfchristian.mueller}@hslu.ch

Abstract

To address limitations of traditional control methods in complex systems, reinforcement learning (RL) combined with simulation models provides an efficient approach for controller development. In this work, a complete toolchain for developing and deploying RL-based neural network controllers using Modelica system models is presented. Serving as a showcase, a real-world double-inverted pendulum is constructed. The system was modeled in Modelica by combining physics-based and data-driven modeling approaches for efficient development. Coupling simulation with an RL environment is achieved through Functional Mock-up Interface (FMI) standard. Successful training and sim-to-real transfer are demonstrated on a single-pendulum setup, validating the approach for extension to a double-inverted pendulum. This paper provides a reproducible and extensible framework which is well-suited for educational purposes and highlights strengths of Modelica in combination with machine learning approaches.

Keywords: data-driven system modeling, reinforcement learning, sim-to-real transfer, FMI, Modelica, double-inverted pendulum

1 Introduction

Combining reinforcement learning (RL) with system models is a promising approach, as it enables generation of training data using simulations of a system. In this work, a neural network control system is trained using RL coupled with a Modelica system model. After training, the controller is successfully applied to a real system, showcasing challenges and feasibility of this toolchain.

1.1 Motivation

Development of advanced controllers for complex systems requires deep expertise and can even become infeasible with traditional control system methods. This is where RL can fill a gap by learning a control strategy on its own, using a data-driven approach. However, real-world data is limited and costly, which is why generating required training data with a virtual system model is often the only possible path. By combining RL with physics-based simulation models, such as those typically built using Modelica, it becomes feasible to generate efficient training data while ensuring that learned controllers respect underlying system dynamics.

Methods combining RL with physics-based simulation models and sim-to-real transfer of neural network control systems have already been successfully applied in recent studies. Lukianykhin and Bogodorova (2019) introduced a toolbox for integration of Modelica-based models in RL via FMI ('Functional Mock-up Interface', 2014) standard. Heuermann et al. (2023) demonstrated that machine learning surrogate models can accelerate Modelica-based simulations by replacing nonlinear algebraic loops, thereby enabling more efficient training of RL controllers. Lee, Ju and Lee (2025) showcased the use of RL-based neural network control systems and sim-to-real transfer for transition control in a double-inverted pendulum on a cart-pole system, using a model based on Euler-Lagrange differential equations. These successful applications provide a strong foundation for further research that combines these approaches, as pursued in this work. The importance of such methods is expected to grow as systems get more complex in Industry 4.0 and beyond. While alternative approaches, such as *Model Predictive Control* (MPC), which rely on online simulation or optimization, also offer significant benefits, they are often less practical, especially for time-critical applications, due to their substantial computational demands (Morcego et al., 2023). A combination of RL and physics-based simulation models has already been addressed in research projects conducted in collaboration with industry. Among them is *IntelliBake* (INNOSUISSE, 2025), which investigates the application of RL-based neural network control systems for baking processes. Building upon these related efforts, this work contributes to this growing field by providing a demonstrator system and a reusable toolchain for RL-based controller development and deployment.

1.2 Goal

This work demonstrates the development of a neural-network control system trained entirely through RL on a Modelica model. Trained controllers are subsequently transferred to a real-world system, expected to perform a desired control task. This contribution establishes a working toolchain that serves both as proof of concept and as a foundation for future industrial and educational applications. In doing so, it aims to highlight typical challenges and potential solutions associated with following steps:

- System modeling with Modelica for state calculation during RL process.
- Training a neural network control system using RL and Modelica-based simulations.
- Sim-to-Real transfer of a trained neural network control system.

2 Showcase Selection and Setup

The chosen showcase system is a double-inverted pendulum. A motor drives the first pendulum, while the second pendulum is attached via a freely rotating plain bearing. The control task is to swing both pendulums from their initial downward-hanging position to an upright position. This involves two control phases: (1) swinging both pendulums upward by actuating the first joint and (2) stabilizing them in an upright position.



Figure 1. 3D model of the double-inverted pendulum.

This system was selected as a suitable showcase for following reasons:

- *Simple and compact construction:* This system is lightweight, portable, and space-efficient, making it suitable for use in various locations.
- *Demonstrative system for educational purposes:* A double-inverted pendulum allows for direct visual interpretation of control performance, making success and failure immediately apparent.
- *Benchmark character:* A double-inverted pendulum is a well-established benchmark in control theory, enabling direct comparison with related research.
- *Stepwise complexity:* A double-inverted pendulum can be easily simplified to a single pendulum, significantly reducing complexity. This allows initial focus on validation of the toolchain before addressing more advanced control challenges.
- *Challenging sim-to-real conditions:* The System's sensitivity to model inaccuracies (e.g., calibration errors) and a required high control frequency are creating challenges for a sim-to-real transfer, making it ideal for studying transferability.

2.1 System Construction

This mechanical setup features 3D-printed housing and pendulums, a metal flange on the motor mount, and a plain bearing between two pendulums. The assembly is clamped to a table. Physical parameters listed in Table 1 were derived from a computer-aided design (CAD) geometry.

Table 1. Physical parameters of the showcase system

Parameter	*Pendulum 1*	*Pendulum 2*
Mass	0.099 kg	0.061 kg
Length	0.19 m	0.19 m
COM distance from joint center	0.111 m	0.076 m
Moment of inertia	1.9e-3 kg·m^2	7.3e-4 kg·m^2

A brushless DC motor (GB54-2 Gimbal Type 3-6S UAV Drone Motor KV26 by T-Motors) serves as actuator and is controlled via a motor driver (SimpleFOCShield v2.0.4). A hollow motor shaft design allows for internal cable routing, and its permanent magnet rotor enables unrestricted rotation of pendulums. Two 14-bit absolute capacitive encoders (AMT222B by CUI Devices) are used to measure joint angles of both pendulums. These encoders provide direct high-resolution angular measurements for sensor feedback in a control loop. A neural network control system computes a motor voltage in real-time based on sensor inputs. These control signals are transmitted via serial communication to an Arduino Uno R3, which interfaces with electronic components. In this current configuration, the maximum achievable control loop frequency is approximately 250 Hz.

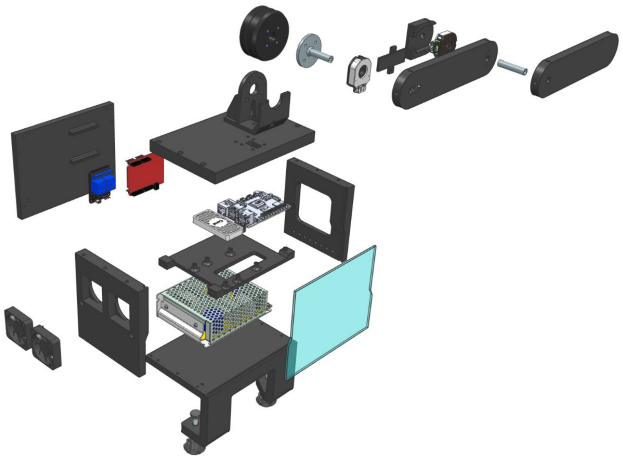


Figure 2. Exploded view of double-inverted pendulum.

DOI 10.3384/ecp218921

3 System Modeling: Combining Data-based and Physics-based Modeling in Modelica

Successful RL training of a neural-network controller requires an adequately accurate system model. This section outlines the modeling approach for a double-inverted pendulum. Due to the nonlinearity of its governing equations, a double-inverted pendulum exhibits chaotic behavior, making modeling and controlling particularly challenging when using analytical or traditional numerical methods.

This model primarily utilizes components from the `Mechanics` package within the `Modelica Standard Library` (Modelica Association, 2020). An exception is the brushless DC (BLDC) electric motor, which is modeled using a data-driven approach. This method will be described in section 3.1.

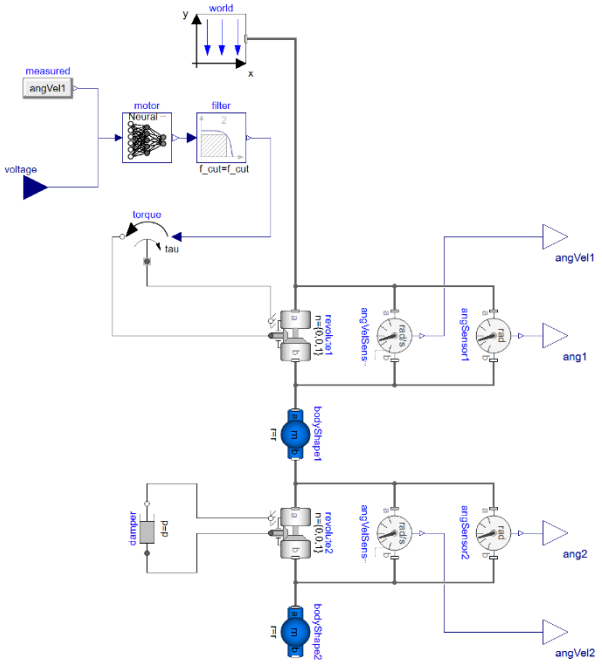


Figure 3. Graphical representation of the Modelica model for the double-inverted pendulum system.

Figure 3 shows a graphical representation of the system model used to train the neural network controller. The model uses two body shape components to precisely model unevenly distributed moments of inertia using a simple method. The pendulums are connected to each other by a joint with a single degree of freedom for rotational motion. Torque is calculated from a data-driven motor model and applied to the first pendulum. For both joints, bearing friction is considered. Friction torque as a function of angular velocity was derived through model calibration using experimental data and mathematical optimization with a Python library called Pygmo. (Biscani and Izzo, 2020). The optimization objective was to minimize normalized root mean squared error (NRMSE) between measurement data and simulation results.

This model has six boundary connectors to interface with an RL environment: two inputs (motor voltage and angular velocity feedback) and four outputs (joint angles and angular velocities of both pendulums). The voltage input is controlled by an RL agent, while angular velocity is required by the data-based motor model to compute torque (see following section). Output signals are used as observations and for reward calculation during training.

3.1 Data-based Methods for Efficient Modeling

A BLDC motor, including its field-oriented control (FOC), requires a specialized modeling approach. Due to its physical complexity, deriving an accurate model from first principles is both challenging and time-consuming. To effectively capture the motor's real-world behavior, a data-driven model was developed. It consists of a feedforward neural network (FNN) trained on measurement data using supervised learning. Available data points primarily cover the motor's static behavior, meaning that transient effects are not fully represented in the training set. To approximate dynamic effects, a Bessel function is included to capture transient torque responses. This approach was deliberately chosen to start with a simple surrogate model. This surrogate model can later be refined, for example by replacing an FNN with a recurrent neural network (RNN), which can leverage time-series information to improve predictions during acceleration and deceleration phases or sudden changes in setpoint speed.

Torque output of a BLDC motor primarily depends on electric current and angular velocity. Since the setup does not permit direct electric current measurement, the motor is controlled using pure voltage input.

For DC motors in general, torque M is proportional to current I, scaled by a torque constant k_m.

$$M = k_m \cdot I \tag{1}$$

Current I can be approximated as proportional to applied voltage U, assuming a constant internal resistance R.

$$I = \frac{U - EMF}{R} \quad \rightarrow \quad I \propto U \tag{2}$$

Therefore, the model assumes that torque M is proportional to input voltage U.

$$U \propto M \tag{3}$$

These assumptions hold for angular velocities near zero, where back- EMF is negligible. Given the motor's application in pendulum control, this simplification is appropriate.

Following on these assumptions, torque values as a function of velocity and input voltage were experimentally determined to develop a motor model. Measurement data was collected for several operational states of a specific BLDC motor. First, no-load operation was recorded across a range of input voltages to establish a zero-torque line in a velocity–voltage plane, with friction effects neglected. Second, the motor's maximum output torque was measured at various input voltages near zero angular velocity. Additionally, data points from runs involving a reversal of rotation direction were recorded along diagonal regions of the plane. Collected measurements are visualized in Figure 4, which displays the motor output torque as a function of angular velocity and input voltage. Measured data points are shown as black dots, while the 3D mesh surface represents the output of the trained feedforward neural network (FNN).

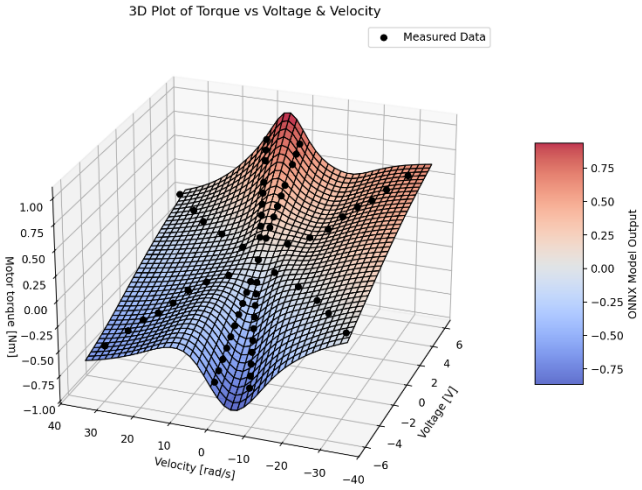


Figure 4. Output visualization of the data-driven motor model.

A supervised learning algorithm was applied to train an FNN. The high-level API `Keras` (Watson et al., 2024), built on top of `TensorFlow` (TensorFlow Developers, 2025), provides a flexible framework for deep learning and enables efficient training of this motor model.

The measured dataset was split into training and test subsets using an 80/20 ratio. This approach allows model performance to be validated on previously unseen data, which is essential for detecting potential overfitting. A Bayesian hyperparameter optimization strategy was employed to efficiently determine a suitable set of hyperparameters (listed in Table 2), after which the FNN was trained for 120 epochs. The resulting model achieved a mean absolute error (MAE) of less than 0.01 Nm on the test dataset. Corresponding mean squared error (MSE) was 0.000095 Nm^2, with a maximum absolute error of 0.038 Nm.

Table 2. Hyperparameters used to train the data-driven motor model.

Parameter	*Value*
Dense Layers	4
Neurons per Layer	80
Learning Rate	0.001
Activation Function	GELU
Optimizer	Adam
Loss	Mean Absolute Error

3.2 Integration of data-driven surrogate models in Modelica

To integrate a data-driven surrogate model of the motor into a Modelica model, the MoONNX (Sergi and Brun, 2024) library has been used. This self-developed library enables execution of Open Neural Network Exchange ('ONNX Project', 2024) models within Modelica environments and is available on GitHub. Currently, it supports Windows 10 (64-bit) and newer Windows operating systems. During simulation, the ONNX runtime library ('ONNX Runtime', 2021), which performs neural network inference, is called with an external function. Resulting outputs can then be used for further computations within the Modelica model.

4 Training Neural Network Controllers using RL and Modelica Simulations

When a system model is available, various methods can be used to develop an optimal control strategy. Neural networks are particularly well suited for implementing complex control policies while keeping real-time computational demands manageable. Discovery of an optimal control policy and training of a neural network are carried out iteratively through RL, involving a resource-intensive upstream training phase, resulting in a neural network controller that can then be executed very efficiently in deployment. When combined with a physics-based system model, it enables rapid and cost-effective generation of large volumes of training data, eliminating need for real-world interaction during training.

4.1 Reinforcement Learning Fundamentals

RL is a subfield of machine learning in which an agent learns to make decisions by interacting with an environment. This process can be modeled as a Markov Decision Process (MDP), defined by a set of *states*, *actions*, *transition dynamics*, and a *reward function*. The goal of an RL algorithm is to find an optimal strategy that maps states to actions in order to maximize cumulative long-term reward. This strategy is typically represented as

DOI 10.3384/ecp218921

a neural network, which can capture complex mappings between state and actions. This can even make high-dimensional, continuous, and partially observed state spaces controllable. Unlike other methods of machine learning, RL does not require pre-collected training data. Instead, an agent learns through trial-and-error interactions with an environment, enabling autonomous data collection and policy improvement.

A basic RL loop consists of an agent selecting an action based on its current policy, applying it to an environment, and receiving a new state and reward based on transition dynamics and reward function. The agent then updates its policy based on the observed outcome. In our setup, state transitions are computed within a Modelica model.

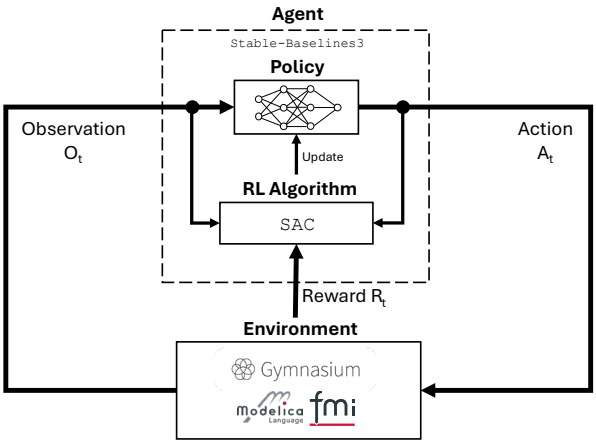


Figure 5. Block diagram of the RL loop, illustrating the interaction between components and the tools applied.

4.2 Coupling RL Environments with Modelica Simulations

Building an RL environment that incorporates physics-based simulations requires integration of multiple software frameworks and standardized interfaces. For this showcase, a training environment was constructed using an RL framework called `Gymnasium` (Mark Towers, Ariel Kwiatkowski et al., 2025), which enables the creation of customized environments via a Python API. `Gymnasium` follows a consistent structure tailored for RL tasks and is compatible with a wide range of existing algorithms, facilitating both development and testing.

The Modelica system model was then exported as a *Functional Mock-up Unit* (FMU), encapsulating relevant simulation equations and parameters according to FMI standard ('Functional Mock-up Interface', 2014). To perform physics-based state updates within an RL loop, the FMU was integrated using the `fmpy` library ('FMPy', 2024). During training, actions chosen by a RL agent are passed to a FMU via model inputs. In turn, FMU outputs are used as observations for an agent as well as values for reward computation. For this work, FMU version 2.0 in Co-simulation mode, which includes an internal solver to ensure stable and reliable simulations, is used. In comparison to Lukianykhin and Bogodorova (2019), the simulation is advanced in each RL step using the `doStep()` function provided by `fmpy`. This enables an RL loop that can be executed efficiently and more quickly, even when model complexity increases.

Listing 1. Illustrative Python code for integrating an FMU into a Gymnasium-based RL environment using FMPy library.

```
# called at beginning of the training
def __init__(…):
        fmu = fmpy.instantiate_fmu(fmu_directory,
        model_description, "CoSimulation")

# called at beginning of each training episode
def reset(…):
        fmu.reset()
        fmu.setupExperiment(startTime=0.0)
        fmu.enterInitializationMode()
        fmu.exitInitializationMode()

# called at each RL step
def step(…):
        fmu.setReal([parameter_names], [action_values])
        fmu.doStep(current_time, step_size)
        observations = fmu.getReal([parameter_names])
```

4.3 RL Agent Setup and Training Configuration

Integration of a RL agent is realized using `stable-baselines3` (Raffin et al., 2021). A library that is fully compatible with `Gymnasium` environments and provides a suite of reliable, state-of-the-art algorithms based on `PyTorch` (Jason Ansel et al., 2024). With only a few lines of code, a neural network is automatically initialized based on defined observation and action spaces and trained using a selected algorithm. Hyperparameters and training settings are freely adjustable to meet specific requirements of a control task.

Primary determinants of a controller's performance are *reward functions*, *observation and action spaces,* and an *environment* (including timestep length, episode termination criteria, initial state, etc.). These elements define an underlying mathematical optimization problem and thereby determine the global optimum. In contrast, the choice of *algorithms*, *hyperparameters*, and *neural network architecture* primarily affects training dynamics, convergence speed, and likelihood of reaching an optimal or near-optimal solution. In practice, especially for complex problems, training often converges only to a local optimum. As a result, these factors still have a significant impact on controllers' performance.

An action selected by an RL agent serves as control input to the system model and is derived from the current state and a probability distribution. In this case, the observable state consists of four state variables, $O_t = \{\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2\}$, representing angular positions and velocities of the

pendulums, respectively. θ_1 is measured relative to the downward vertical, and θ_2 is measured relative to the first pendulum. Thereby, $\dot{\theta}_1$ and $\dot{\theta}_2$ denote their respective angular velocities.

The action space includes the motor voltage, normalized to $-1 \leq a \leq 1$. This normalization improves learning efficiency, as many algorithms in `stable-baselines3` assume a Gaussian action distribution centered around zero.

The *reward function*, which evaluates the quality of the action-state pair at each timestep, is based on the work of Lee et al. (2025) for a double-inverted pendulum on a cartpole. It was adapted to match the system described in this paper and is composed of two components described in Eq. (4) and Eq. (5). A first component that focuses on pendulum position and a second one that focuses on pendulum velocity. As the objective is to stabilize the pendulum in an upright position with minimal angular velocity, an upright pendulum that is not moving yields the highest reward. This reward increases if the pendulum's height gets closer to the target height. Additionally, the reward increases with velocity if the pendulum's height is close to its initial position. However, if the pendulum is close to the target position, the reward increases when its velocity decreases. Unlike the approach taken by Lee et al. (2025), this reward function is specifically designed for a double-inverted pendulum. It ensures a reward path from the initial to the goal position that provides continuously increasing rewards when followed.

$$R_\theta = 0.5 * \sin(\theta_1) + 0.5 * \sin(\theta_1 + \theta_2) \quad (4)$$

$$R_{\dot{\theta}} = e^{\left(-0.02 * \sqrt{(\dot{\theta}_1)^2 + (\dot{\theta}_2)^2}\right)} \quad (5)$$

$$Reward = R_\theta \cdot R_{\dot{\theta}} \quad (6)$$

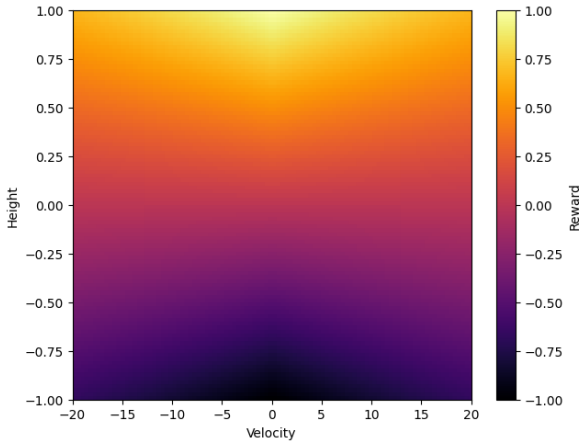


Figure 6. Reward function visualization for the double-inverted pendulum.

The environment is configured with an episode duration of 4 s, allowing the agent to interact with it up to 1'000 times per episode, given a reinforcement learning timestep of 4 ms. At the beginning of each episode, the pendulums are initialized in a downward position with zero angular velocity, corresponding to the natural starting state of a real-world pendulum system.

For training Soft Actor Critic (SAC) (Haarnoja et al., 2018), one of the most widely used modern off-policy RL algorithms, was used. It supports continuous action spaces and demonstrates strong performance in benchmark tests (Haarnoja et al., 2018; Huang et al., 2024). The hyperparameter configuration used in this work is summarized in Table 3.

Table 3. Hyperparameters used for this showcase with the SAC algorithm from stable-baselines3.

Hyperparameter	*Value*
Optimizer	Adam
Learning rate	0.0003
Discount factor (γ)	0.99
Replay buffer size	1e6
Number of hidden layers in critic networks	3
Number of neurons in hidden layers of critic network	512
Number of hidden layers of policy network	2
Number of neurons in 1st hidden layer of policy network	400
Number of neurons in 2nd hidden layer of policy network	300
Minibatch size	256
Nonlinearity	ReLU
Target smoothing coefficient (β)	0.005

Training was conducted on a Windows PC, parallelized across 12 cores of an Intel Xeon W-2275 3.3 GHz CPU. With this configuration, approximately 360 timesteps per second were processed during training.

4.4 Results and Discussion

As an initial step, RL training was conducted on a single-pendulum system. This simpler setup served as a validation of the toolchain by significantly reducing challenges associated with training and sim-to-real transfer. A stepwise increase in system complexity allows for targeted troubleshooting and a structured development approach. RL training for the single-pendulum system converged at a maximum episodic reward of 95%, using the reward function defined in Equation 7, where θ represents the pendulum angle relative to the downward vertical.

$$Reward = \cos(\theta) \quad (7)$$

DOI 10.3384/ecp218921

Convergence was reached after 500’000 timesteps, but training was extended to 1’100’000 timesteps to ensure model stability and robustness. The entire training process was completed in just 52 min. When evaluated with a system model, the trained neural network successfully stabilized the pendulum in an upright position.

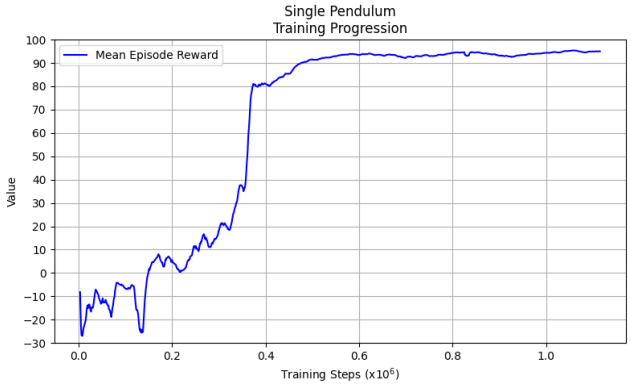


Figure 7. Mean episode reward during training of the single-pendulum system.

To deploy the trained neural network on a real-world system, it was converted into ONNX format, enabling interoperability and execution across different tools. The sim-to-real transfer introduced no additional issues, and the neural network successfully controlled the real-world system with performance comparable to the virtual system. The deployed controller for the single pendulum was also capable of handling external disturbances (e.g., applied forces) and inevitable model inaccuracies, demonstrating robust real-world performance.

Following the successful validation of the toolchain's functionality, it is applied to the double pendulum case. In that the toolchain was used to test various training configurations, each employing different training and pendulum parameters. These trainings have only been successful when system parameters have been chosen carefully. Figure 8 shows the mean reward progress for a representative successful training.

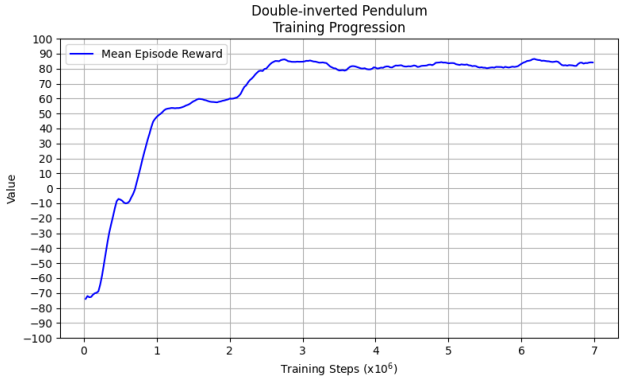


Figure 8. Mean episode reward during training of the double-inverted pendulum system.

In this example, training convergence was reached after 1’100’000 timesteps with a maximum mean episode reward of 86.5%. Despite some successful trainings, a sim-to-real transfer has not yet been achieved. In many cases, the controller could make the real-world double-inverted pendulum swing upwards but failed to balance it.

The conducted grid search indicates that stable control is much more likely when the proportions of the pendulums meet specific conditions. The first observed requirement is an appropriate ratio between the time constants of the motor and second pendulum that must be bigger than 28.5 to ensure stable control. (Eq. 8) The time constant of the second pendulum is determined solely by its geometry and friction, whereas the motor's time constant depends on the geometry of the first pendulum and a small control loop delay. Since this delay is minimal, it can be considered negligible. The second condition is related to the ratio between the damping coefficient D (i.e., the friction in the joint) and the second pendulum's moment of inertia. (Eq. 9) In the trainings, successful controllability could only be observed if this ratio was lower than 3.0.

$$\frac{\tau_{rod,2}}{\tau_{motor}} = \frac{\tau_{rod,2}(I_2, m_2, l_{cm}, D)}{\dot{\tau}_{motor}(\tau_{motor}, I_1)} > 28.5 \tag{8}$$

$$\frac{D}{2 * I_2} < 3.0 \tag{9}$$

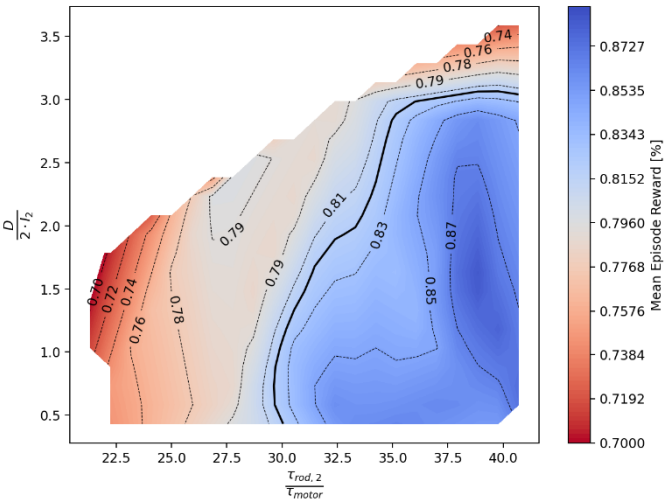


Figure 9. Controllability Analysis for double-inverted pendulum parameters.

Figure 9 visualizes the subset of our sweep that yielded mean rewards > 0.82. Within this empirical region controllability was consistently observed. This region can be accessed through a variety of parameter combinations. A straightforward approach would involve utilization of an electrical motor characterized by a low time constant. It is also plausible to utilize a secondary pendulum with an enhanced time constant. However, this requires a more detailed evaluation due to the dependence of the previously mentioned second condition on properties of the second pendulum.

5 Conclusion and Outlook

This work presents a comprehensive toolchain for training and deploying neural network control systems using reinforcement learning (RL) in combination with physics-based simulations developed in Modelica. A key achievement of this work is the successful transfer of a controller trained entirely in a virtual environment to a real-world system, demonstrating the feasibility of simulation-based training for physical control tasks. By offering a unified and structured workflow, this approach provides an accessible toolchain for both research and educational purposes. The use of well-established tools and frameworks ensures an efficient, reliable, and reproducible development process, laying the foundation for future applications of RL-based neural network controllers in complex, real-time dynamic systems across academic and industrial domains.

The RL training process poses a complex optimization challenge, where success relies heavily on the selected training setup, including algorithm choice, neural network architecture, and reward-shaping strategies. Numerous environment and training parameters, such as hyperparameters, observation augmentation, and initial state distributions, are highly system-specific and interact in intricate ways, adding further complexity to the optimization task. Identifying a configuration that allows a RL agent to reliably converge toward an optimal or near-optimal policy often requires repeated experimentation, with each training run being both time- and computation-intensive. Leveraging prior domain knowledge, heuristics, and insights from previous experiments can help reduce this search space, yet the process remains inherently challenging and iterative.

During the development of this showcase, it became evident that achieving good system controllability is a fundamental yet challenging prerequisite for successful RL training. The controllable region of this system is not intuitively defined, making it difficult to identify suitable configurations, and even a well-designed RL setup cannot compensate for a system that is only partially or poorly controllable. For this double-inverted pendulum case, ensuring controllability requires carefully selecting parameter ratios between the pendulum components, such as their respective moments of inertia, damping coefficients, and lengths, to guarantee that the system can, in principle, be well stabilized. Ongoing work focuses on improving these mechanical design aspects and thereby enabling a sim-to-real transfer of the learned policies. Completing these steps will be essential not only for finalizing the double-pendulum showcase but also for strengthening the reliability, scalability, and educational value of the entire toolchain as a practical teaching and research platform for reinforcement learning in real-world control applications.

5.1 Future Work

The next phase in the development of this showcase focuses on redesigning the physical system to enable a sim-to-real transfer for the double-inverted pendulum. This involves adjusting the system's proportions to align with the empirical guidelines identified in Chapter 4.4.

Optimization efforts will primarily target pendulum components. The first pendulum will be redesigned to reduce its moment of inertia, thereby improving the motor's time constant. In parallel, the second pendulum will be modified to increase its moment of inertia. Since replacing the motor would be significantly more complex, optimizing the pendulum design presents a more practical and effective approach to start with.

Other potential further steps will be focused on:

- *Controller implementation for embedded systems:* Investigating the use of trained neural network controllers on embedded platforms such as industrial PLCs (e.g., Beckhoff SPS)
- *Enhancing controller robustness through system randomization:* Applying randomization techniques during training to improve sim-to-real transfer, increasing resilience to virtual–physical discrepancies, sensor noise, and limitations in the control loop or actuators
- *Educational applications:* The developed showcase can be utilized in lectures to teach reinforcement learning (RL) in an engaging and interactive manner. A possible exercise would involve designing and comparing different reward functions for balancing the pendulums in an upright position. Using this provided toolchain, students can gain practical, hands-on experience with RL training. Furthermore, the most promising neural networks trained in simulation could be deployed on the physical system, enabling a real-world demonstration of RL.

Acknowledgements

The authors acknowledge Lucerne University of Applied Sciences and Arts (School of Engineering and Architecture) for supporting this study.

Language editing and writing style improvements were assisted by generative AI tools, including ChatGPT and DeepL. Scientific content, concepts, and results were independently developed by the authors.

References

Biscani, F. and Izzo, D. (2020) 'A parallel global multiobjective framework for optimization: pagmo', *Journal of Open Source Software*, 5(53), p. 2338. Available at: https://doi.org/10.21105/joss.02338.

'FMPy' (2024). Dassault Systems. Available at: https://github.com/CATIA-Systems/FMPy (Accessed: 25 April 2025).

'Functional Mock-up Interface' (2014). Modelica Association. Available at: https://fmi-standard.org/ (Accessed: 25 April 2025).

Haarnoja, T. et al. (2018) 'Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor'. arXiv. Available at: https://doi.org/10.48550/ARXIV.1801.01290.

Heuermann, A. et al. (2023) 'Accelerating the simulation of equation-based models by replacing non-linear algebraic loops with error-controlled machine learning surrogates', in. *15th International Modelica Conference 2023, Aachen, October 9-11*, pp. 275–284. Available at: https://doi.org/10.3384/ecp204275.

Huang, S. et al. (2024) 'Open RL Benchmark: Comprehensive Tracked Experiments for Reinforcement Learning'. arXiv. Available at: https://doi.org/10.48550/ARXIV.2402.03046.

INNOSUISSE (2025) *IntelliBake*, *ARAMIS*. Available at: https://www.aramis.admin.ch/Grunddaten/?ProjectID=54123 (Accessed: 23 April 2025).

Jason Ansel et al. (2024) 'PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation', in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*. *ASPLOS '24: 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, La Jolla CA USA: ACM, pp. 929–947. Available at: https://doi.org/10.1145/3620665.3640366.

Lee, T., Ju, D. and Lee, Y.S. (2025) 'Transition Control of a Double-Inverted Pendulum System Using Sim2Real Reinforcement Learning', *Machines*, 13(3), p. 186. Available at: https://doi.org/10.3390/machines13030186.

Lukianykhin, O. and Bogodorova, T. (2019) 'ModelicaGym: Applying Reinforcement Learning to Modelica Models'. Available at: https://doi.org/10.48550/ARXIV.1909.08604.

Mark Towers, Ariel Kwiatkowski et al. (2025) 'Gymnasium: A Standard Interface for Reinforcement Learning Environments'. Farama Foundation. Available at: https://gymnasium.farama.org/ (Accessed: 23 April 2025).

Modelica Association (2020) 'Modelica Standard Library'. Modelica Association.

Morcego, B. et al. (2023) 'Reinforcement Learning Versus Model Predictive Control on Greenhouse Climate Control'. arXiv. Available at: https://doi.org/10.48550/ARXIV.2303.06110.

'ONNX Project' (2024). ONNX Community. Available at: https://onnx.ai.

'ONNX Runtime' (2021). ONNX Runtime developers. Available at: https://onnxruntime.ai/ (Accessed: 25 April 2025).

Raffin, A. et al. (2021) 'Stable-Baselines3: Reliable Reinforcement Learning Implementations', *Journal of Machine Learning Research*, 22(268), pp. 1–8.

Sergi, T. and Brun, J. (2024) 'MoONNX'. Available at: https://github.com/sertho/modelica-ONNX.

TensorFlow Developers (2025) 'TensorFlow'. Zenodo. Available at: https://doi.org/10.5281/ZENODO.4724125.

Watson et al. (2024) 'KerasHub'. Available at: https://github.com/keras-team/keras-hub.

DOI
10.3384/ecp218921

Modeling and Simulation of a Direct Heat Recovery System for Cabin Heating in Battery-Powered Mobile Machines

Fabian Lagerstedt[1,2] Samuel Kärnell[2] Marcus Rösth[3] Liselott Ericson[2]

[1]Huddig AB, Sweden, fabian.lagerstedt@huddig.se
[2]Division of Fluid and Mechatronic Systems, Linköping University, Sweden,
{samuel.karnell,liselott.ericson}@liu.se
[3]Hudiksvalls hydraulikkluster, Sweden, marcus@rosth.com

Abstract

The transition from internal combustion engines (ICE) to alternative technologies—such as battery-electric powertrains—in mobile machinery places increased demands on thermal management systems. Cooling requirements below the ambient temperature during the summer, and intensive heating requirements during the winter, lead to holistic but complex integrated solutions where energy efficiency is of high priority. Research into integrated system solutions including heat pumps and waste-heat recovery has been carried out mainly on passenger cars. In this study, mobile machines are considered, and an articulated excavator-loader—also known as backhoe loader—is used as an example. Apart from operating tasks, times, and conditions, the system architecture under the hood differs from architectures usually found in passenger cars, including working hydraulic systems. During the early stages of vehicle development, modeling and simulation of integrated thermal management systems are crucial for proof-of-concept, developing control strategies, and understanding subsystem interactions. These processes rely on data that would otherwise require testing on a complete vehicle. This paper presents a model of a heat recovery system for cabin heating using the DLR ThermoFluid Stream Modelica library, together with input data from previous research based on experiments on a series-hybrid electric machine. The study investigates the initial feasibility and performance of a direct heat recovery system for the architecture of a battery-powered mobile machine. The results show that a simple system design can provide a strong foundation for cabin heating under many of the studied excavating conditions, though it does not fully match the performance of the reference system, which is supplied with heat from an ICE. *Keywords: Thermal management, electric vehicle, heat recovery, cabin heating, mobile machine, ITMS*

1 Introduction

Battery-powered vehicles have become a common sight on the roads. Despite this, the electrification of heavy off-road mobile machines seems delayed in comparison with passenger vehicles. This can be partly explained by small production series and the design challenges arising with varying or unknown duty cycles, in combination with high performance requirements, and end user expectations (Beltrami et al. 2021; Lajunen et al. 2016). Mobile machines are specialized production vehicles designed for specific tasks, where reduced operating costs, minimized downtime, and enhanced performance drive the electrification (Monnay 2017; Lajunen et al. 2016). Battery as an energy storage technology is a main contender in this ongoing transition (Niemelä et al. 2023), where energy density, component costs and demanding operating conditions are some of the major challenges. To mitigate costs for end users and enhance the viability of battery-powered mobile machines, energy efficiency is a high priority—particularly because the cabin requires substantial energy for heating during winter—where thermal management systems play an important role.

Replacing the internal combustion engine (ICE) with electric motors will likely increase the total energy efficiency of the vehicle. However, removing the ICE will also remove a lot of available waste heat, which was previously especially useful during winter to keep the cabin warm and comfortable. Instead, this energy needs to be taken from somewhere else. Ideally, the waste heat previously recovered from the ICE could instead be recovered from the electric motors in a battery-powered mobile machine. On the other hand, the amount of available heat flow from one single source may not always be sufficient to cover the cabin heat demand. Heat sources on these types of vehicles are also specifically dependent on the operating mode of the machine (Lagerstedt et al. 2024). An integrated thermal management system (ITMS)—that enables the collective recovery of heat from multiple components and subsystems within the machine—is therefore a necessary strategy moving forward in order to reduce the energy consumption of the thermal management system and to fulfill the runtime requirements of the machine at lowest possible cost impact on the end user.

Unlike passenger cars, mobile machines are primarily built to perform paid work. In some cases by being stressed to the system limits during a full mission, in others by only occasional peak loads. In both cases, system cost and reliability are key aspects to ensure the profitabil-

ity of the operation. Since the maximum available runtime at high-power utilization is crucial for a mobile machine, the overall vehicle efficiency at these operating points defines the minimum required energy storage needed to reach the lower acceptable runtime—and thus the machine cost. As illustrated in Figure 1, the potential for recovering waste heat is greatest during short runtimes—corresponding to high-utility operation—where effective heat recovery methods are most beneficial, while longer runtimes may require supplementary heating, though this does not necessarily affect the required energy storage if the added energy remains small compared to the high-utility consumption.

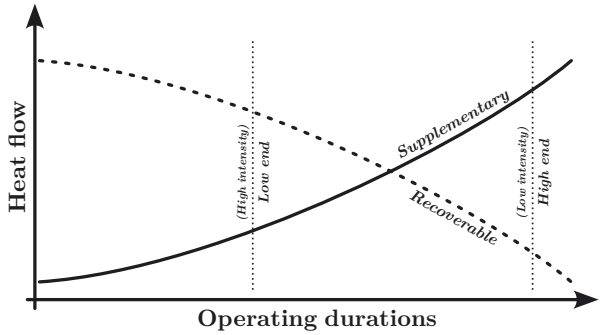


Figure 1. Conceptual representation of the heat flows over operating durations—from a full energy storage to empty—for an electrified mobile machine during cold conditions. At short operating durations—corresponding to high-power utilization—the amount of recoverable heat is high, reducing the need for supplementary energy. As power utilization decreases and operating durations increase, recoverable heat drops, and supplementary heating becomes increasingly necessary. Fictive duration limits, marked as 'Low end' and 'High end' along the horizontal axis, illustrate the design requirement for a supplementary heat source.

Relying solely on waste-heat sources from a battery-electric powertrain may not provide satisfactory heating results in the cabin if the power output from the vehicle is not high enough. To counter this issue, a simple and commonly used method in the history of electrification has been to add resistive heating elements to supplement insufficient heat flow (He et al. 2024). Another solution is to use the existing air conditioning system as a heat pump, which is gaining attention due to the range loss often related to the use of resistive heaters (Kwon et al. 2024). Even if a heat pump system can be designed so that waste heat is recovered from various heat sources in the vehicle, a heat pump is a supplementary heat source requiring energy to power the compressor. Thus, an efficiency limit is introduced which is defined by the energy consumption of the heat pump system. This limit can be passed with a direct heat recovery system, if the amount of waste heat within the system is high enough for the heating demands.

Since the most studied use case within the research of ITMS solutions still is the passenger car (Leoncini et al. 2024), more knowledge is needed regarding mobile machines and the implications of differences in system architecture and duty cycles on the availability of waste heat—and viable methods for recovering it.

1.1 Mobile Machines

In this and previous studies, an articulated excavator-loader—also known as backhoe loader—is used to represent a mobile machine (Lagerstedt et al. 2024; Lagerstedt et al. 2025). The Huddig 1370T, illustrated in Figure 2, is an example of an electric hybrid mobile machine with a specified standard weight of 14.6 t and a top speed of 50 km/h.



Figure 2. Illustration of an articulated excavator-loader, the Huddig 1370T—an example of a mobile machine capable of both road transportation and load handling, featuring a front loader and a rear excavating unit.

Mobile machines in general—and Huddig machines in particular—are multi-purpose vehicles. The main purpose is rarely human transportation—however, many mobile machines still have a physical operator on board with requirements on comfort and visibility. Standardizing mobile machine duty cycles is challenging, due to the variety of functions, capacity and operating conditions. In a previous study, one approach to evaluate thermal loads during extreme, high-utility operation was presented (Lagerstedt et al. 2024). The approach was based on dividing the machine's mode of operation into the two known, most energy-intensive and contrasting modes—transportation and excavating. Combining the operating modes with ambient temperatures, forms a coordinate system of machine operation illustrated in Figure 3. Extreme operating points were selected in each corner and evaluated experimentally on the electric hybrid machine Huddig 1370T to estimate the thermal loads. The thermal loads divided into approximate temperature categories, together with the cabin heating demand, is illustrated in Figure 4.

In this study, a proposed system to recover parts of the available waste heat in order to warm the vehicle cabin is modeled, simulated and evaluated. The proposed system is a direct heat recovery system, meaning that waste heat is transferred to its destination as sensible heat, without phase changes or significant mechanical work on the transport medium. The system covers heat

DOI 10.3384/ecp218931

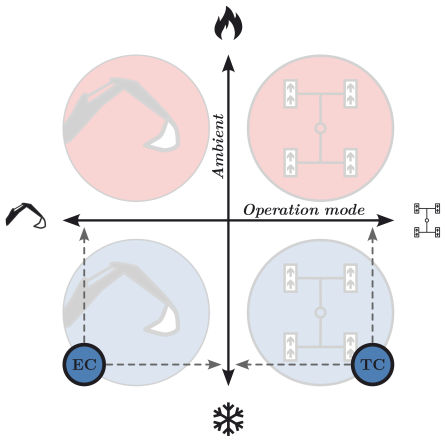


Figure 3. Overview of the operating scenarios for a mobile machine, defined by ambient temperature on the vertical axis and operating mode on the horizontal axis. 'Excavating Cold' (EC) and 'Transportation Cold' (TC) marks operating points previously evaluated by experiments—representing corner cases during cold ambient conditions (Lagerstedt et al. 2024). The dashed lines illustrate the operating point directions of interest to evaluate through simulations in this paper.

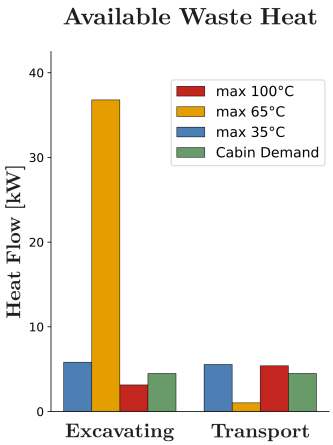
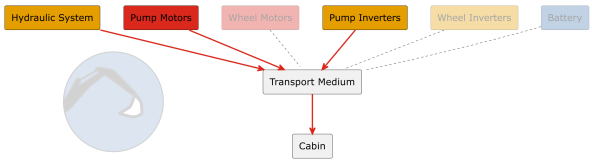


Figure 4. Bar chart from previous work, illustrating available waste-heat flow on a battery-powered mobile machine at worst case operating points during cold ambient conditions—excavating and transportation at –15°C (Lagerstedt et al. 2024).

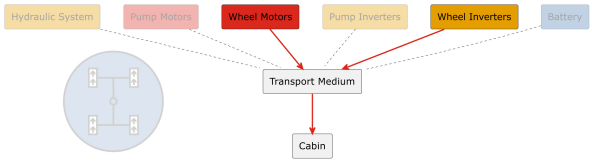
recovery in the two separate operating modes excavating and transportation. During excavating, heat is recovered from the hydraulic system—which is significant for a mobile machine—along with the electric motors and inverters powering the hydraulic pumps. The heat recovery principle for excavating is illustrated in Figure 5a. For transportation, the heat recovery principle is more similar to that of a road transportation vehicle and is illustrated in Figure 5b.

1.2 Contributions

This study investigates the initial feasibility and performance of an ITMS for the architecture of a battery-powered mobile machine. A simple and low-complexity



(a) Heat recovery paths during excavating.



(b) Heat recovery paths during transportation.

Figure 5. Principal overview of the proposed direct heat recovery system during excavating and transportation respectively, as studied in this paper. Heat is recovered from the active subsystems via a transport medium that delivers thermal energy to the cabin. Solid red arrows indicate the active heat flow paths in the respective operating mode, while dashed lines represent inactive heat paths in the ITMS. The heat sources are colored according to the temperature categories used in Figure 4.

system design of a heat recovery system is suggested, adopting the main principle of a traditional heat recovery system—but replacing the ICE with heat sources related to an electrified mobile machine. By the modeling and simulation of the proposed heat recovery system, a wide variety of operating points can efficiently be evaluated, and the effect of control strategies can be understood.

2 Modeling

The models presented in this section were made using the DLR Thermofluid Stream Library (TFS) (Zimmer 2020; Zimmer, Meißner, and Weber 2022), together with the Modelica Standard Library and the modeling and simulation tool OpenModelica v1.22.2. Both the libraries used and the simulation tool are open-source, which was an important reason behind the selected approach. Additionally, ITMSs in electric vehicles can evolve into complex designs with many actively controlled flow routes, where the TFS library can serve as a strong alternative (Zimmer, Meißner, and Weber 2022). Two models were created, where the first model represents a traditional heat recovery system on a vehicle powered by an ICE, in this study acting as a reference system. The second model represents the proposed system, but with two different control strategies. The control strategies refer to the mass flow control of the transport medium—the first maintains the same fixed flow as the reference system, while the second actively controls the flow to keep the electric motors at their maximum temperature.

To evaluate the system models in the different operating points illustrated in Figure 3, two parameters are used—ambient temperature T_{amb} for the vertical axis, and intensity factor X_I for the horizontal axis. The intensity factor

is a simplified representation of both operating mode and degree of operating intensity, where negative and positive values correspond to excavating and transportation respectively, with –1 and 1 both being equal to maximum intensity. The models are evaluated in the temperature range –15 to 15°C.

2.1 Reference System

First, a baseline system was modeled representing a heat recovery system of a traditional vehicle powered by an ICE, and is illustrated in Figure 6. The model consists of two separate circuits—one transport medium circuit delivering heated glycol-mixed water to the heating core in the cabin, and one circuit heating ambient air and delivering it to the cabin.

The transport medium circuit, highlighted with thick blue lines, is modeled as an open circuit with a fixed temperature source of 82°C and a constant flow. The constant flow is modeled with a pump component with input signal from a mass flow regulator, consisting of a PI controller and a limiter. Due to the excessive heat flow available in the ICE, and the thermostat maintaining the engine temperature via an external heat rejection system, the heat supply to the cabin is assumed to be constant. The temperature value was measured in previous work (Lagerstedt et al. 2024).

The second circuit, highlighted with a purple zone, corresponds to the cabin and the fresh air supply, modeled as an open circuit with a fixed air flow—maintained with the same method as the transport medium—corresponding to the maximum possible air flow based on previous measurements (Lagerstedt et al. 2024). The cabin is modeled as a volume component, with the output temperature state representing the cabin temperature in this study. The temperature of the source component is defined with the parameter T_{amb}.

2.2 Proposed System

The system model—covering the hydraulic system, pump motors, pump inverters, wheel motors and wheel inverters—is illustrated in Figure 7, and operates according to the two heat recovery principles presented in Figure 5a and Figure 5b. The system consists of four circuits—two representing the working hydraulic system and its heat rejection system, one transport medium circuit, and a cabin air flow circuit identical to that found in the reference system. Aside from TFS library components described in detail in (Zimmer, Meißner, and Weber 2022), standard Modelica components are used for the control logic and to group the inverter and electric motor models for space-saving purposes. The heat losses rejected by each part of the system are specified in the constants, multiplied with the absolute value of the intensity factor $|X_I|$.

The working hydraulic system, highlighted in a yellow zone in Figure 7, was based on a model used in previous work but with slight modifications (Lagerstedt et al.

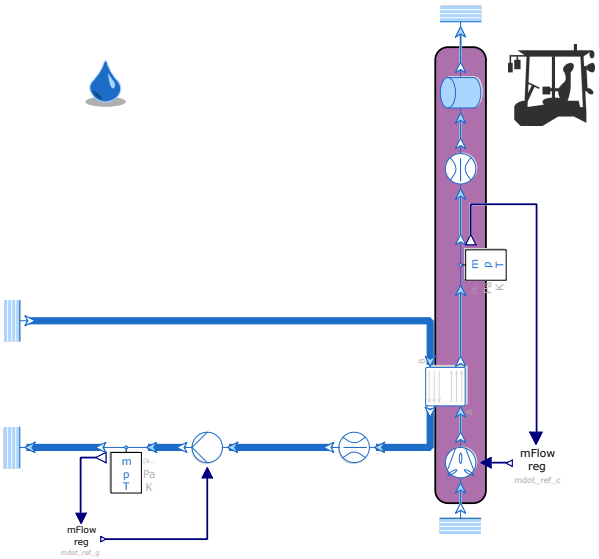


Figure 6. Modelica model of the reference system—aimed to represent a traditional vehicle powered by an ICE—composed of a cabin highlighted in purple and a transport medium circuit modeled as a source with constant flow and temperature.

2025). In this study, a PI controller is used to control the ambient chiller, while constant heat flow and mass flow are used to create a more computationally lightweight model. The hydraulic circuit is modeled as a closed system consisting of a reservoir, a flow restriction, and two heat exchangers—one for heat recovery and one for heat rejection. The circuit leading through the heat rejection exchanger is designed to maintain a maximum allowed temperature of 65°C in the hydraulic reservoir, since the hydraulic heat flow far exceeds the cabin demand. The model represents the return line of the working hydraulic system, transporting the hydraulic heat losses to the ambient chiller. The hydraulic heat losses are modeled as a conduction element and a pump component. The pump component is controlled to maintain a specified hydraulic flow. The conduction element is supplied with a specified heat flow. Both the specified mass flow and heat flow are multiplied with the absolute value of the intensity factor $|X_I|$.

The thick blue lines in Figure 7 highlights the transport medium circuit, creating the integration between vehicle heat sources and the cabin. For each operating mode, the active heat sources are connected in series, while the circuits for the inactive systems are bypassed. The activation and deactivation of systems are controlled by Modelica Standard Library components, together with switching valves from the TFS library. The pump component in the transport medium circuit is controlled by a mass flow regulator—described earlier in the paper—which maintains a fixed flow rate identical to that of the reference system. An alternative control method for the transport medium mass flow rate was also suggested which maintains the active motor temperatures at 100°C—an approximate maximum operating temperature—and is illustrated

DOI
10.3384/ecp218931

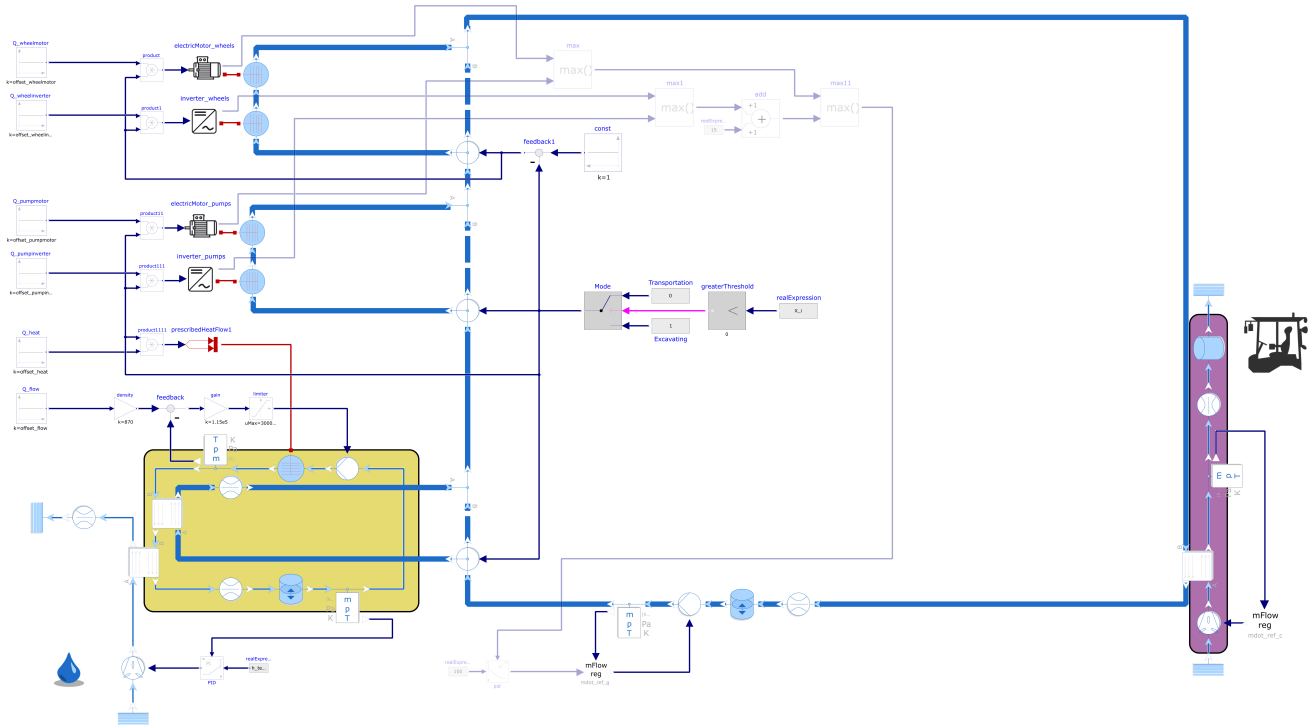


Figure 7. Modelica model of the proposed system, composed of a cabin highlighted in purple, heat sources, and a transport medium circuit highlighted with thick blue lines. The heat sources in the left area of the figure are modeled in series, with loops enabled by switching valves depending on the active operating mode parameter X_I. The hydraulic system, highlighted by the yellow zone, is enabled together with the loop for the inverter and electric motor powering the hydraulic pumps. The transparent lines and components correspond to the alternative mass flow control strategy.

by the transparent components and lines in Figure 7.

To obtain a system model slightly more compact and easier to read, separate but identical models to represent electric motors, and inverters were created, as illustrated in Figure 8. The model converts a real signal into a heat flow, acts as a thermal mass, and converts the temperature state of the heat port into an output value in Celsius. The model is built using Modelica Standard Library components, and the only difference between the motor and inverter models is the icon.

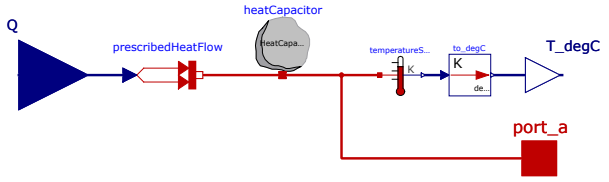


Figure 8. Modelica model of the inverter and electric motor—used to improve the readability of the system model—composed of a prescribed heat flow, a heat capacitor, and a temperature conversion component. The only differences between the inverter and motor models are the icon and the heat capacity parameter value.

3 Results

In this section, simulation results from the reference system and the proposed direct heat recovery system for a battery-powered mobile machine are presented. The reference system represents a conventional cabin heating system based on heat recovery from an ICE. The proposed system adopts the main principle of the reference system—but with heat recovery from alternative sources—and is evaluated using two different strategies. First, with the transport medium mass flow rate fixed at the same level as in the reference system—and second, with the mass flow rate controlled to maximize the electric motor temperature. The two strategies are evaluated in comparison with the reference system and with one another. Contour plots are used to illustrate performance across different operating modes, ambient temperatures, and operating intensities.

In Figure 9, the simulated cabin temperature from the reference system is shown, using the maximum available heating power. The straight contour lines illustrate how the traditional cabin heating performance is independent of operating mode and intensity. The different cabin temperatures are an effect of the intake air temperature varying with the ambient temperature on the y-axis. The high absolute temperature values observed are a consequence of the system's passive exposure to ambient temperatures not being included in the model. This result is used as a baseline for comparison with the proposed system.

Figure 10 shows the simulated cabin temperature from the proposed heat recovery system with a fixed transport

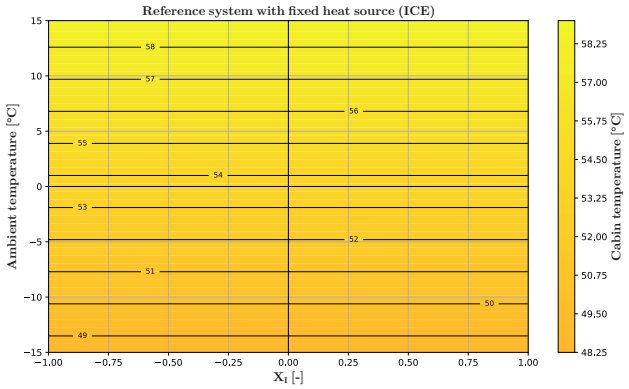


Figure 9. Resulting contour plot of the simulated cabin temperature using the reference system at maximum heating power, with fixed transport medium mass flow rate and temperature. Ambient temperature is represented on the y-axis and intensity factor on the x-axis, where negative and positive values correspond to excavating and transportation, respectively.

medium mass flow rate—relative to the reference system results shown in Figure 9. The mass flow rate is the same as that of the reference system. Negative values of cabin temperature difference means that the cabin temperature is lower compared to the reference system, which is true for all simulated operating points. The cabin temperature is generally higher in excavating mode—denoted by negative values of X_I in the plot—compared to transportation mode, denoted by positive values, as indicated by the smaller cabin temperature differences.

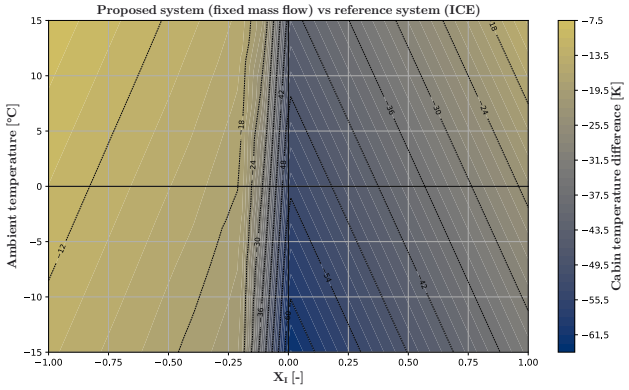


Figure 10. Resulting contour plot of the simulated cabin temperature using the proposed system at maximum heating power—with fixed transport medium mass flow rate—shown relative to the reference system results. Ambient temperature is represented on the y-axis and intensity factor on the x-axis, where negative and positive values correspond to excavating and transportation, respectively.

Simulation results for the proposed system, but with active control of the transport medium mass flow rate is seen in Figure 11. Compared to the results with a fixed mass flow rate, the cabin temperature is more similar between the operating modes—excavating and transportation. The cabin reaches slightly higher temperatures during medium to high intensity transportation, while the opposite is true across all intensities of excavating.

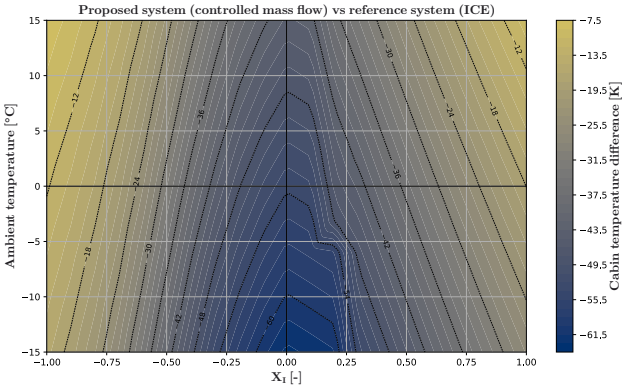


Figure 11. Resulting contour plot of the simulated cabin temperature using the proposed system at maximum heating power—with controlled transport medium mass flow rate—shown relative to the reference system results. Ambient temperature is represented on the y-axis and intensity factor on the x-axis, where negative and positive values correspond to excavating and transportation, respectively.

To better understand the results in Figure 11, a comparison between the two strategies is shown in Figure 12, where the cabin temperature achieved with controlled transport medium mass flow rate is shown relative to the fixed flow rate case. It is observed that the cabin temperature during transportation increases with the control strategy, but decreases during excavating. One exception to the improved performance during transportation is observed at intensities $0 \leq X_I \leq 0.2$, where the electric motors do not reach their maximum temperature and therefore do not demand an increased mass flow rate. During excavating, the strategy of controlling the mass flow rate solely to maintain the electric motors at 100°C results in reduced heat recovery from the working hydraulic system. Even though a significant amount of waste heat is available in the hydraulic system at most operating points, the recovered heat depends on—and is limited by—the transport medium mass flow rate, as the hydraulic system temperature is maintained at 65°C by the ambient chiller.

In Figure 13, the recovered heat flow from each subsystem and the cabin's heating performance are shown for the worst-case operating points during excavating and transportation—indicated by 'EC' and 'TC' in Figure 3. Apart from confirming the earlier presented results—that the heating performance of the cabin with the proposed heat recovery system is lower than for the reference system—it is shown that all the available waste heat from the motors and inverters are recovered in both operating modes. However, during excavating only a fraction of the available hydraulic waste heat is recovered. As already mentioned, this is due to the temperature limit in the hydraulic system, here set to 65°C to protect the oil from degrading.

DOI 10.3384/ecp218931

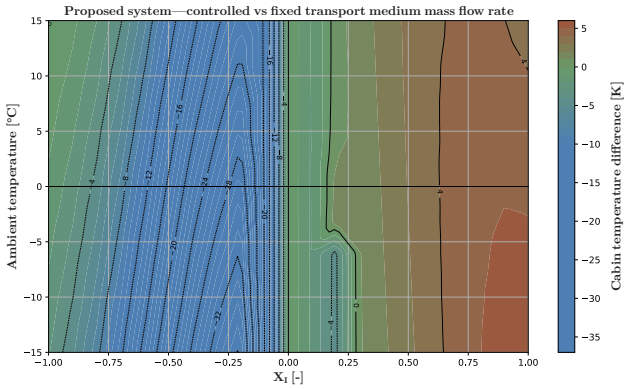
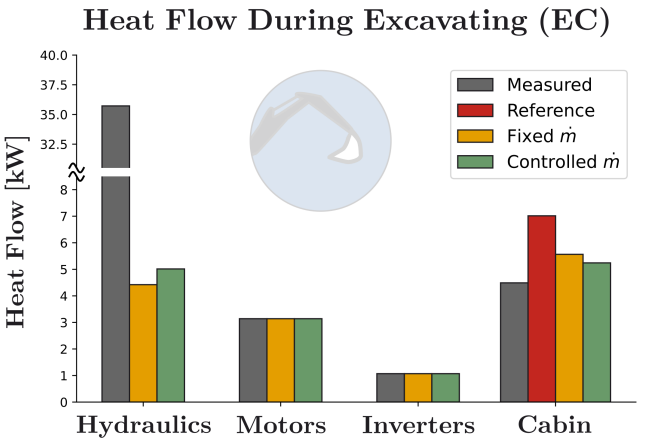
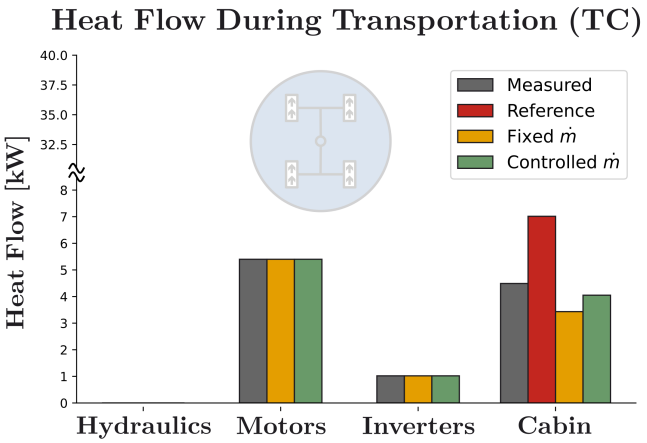


Figure 12. Resulting contour plot of the simulated cabin temperature using the proposed system at maximum heating power—with controlled transport medium mass flow rate—shown relative to the proposed system results with fixed mass flow rate. Ambient temperature is represented on the y-axis, and intensity factor on the x-axis, where negative and positive values correspond to excavating and transportation, respectively.



(a) Heat flows during excavating.



(b) Heat flows during transportation.

Figure 13. Simulated heat flows in the worst case operating points, marked as 'EC' and 'TC' in Figure 3. The categories on the horizontal axis refer to the subsystems in the studied heat recovery systems. The gray bars correspond to the measured values of available waste heat and cabin heat demand, illustrated in Figure 4. The red bar is the cabin heating performance in the reference system. The yellow and green bars show recovered heat from working hydraulics, motors and inverters, and the cabin heating performance for each control strategy respectively.

4 Discussion

The purpose of this paper was to investigate and understand the potential of direct heat recovery for warming the cabin of a battery-powered mobile machine—and to what extent the heat source of an ICE can be replaced by multiple alternative sources. A proposed direct heat recovery system was modeled and simulated using two different control strategies, and the results were compared with a reference system. The focus of this study was to understand the initial feasibility of the direct heat recovery system principle. Even though previously measured data has been used within the model, some uncertainty remains regarding component parameters. Therefore, the results are compared between models with equal uncertainty, and absolute values should be interpreted with caution.

The results suggest that the heat recovery system and control strategies as presented in this paper do not match the performance of the reference system—despite the available waste heat shown in Figure 4, where the heat exceeds the cabin demand. However, it is demonstrated that a simple system design can warm the cabin while keeping the included system components within their temperature limits. The results further indicate that the heating performance can be improved by the selection of control strategy—especially in order to maximize the heat recovery from the working hydraulic system during excavating. Even though the performance in the reference system is not reached, there is room for further improvement, including solutions to close the performance gap. To improve the heat recovery during excavating, the transport medium flow could be divided into two parallel circuits, with the power electronics in one circuit, and the hydraulic system in the other. This way, maintaining the electric motor at 100°C will not negatively affect the heat recovery from the working hydraulic system. Including the battery in the heat recovery system is also a possible option, but it would likely increase system complexity due to its low nominal temperature. To close the heating performance gap, stored electric energy in the battery could be used when needed—either through the implementation of a heat pump or via resistive heating.

A direct heat recovery system is—as indicated by the results—dependent on the actual temperature on the individual heat sources. In this study the results are presented during steady state, after 8 hours continuous operation. In a product application, the heat-up time before heat can be effectively recovered becomes important, and should be considered in a comparison with alternative systems designs or principles.

The presented models are simplified such that the ambient effects on the system are not considered, other than the

simulation starting temperature, and the intake air temperature of the cabin heating core. This makes the resulting cabin temperatures only valid in relation to the reference system, where the same simplifications are used. Additionally, a cabin on a vehicle commonly uses an air recirculation system, not included in the models presented here. The absence of air recirculation leads to an overestimation of the modeled heating power, which can be observed in Figure 13, and an underestimation of the cabin temperatures—however, the same applies to both systems, keeping them comparable.

Finally, using the cabin performance powered by an ICE as a reference system and a baseline to determine whether a proposed system design is 'good enough' may not be entirely relevant—due to the fact that cabins on traditional vehicles are not designed for limited heat sources and high energy efficiency. However, it can serve as a useful benchmark for evaluating the specific performance of a chosen design and control strategy—particularly in future assessments aimed at closing the heating power gap.

5 Conclusions

Direct heat recovery from inverters, electric motors, and the working hydraulic system is not sufficient on its own to replace the heat delivered by an ICE—but it can provide a significant foundation for cabin comfort management during excavating, offering low system complexity.

Acknowledgments

This research was funded by the Swedish Energy Agency (Energimyndigheten) with grant number P2023-00596.

References

Beltrami, Daniele et al. (2021-09). "Electrification of Compact Off-Highway Vehicles—Overview of the Current State of the Art and Trends". en. In: *Energies* 14.17, p. 5565. ISSN: 1996-1073. DOI: 10.3390/en14175565. URL: https://www.mdpi.com/1996-1073/14/17/5565 (visited on 2024-10-27).

He, Liange et al. (2024-11). "Performance investigation of integrated thermal management system for electric vehicle with waste heat recovery of electric drive system". en. In: *Journal of Energy Storage* 102, p. 114075. ISSN: 2352152X. DOI: 10.1016/j.est.2024.114075. URL: https://linkinghub.elsevier.com/retrieve/pii/S2352152X24036612 (visited on 2024-10-26).

Kwon, Soonbum et al. (2024-06). "Performance comparison of a direct heat pump using R1234yf and indirect heat pumps using R1234yf and R290 designed for cabin heating of electric vehicles". en. In: *Energy* 297, p. 131311. ISSN: 03605442. DOI: 10.1016/j.energy.2024.131311. URL: https://linkinghub.elsevier.com/retrieve/pii/S0360544224010843 (visited on 2024-10-26).

Lagerstedt, Fabian et al. (2024). "Analysis of Opportunities for Integrated Thermal Management on Battery Powered Mobile Machines". In: *Global Fluid Power Society PhD Symposium, GFPS 2024*. submitted.

Lagerstedt, Fabian et al. (2025). "Potential of Hydraulic Waste Heat Recovery for Cabin Comfort". In: *The 19th Scandinavian International Conference on Fluid Power, SICFP'25*. submitted.

Lajunen, Antti et al. (2016-03). "Electric and Hybrid Electric Non-Road Mobile Machinery – Present Situation and Future Trends". en. In: *World Electric Vehicle Journal* 8.1, pp. 172–183. ISSN: 2032-6653. DOI: 10.3390/wevj8010172. URL: https://www.mdpi.com/2032-6653/8/1/172 (visited on 2024-10-28).

Leoncini, Gabriele et al. (2024-01). "A review on challenges concerning thermal management system design for medium duty electric vehicles". en. In: *Applied Thermal Engineering* 236, p. 121464. ISSN: 13594311. DOI: 10.1016/j.applthermaleng.2023.121464. URL: https://linkinghub.elsevier.com/retrieve/pii/S135943112301493X (visited on 2024-03-18).

Monnay, Cedric (2017). *Potential and Trends in Off-Highway Vehicle's Electrification*. en. Tech. rep. Gothenburg: Semcon.

Niemelä, Jesper et al. (2023). *Machinopedia—Review of Electric and Hybrid Off-Road Mobile Machines*. Tech. rep. Tampere University: IHA - Innovative hydraulics and Automation lab.

Zimmer, Dirk (2020). "Robust object-oriented formulation of directed thermofluid stream networks". In: *Mathematical and Computer Modelling of Dynamical Systems* 26.3, pp. 204–233. DOI: 10.1080/13873954.2020.1757726. URL: https://doi.org/10.1080/13873954.2020.1757726.

Zimmer, Dirk, Michael Meißner, and Niels Weber (2022). "The DLR ThermoFluid Stream Library". In: *Electronics* 11.22. ISSN: 2079-9292. DOI: 10.3390/electronics11223790. URL: https://www.mdpi.com/2079-9292/11/22/3790.

Physics-Based Dynamic Modeling of Solar-Powered Off-Grid Cold Storage for Perishables Using Modelica: A Case Study – Xingalool, Somalia

Bahareh Bakhsh Zahmatkesh[1*] Mina Shahi[1] Amirhoushang Mahmoudi[1]

[1]Department of Thermal and Fluid Engineering, Faculty of Engineering Technology, University of Twente, The Netherlands,

Bahareh.bakhshzahmatkesh@utwente.nl, m.shahi@utwente.nl, a.mahmoudi@utwente.nl

Abstract

This paper presents a dynamic, physics-based Modelica model for simulating solar-powered, off-grid cold storage systems used to preserve perishables. A preliminary component library, developed using the Modelica Standard Library, supports modular modeling of photovoltaic (PV)-powered chillers, thermal loads, and latent thermal energy storage (LTES), which maintains cooling during non-solar hours. The library is applied to a 200 m³ cold room in Xingalool, Somalia, designed to stay at 5 °C. A case study simulating 500 kg of crop loading at 8:00 and unloading at 17:00 demonstrates the system's dynamic behavior under realistic solar and ambient conditions.

Keywords: Off-Grid, Cold Storage, Digital Twin, Agriculture

1 Introduction

Post-harvest losses of perishable crops remain a pressing challenge, particularly in low- and middle-income countries. In regions such as sub-Saharan Africa and South Asia, up to 50% of harvested fruits and vegetables are lost due to the absence of proper cold storage near farms (Amjad et al. 2023). These losses not only reduce farmer income and worsen food insecurity but also contribute significantly to greenhouse gas emissions (de Souza Garcia et al. 2024).

To mitigate these issues, solar-powered off-grid cold storage systems are increasingly seen as promising solutions for remote agricultural communities (Natarajan et al. 2023). However, their design is inherently complex due to the fluctuating nature of solar energy, variable cooling demands, and the need for reliable energy storage (electrical or thermal) for maintaining the desired temperature in the cold room continuously when the sun light is not present. Dynamic modeling provides a way to calculate system performance under varying conditions and improve design and operation (Purcell et al. 2023). Modelica, a physics-based, equation-oriented modeling language, offers a powerful framework for capturing such multi-domain, dynamic behavior and guiding performance-driven design (Modelica Association 2017).

In this research, a dynamic Modelica model of a solar-PV-powered off-grid cold storage system for perishables is developed for the first time. The system integrates a vapor compression chiller and a latent thermal energy storage (LTES) unit using water/ice to extend cooling during periods without sunlight. The model captures dynamic processes on both demand (e.g., product loading/unloading, wall and infiltration loads) and supply sides (e.g., PV variation, LTES charging/discharging). This work lays the foundation for a digital twin capable of scenario testing, optimization, and real-time analysis.

2 Modelica Modeling Approach

A preliminary component library was developed in Modelica with open-source Modelica Standard Library (MSL) to represent the thermal behavior of solar-powered, off-grid cold storage systems for perishables. The library enables modular and object-oriented construction of system models, simplifying configuration and simulation using reusable components.

In this paper, the library is applied to simulate a representative system in Xingalool, Somalia, consisting of a 200 m³ cold room maintained at 5 °C with time-dependent cooling loads such as product entry and removal, product respiration, wall and infiltration heat gains, and lighting. A vapor compression chiller with a variable-frequency drive (VFD) compressor is powered directly by photovoltaic (PV) panels. The compressor speed adjusts with available solar power, delivering variable cooling capacity. To ensure reliable operation during low or no solar input, a latent thermal energy storage (LTES) unit is incorporated, using roll-bond heat exchangers for energy exchange.

The model focuses on thermal energy transfer through heat ports and omits fluid flow dynamics to maintain computational efficiency. Figure 1 illustrates the three-stage energy flow: during sunlight hours, PV electricity drives the chiller to cool the cold room directly and any surplus power charges the LTES unit; once solar input falls below chiller demand (e.g. at dusk or under cloud), the stored latent thermal energy becomes the sole

cooling source, sustaining the cold-room temperature until PV power resumes.

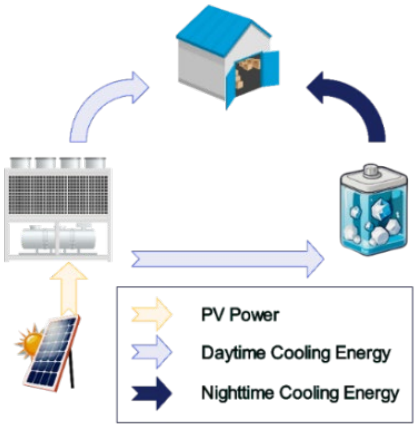


Figure 1. Thermal Model Energy Flow

3 Component Modeling

This section describes the dynamic modeling approach for key components in the cold storage system.

Cold Room and Cooling Loads: The cold room, sized at 200 m³ (4 m × 11 m × 4.6 m), is modeled as a single-node control volume, assuming uniform air temperature throughout, which justifies a lumped capacitance assumption due to well mixing of air in the room. The space is maintained at 5 °C and insulated from direct solar exposure, and is assumed to be filled to 50 % capacity with perishable crops. Cooling demand is dynamically driven by both internal and external loads, including crop loading/unloading, wall heat transfer, infiltration, lighting, and occupant presence.

- Product Sensible Heat Load: To capture the dynamic thermal impact of crop handling, a simplified discretization approach is used. The product mass is divided into two tanks to approximate the first-in-first-out (FIFO) behavior during cooling and storage. Fresh crops enter the first tank at 8:00, exchange heat with the cold room air, and are transferred to the second tank once their temperature converges with that of the second tank. Crops are assumed to be removed at 17:00. This method captures the transient sensible load from crop cooling and removal throughout the day. The logic of the proposed approach is presented in Figure 2 and Equation (1), where Tm_1 and Tm_2 denote the temperatures of the product masses m_1 and m_2 in the first and second tanks, respectively; ε is an infinitesimal tolerance (ε→0).

$$if\ T_{m_1} = T_{m_2} \pm \varepsilon\ \ (\varepsilon \to 0)\ then$$
$$m_1 \to 0; m_2 = m_1 + m_2 \quad (1)$$

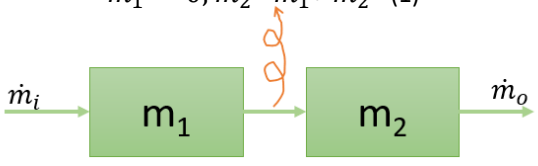


Figure 2. The logic of product sensible heat load model

- Additional Cooling Loads: Time-dependent cooling loads are included to reflect realistic system behavior. Wall heat transfer is modeled using an R-C approach with a U-value of 0.125 W/m²·K, and material properties accounting for thermal mass. Respiration heat from stored crops is computed based on cold room temperature and total product mass dynamically (Blumenthal et al. 2023). Infiltration gains arise from air leakage through a 1.4 m × 2.4 m door and vary with ambient conditions. Occupant heat is modeled using a fixed schedule with two persons entering the room for 2 hours in the morning and afternoon. Lighting is triggered by occupancy and set at 6 W/m².

PV Power Generation: On the supply side, the model accounts for variations in solar irradiance and ambient temperature, which affect PV output. A temperature-dependent efficiency model is used to convert solar input into electrical power, capturing realistic system behavior under varying environmental conditions. (Olukan and Emziane 2014).

Latent Thermal Energy Storage (LTES): To maintain cooling during non-solar hours, a latent thermal energy storage system using roll-bond heat exchangers (RHXs) is integrated. The LTES tank is filled with phase change material (PCM) and connected to a heat transfer fluid (HTF) circuit that charges or discharges the PCM based on energy availability and cooling demand. In the simplified thermal energy-based Modelica model the heat transfer to or from LTES is carried out via a heat port rather than directly involving fluid medium.

Figure 3 shows the LTES tank as a rectangular prism filled with water/ice, into which identical roll-bond heat-exchanger plates are inserted in parallel at 31 mm center-to-center spacing.

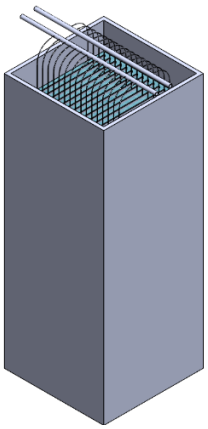


Figure 3. 3D schematic of LTES

Because empirical correlations for natural convection during PCM melting and solidification over a vertical roll-bond heat exchanger are lacking, heat transfer within the PCM has been assumed to occur solely by one-dimensional conduction. A single roll-bond plate plus the half-spacing of PCM is thus defined as the "base unit." A lumped-capacitance assumption for the entire PCM domain in the base unit would introduce large errors due

DOI
10.3384/ecp218939

to steep temperature gradients during phase change; consequently, the PCM thickness of each base unit is discretized into multiple control volumes (Figure 4).

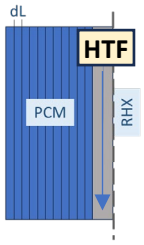


Figure 4: Base element of the LTES model

Within each volume, transient heat storage and transfer are represented by a thermal capacitance and a conduction resistance, whose governing equations are presented in Equations (2)–(7).

PCM Heat Capacity:

$$m.C.\frac{dT}{dt} = Q_{flow} \tag{2}$$

$$C = \begin{cases} C_{ice} & T < T_s \\ \dfrac{h_{sl}}{T_l - T_s} & T_s \le T \le T_l \\ C_w & T > T_l \end{cases} \tag{3}$$

$$f_{liq} = \begin{cases} 0 & T < T_s \\ \dfrac{T - T_s}{T_l - T_s} & T_s \le T \le T_l \\ 1 & T > T_l \end{cases} \tag{4}$$

PCM Heat Transfer:

$$Q_{flow} = G.(T_{hot} - T_{cold}) \tag{5}$$

$$G = \frac{K_{eff}.A}{d} \tag{6}$$

$$K_{eff} = K_{ice}.(1 - f_{liq}) + K_w.f_{liq} \tag{7}$$

In these equations, m is the mass of PCM in each control volume, C the specific heat capacity of the PCM, T the local PCM temperature, Q_{flow} the heat flow into or out of the PCM, f_{liq} the liquid fraction of the PCM, G the thermal conductance between hot and cold faces, A the heat-exchange area, d the conduction path length, K_{eff} the effective thermal conductivity, The solidus and liquidus temperatures, T_s and T_l, are specified as 272.85 K and 273.15 K, respectively. A CFD conduction-only charging simulation was conducted with the roll-bond boundary held at 259.15 K, and the Modelica model was run under the same condition; the validation comparison is presented in Figure 5.

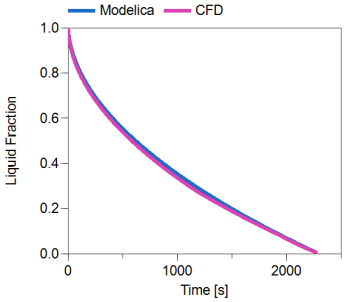


Figure 5: CFD validation results for liquid fraction

4 System Integration and Case Study Results

With all key components developed, the system is now ready for integration under a coordinated control logic. The initial control strategy, shown in Figure 6, distinguishes between normal and cloudy days. On normal days, energy from the PV is first used to cool the cold room. Once the cold room reaches the desired temperature, excess energy is redirected to charge the LTES. On cloudy days, if solar radiation is insufficient to run the chiller, the LTES is discharged to maintain the cold room temperature.

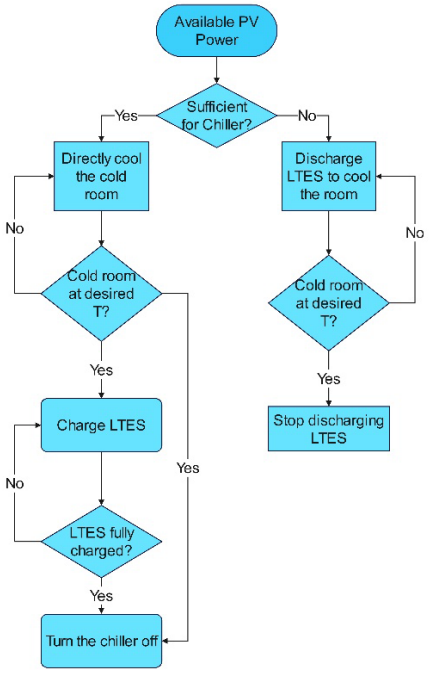


Figure 6: Control Strategy for Normal and Insufficient PV Power Conditions

Representative simulation results are provided for Xingalool, Somalia. Figure 7a shows the cold room temperature during a typical sunny day. The blue line indicates how crop loading in the morning introduces a heat disturbance, prompting the chiller to cool the products. Later, door opening in the afternoon causes another rise in temperature due to infiltration of warm air. The temperature zig-zagging in the cold room is due to the hysteresis mechanism considered for the chiller to avoid chattering and unlimited on/off cycles. To demonstrate the role of LTES in mitigating temperature fluctuations, two consecutive cloudy days were simulated (Fig. 7b), showing how the LTES supports temperature regulation when solar power is unavailable. It can be seen that LTES with 20 roll-bond heat exchangers is not capable of maintaining the cold room temperature under normal loading conditions in two consecutive cloudy days. Hence, in case this occasion is going to be repeated, either the LTES capacity should increase or loading behavior should change when sun light is absent.

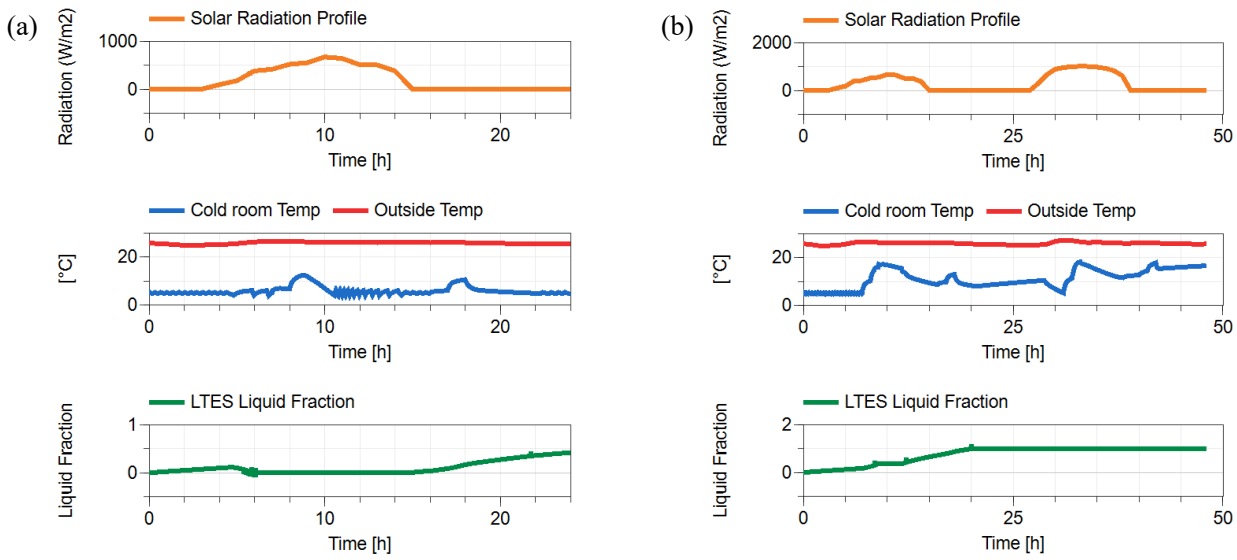


Figure 6: Represantative results for (a) normal sunny day, (b) two consecutive cloudy days

5 Conclusion

This work presents a dynamic, physics-based Modelica model for simulating solar-powered, off-grid cold storage systems for perishables. By leveraging Modelica's object-oriented, equation-based framework, a modular component library was developed that can be readily extended or reconfigured for alternative system variants, such as different PCM geometries, chiller types, or control strategies, without rewriting core equations. The library was applied to a 200 m³ case study in Xingalool, Somalia, demonstrating accurate capture of time-dependent thermal behavior under realistic solar and load profiles. These results underscore Modelica's strengths in rapid prototyping, and parameterized system variation, for sustainable cold-storage design and optimization..

Acknowledgements

This research was funded by European Union through the Horizon Europe research grant n. 101147102, "AGRI-COOL: Advancing Sustainable Agriculture through Off-Grid Energy and Cooling Solutions in Africa". Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or CINEA. Neither the European Union nor the granting authority can be held responsible for them.

References

Amjad, Waqas, Aftab Munir, Fatima Akram, Akshay Parmar, Mathilde Precoppe, Farooq Asghar, and Faisal Mahmood (2023). "Decentralized solar-powered cooling systems for fresh fruit and vegetables to reduce post-harvest losses in developing regions: A review". In: Clean Energy 7, pp. 635–653. DOI: 10.1093/ce/zkad015.

Blumenthal, K., J. Evans, L. Kitinoja, S. McCarney, K. Sasidharan, J. A. Verschoor, J. Vrba, G. Cortella (Ed.), and J. Tait (Ed.) (2023). Walk-in cold rooms, a practitioner's technical guide: Design and operation of walk-in cold rooms for precooling and storage of fresh produce in hot climates, in off-grid and unreliable grid situations. Efficiency for Access. URL: https://efficiencyforaccess.org/wp-content/uploads/IIFIIR-Livre-Cold-storage-_modif-efficiency-AD_clic.pdf.

de Souza Garcia, Eduardo, Nuno Quaresma, Yibeltal B. Aemro, António P. Coimbra, and Aníbal T. de Almeida (2024). "Cooling with the sun: Empowering off-grid communities in developing countries with solar-powered cold storage systems". In: Energy Research & Social Science 117. DOI: 10.1016/j.erss.2024.103686.

Natarajan, Balamurugan, Chellachi Kathiresan, and Subramanium S.K. (2023). "Development and performance evaluation of a hybrid portable solar cold storage system for the preservation of vegetables and fruits in remote areas". In: Journal of Energy Storage 72. DOI: 10.1016/j.est.2023.108292.

Modelica Association (2017-04). Modelica – A Unified Object-Oriented Language for Systems Modeling. Specification Version 3.4. Tech. rep. Linköping: Modelica Association.

Olukan, T.A. and M. Emziane (2014). "A comparative analysis of PV module temperature models". In: Energy Procedia, Elsevier Ltd, pp. 694–703. DOI: 10.1016/j.egypro.2014.12.433.

Purcell, W., and Neubauer, T. (2023). "Digital Twins in Agriculture: A State-of-the-art review". In: Smart Agricultural Technology 3. DOI: 10.1016/j.atech.2022.100094.

Selective Evaluation of RHS during Multi-Rate Simulation

Philip Hannebohm [1] Bernhard Bachmann [1]

[1]Institute for Data Science Solutions, Bielefeld University of Applied Sciences and Arts, Germany,
philip.hannebohm@hsbi.de, bernhard.bachmann@hsbi.de

Abstract

Modeling across physical domains can lead to coupled systems that operate on different time scales. Moreover, spatially discretized models can have dynamic behavior on only a small portion of the whole system, while that portion might move over time. With a single rate solver like DASSL, the step size is restricted to fast dynamics of a few states while the others could do much larger time steps, degrading the overall simulation speed. Recent development of the Generic Bi-rate ODE (GBODE) solver within the OpenModelica simulation environment addresses these models. It allows for adaptive partitioning of slow and fast states and progresses these partitions with different time steps. However, substantial performance improvements were not yet observed, in part because of the rigid evaluation structure of the generated code. During each small step only the derivatives of fast states are required, but all derivatives are computed. This paper demonstrates the use of selective equation evaluation to reduce the computational cost of multi-rate integration as shown on a scalable example of a distributed heating system. Further uses for selective evaluation during the simulation process are discussed and a complexity analysis is given.

Keywords: Equation-based modelling, Multi-rate simulation, Structural analysis

1 Introduction

In computer science strategies like partial or lazy evaluation are used to improve computation speeds. This paper introduces a similar approach for use in the simulation of equation-based models. These models are commonly represented as one coupled differential algebraic equation system (DAE), so for the purposes of this paper we consider the equation

$$0 = F(x, \dot{x}, u) \tag{1}$$

which is solved for $\dot{x}$ at each integration step assuming knowledge of all the states x. Further inputs u may influence the system dynamics as well.

Multi-rate integration methods are suited for systems with dynamics on multiple time scales (Thiele, Otter, and Matsson 2014). Furthermore, for models where a subsystem A undergoes rapid changes while another subsystem B stays relatively stationary during some interval of the simulation time, at a later time B may have dynamic changes while A does not. In these scenarios adaptive multi-rate simulation is clearly superior to statically partitioned multi-rate or even single-rate simulations. One such adaptive multi-rate solver is GBODE which stands for Generic Bi-rate ordinary differential equation (ODE) solver and is a generic implementation for any Runge-Kutta (RK) scheme (Bachmann 2023; Bonaventura et al. 2025). As a case study, our method is applied to the GBODE solver of OpenModelica.

During a multi-rate simulation step a subset of fast states is identified. For these states several small steps are taken while the remaining states take only one large step. At each fast step only a subset of $\dot{x}$ needs to be computed. Thus, computing the whole right-hand side (RHS) means superfluous calculations.

For efficiently evaluating parts of a model without evaluating the whole RHS, it is necessary to have a fast and flexible way to evaluate only a selected subset of equations. We demonstrate how this can be easily done within the OpenModelica Compiler (Fritzson, Pop, Abdelhak, et al. 2020). For the relatively newly developed GBODE solver this feature is crucial in providing improved simulation performance compared to other integration schemes.

2 Evaluation Structure

This section builds up the notation used in this paper. A general system of equations can be given in residual form as

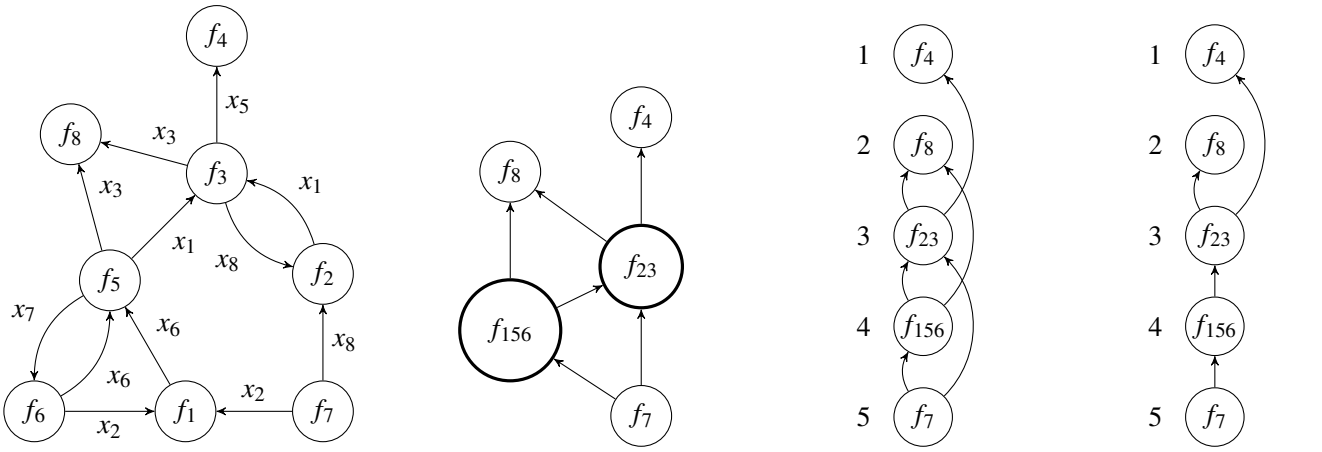
$$\begin{aligned} 0 &= f_1(x_{I_1(1)}, \ldots, x_{I_1(d_1)}) \\ &\vdots \\ 0 &= f_n(x_{I_n(1)}, \ldots, x_{I_n(d_n)}) \end{aligned} \tag{2}$$

where d_k is the number of variables occurring in equation k and $I_k(1), \ldots, I_k(d_k)$ are their corresponding indices.

If this system is given symbolically, or the incidence is given explicitly by the $I_1, \ldots, I_n$, Equation (2) can be transformed into semi-explicit form. The OpenModelica Compiler does this via the following steps.

A *perfect matching* between variables and equations is found (Pantelides 1988). The result can be interpreted as a directed graph with matched pairs of variables and equations as nodes and edges from equations to variables used in the equation.

In the context of this paper it is advantageous to define the edges in the opposite direction such that $f_i \to f_j$ when f_i depends on f_j, instead of the usual definition where we would have $f_j \to f_i$ as f_j influences f_i.



(a) Directed graph based on the given matching.

(b) DAG constructed by contracting the SCCs $f_{23} = \{f_2, f_3\}$ and $f_{156} = \{f_1, f_5, f_6\}$.

(c) Topologically sorted DAG.

(d) Transitive reduction of the DAG in (c).

Figure 1. Structure of the system in Equation (3).

As an example take the following system of equations and assume the equations are matched to the underlined variables:

$$\begin{aligned} 0 &= f_1(\underline{x_2}, x_6) & 0 &= f_5(x_1, x_3, \underline{x_6}, x_7) \\ 0 &= f_2(x_1, \underline{x_8}) & 0 &= f_6(x_2, x_6, \underline{x_7}) \\ 0 &= f_3(\underline{x_1}, x_3, x_5, x_8) & 0 &= f_7(x_2, \underline{x_4}, x_8) \\ 0 &= f_4(\underline{x_5}) & 0 &= f_8(\underline{x_3}) \end{aligned} \tag{3}$$

Figure 1a shows the graph resulting from Equation (3).

All *strongly connected components* (SCCs) of the graph are contracted into single nodes resulting in a *directed acyclic graph* (DAG). This DAG is the basis for evaluating the RHS of any ODE. The nodes of a DAG can be *topologically sorted*, i.e. they can be labeled with integer indices in such a way that $i > j$ for every edge $i \to j$. One such choice of indices is shown in Figure 1c. A valid order of execution is obtained by progressing from nodes with lower index to nodes with higher index. These two steps, contracting SCCs and topological sorting, are done simultaneously (Tarjan 1972).

There is a map $\mathscr{M}$ from variable to index corresponding to the node that computes the variable. This can be seen as a modification of the matching where variables solved in the same SCC are given the same index. For the system of Equation (3) this would be

$$\begin{aligned} &\mathscr{M}(x_1) = \mathscr{M}(x_8) = 3 \\ &\mathscr{M}(x_2) = \mathscr{M}(x_6) = \mathscr{M}(x_7) = 4 \end{aligned} \qquad \begin{aligned} &\mathscr{M}(x_3) = 2 \\ &\mathscr{M}(x_4) = 5 \\ &\mathscr{M}(x_5) = 1 \end{aligned} \tag{4}$$

Similarly, a modified incidence map $\mathscr{I}$ from index to variable sets can be constructed based on the I_k as defined in Equation (2) and the contraction of SCCs. The variables solved in the respective SCC are removed so only its dependencies remain. For Equation (3) this would be

$$\begin{aligned} &\mathscr{I}(1) = \mathscr{I}(2) = \{\} && \mathscr{I}(4) = \{x_1, x_3\} \\ &\mathscr{I}(3) = \{x_3, x_5\} && \mathscr{I}(5) = \{x_2, x_8\} \end{aligned} \tag{5}$$

Applying $\mathscr{M}$ to each element of $\mathscr{I}(k)$, we get the direct dependencies of the SCC with index k. This corresponds to the direct children of node k. In general we get the direct dependency map $\mathscr{D}$ as

$$\mathscr{D}(k) = \{\mathscr{M}(x) \mid x \in \mathscr{I}(k)\}. \tag{6}$$

Again, for Equation (3) the direct dependencies are

$$\begin{aligned} &\mathscr{D}(1) = \mathscr{D}(2) = \{\} && \mathscr{D}(4) = \{3, 2\} \\ &\mathscr{D}(3) = \{2, 1\} && \mathscr{D}(5) = \{4, 3\} \end{aligned} \tag{7}$$

Note that the indices in $\mathscr{D}(k)$ are all smaller than k. This is by construction of the topological ordering.

3 Selective Evaluation

The goal is to only evaluate the minimal part of the system that is needed to compute values for a set X of selected variables.

3.1 Computing Reachability

Finding all dependencies of X is equivalent to computing the reachability of the corresponding nodes in the DAG. Since the DAG is topologically sorted, the reachability can be found without the need for depth-first search. This is done in Algorithm 1 by going through the nodes from last to first according to the topological order and on each selected node k marking its direct dependencies $\mathscr{D}(k)$ as selected. All nodes in $\mathscr{D}(k)$ are smaller than k and their direct dependencies will be selected once the outer loop in line 3 gets to them. In this way the deeper paths are marked automatically by later loop iterations. The

DOI 10.3384/ecp218943

Algorithm 1 mark reachability by backwards loop

1: **for** $x \in X$ **do**
2: select $\mathscr{M}(x)$
3: **for** $k = N, \ldots, 1$ **do** ▷ backwards
4: **if** k is selected **then**
5: select all $j \in \mathscr{D}(k)$ ▷ $j < k$

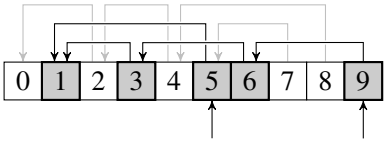


Figure 2. Example of reachability computation exploiting the topological ordering of the DAG.

complexity of Algorithm 1 is $O(|X| + |V| + |E|)$ where $|E| = \sum_k \mathscr{D}(k)$ is the number of edges in the DAG and $|V| = N$ is the number of vertices.

This is demonstrated in Figure 2 with a small example. First, nodes 5 and 9 corresponding to fast states are selected, then direct dependencies of selected nodes are selected going from right to left, so the order of selection is: $9 \to 6$, $6 \to 3$, $5 \to 1$, $3 \to 1$.

3.2 Transitive Reduction

The *transitive reduction* R of a DAG G is a subgraph of G which has the same reachability relation but minimal number of edges (Aho, Garey, and Ullman 1972). If there is a longer path from i to j then a direct edge $i \to j$ can be removed, keeping the overall reachability the same. Figure 3 shows the simplest scenario where the edge from 3 to 1 can be removed because dependency is still given through 2. The cost of computing R is $O(|V||E|)$ for sparse graphs (Goralčíková and Koubek 1979).

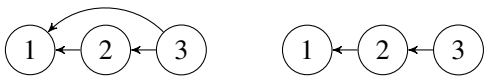


Figure 3. One step of transitive reduction.

Using R instead of G we can reduce the amount of work in line 5 of Algorithm 1 because fewer nodes are in direct dependencies. Since R needs to be computed only once in a preprocessing step, its cost is negligible when doing a reachability analysis many times during simulation. In highly connected systems this might improve the overall performance significantly. For the example system in Equation (3) this removes two out of six edges as depicted in Figure 1d.

3.3 Implementation

Prior to our changes the simulation code generated by the OpenModelica Compiler had the following flat structure:

```
void functionODE()
{
  eqFunction_1();
  // [...]
  eqFunction_N();
}
```

While very simple and efficient, the code does not allow for partial evaluation. One way to achieve this is by giving each `eqFunction_i` a flag to tell whether or not it should be evaluated. However, this would still lead to $O(N)$ operations. Replacing the function with the following evaluation scheme gives the desired flexibility with minimal overhead:

```
void functionODE()
{
  static f_ptr eqFunctions[N] = {
    eqFunction_1,
    // [...]
    eqFunction_N
  };
  for (int i = 0; i < evalN; i++)
    eqFunctions[evalI[i]]();
}
```

Here `evalI` is an array of SCC indices and `evalN` is the length of `evalI`, whereas `eqFunctions` is an array of function pointers `f_ptr`.

During simulation, `evalI` needs to be recomputed via Algorithm 1 each time X changes. If there is a rapid swap between two unchanging sets X_s and X_f, as is the case for GBODE bi-rate integration, the corresponding `evalI` can be buffered to avoid recomputation.

Some additional overhead can be expected from making equation evaluation flexible in this way because of the additional pointer indirection and as a consequence the less efficient cache handling. However, this overhead should be negligible for large systems when only a fraction of equations needs to be evaluated.

4 Preliminary Results

A modified version of GBODE was tested on the model `HeatingSystem` from the ScalableTestSuite Modelica Library (Casella 2015). It has the following equations:

```
Text = 278.15 + 8*sin(2*pi*time/86400)"f1";
Cd*der(Td) = Qd - sum(Qh)             "f2";
Qd = sat(Kp*(Td0 - Td),0, Qmax)       "f3";
for i in 1:N loop
  Qh[i] = Gh*(Td - Tu[i])*u[i]        "f4";
  Que[i] = Gu*(Tu[i] - Text)          "f5";
  Cu[i]*der(Tu[i]) = Qh[i] - Que[i]   "f6";
  der(x[i]) = a*hist(x[i], Tu0 - Tu[i],
                     Teps)            "f7";
  u[i] = sat(b*x[i], -0.5, 0.5) + 0.5 "f8";
end for;
```

Its equation structure, depicted in Figure 4, is well suited to selective evaluation when solving for fast states. There are no algebraic loops involved and thus every variable can be computed explicitly. Because of this, each node corresponds exactly to one variable.

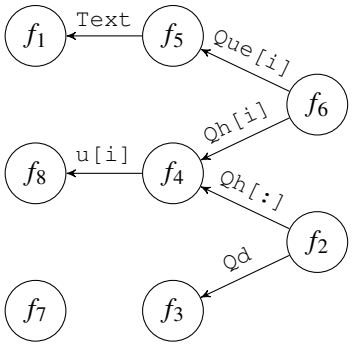


Figure 4. DAG of `HeatingSystem`. Arrays and for-loops are kept as single nodes for better readability.

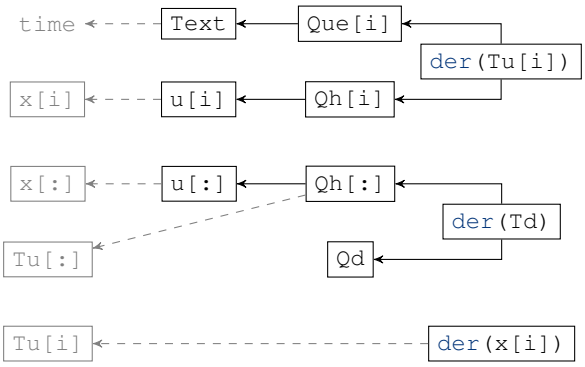


Figure 5. Sub-DAGs of `HeatingSystem` for each state derivative. In dashed are dependencies on states that may need to be interpolated. Self-dependencies are omitted for readability.

The various sub-DAGs for computing single derivatives are shown in Figure 5. Here variables are used as names for the nodes to make the structure easier to recognize. There is some overlap between the dependencies of each of the `der(Tu[i])` and `der(Td)` due to `Qh`.

We performed a simulation of `HeatingSystem` with the solver settings `-s=gbode -gbratio=0.5`. Each time a new set of fast states was identified, the fraction of equations needed to evaluate the derivatives was determined by Algorithm 1 and logged. The top row of Table 1 shows the number of equations to evaluate. The bottom row shows how often these many equations were needed to evaluate the new set of fast states. In most cases the number of equations needed is less than 7 out of a total of 64, which is about 11%. More than half of the time only two equations (3%) were needed, while the number never went beyond 17 (27%).

Table 1. Number of phases that require evaluation of `evalN` equations for fast states. Entries not in the table have a count of zero.

`evalN`	2	4	6	13	15	17
#	365	184	75	4	3	3

5 Conclusions and Future Work

We showed that selectively computing parts of the equation system of a DAE is practically possible and most of the work needed is already done by Modelica compilers. We also demonstrated that it would be desirable for multi-rate simulation of at least some models. Selective evaluation is also used to compute only needed parts of the Jacobian when solving fast states with implicit methods. In particular for the `HeatingSystem` there are significant savings in computational load. This is not a direct measure of performance gain, but it gives a rough idea that selective evaluation should lead to significantly faster simulations.

The technique of selective RHS-evaluation and the corresponding dependency analysis can be used even more broadly. During event search of hybrid DAE simulations only the zero crossings need to be computed in order to narrow down the exact event time. Then during event iteration only parts influenced by discrete changes need to be recomputed iteratively. For this the dependency analysis needs to be extended on discrete variables and a slightly different selection mechanism needs to be developed. This will be investigated in future works.

Acknowledgements

This work was conducted as part of the OpenSCALING project (Grant No. 01IS23062E) at the University of Applied Sciences and Arts Bielefeld, in collaboration with Linköping University. The authors would like to express their sincere appreciation to both the OpenSCALING project and the Open Source Modelica Consortium (OSMC) for their support, collaboration, and shared commitment to advancing open-source modeling and simulation technologies.

References

Aho, A. V., M. R. Garey, and J. D. Ullman (1972). "The Transitive Reduction of a Directed Graph". In: *SIAM Journal on Computing* 1.2, pp. 131–137. DOI: 10.1137/0201008.

Bachmann, Bernhard (2023). "GBODE - The Generic Bi-Rate Ordinary Differential Equation Solver in OpenModelica". In: *OpenModelica Annual Workshop*. Linköping, Sweden. URL: https : / / openmodelica . org / images / M _ images / OpenModelicaWorkshop _ 2023 / GBODE - OpenModelicaWorkshop_2023.pdf.

Bonaventura, Luca et al. (2025). "Self-Adjusting Multi-Rate Runge-Kutta Methods: Analysis and Efficient Implementation in An Open Source Framework". In: *Journal of Scientific Computing* 105. DOI: 10.1007/s10915-025-03049-y.

Casella, Francesco (2015). "Simulation of Large-Scale Models in Modelica: State of the Art and Future Perspectives". In: *Proceedings of the 11th International Modelica Conference*. Versailles, France, pp. 459–468. DOI: 10.3384/ecp15118459.

Fritzson, Peter, Adrian Pop, Karim Abdelhak, et al. (2020-10). "The OpenModelica Integrated Environment for Modeling, Simulation, and Model-Based Development". In: *Modeling, Identification and Control: A Norwegian Research Bulletin* 41, pp. 241–295. DOI: 10.4173/mic.2020.4.1.

Goralčíková, Alla and Václav Koubek (1979). “A reduct-and-closure algorithm for graphs”. In: *Mathematical Foundations of Computer Science 1979*. Ed. by Jiří Bečvář. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 301–307. DOI: 10.1007/3-540-09526-8_27.

Pantelides, Constantinos C. (1988). “The Consistent Initialization of Differential-Algebraic Systems”. In: *SIAM Journal on Scientific and Statistical Computing* 9.2, pp. 213–231. DOI: 10.1137/0909014.

Tarjan, Robert (1972). “Depth-First Search and Linear Graph Algorithms”. In: *SIAM Journal on Computing* 1.2, pp. 146–160. DOI: 10.1137/0201010.

Thiele, Bernhard, Martin Otter, and Sven Erik Matsson (2014). “Modular Multi-Rate and Multi-Method Real-Time Simulation”. In: *Proceedings of the 10th International Modelica Conference*. Lund, Sweden, pp. 381–393. DOI: 10.3384/ecp14096381.

DOI
10.3384/ecp218943

Absolut Modelica library

Carles Ribas Tugores[1] Gerald Zotter[1] Carina Seidnitzer-Gallien[1]

[1]AEE - Institute for Sustainable Technologies, Feldgasse 19, Gleisdorf, 8200, Austria,
{c.ribastugores,g.zotter,c.seidnitzer-gallien}@aee.at

Abstract

This publication presents a Modelica library developed to investigate absorption thermodynamic cycles, with special attention to the absorption heat exchanger (AHE). The library includes models for different absorption cycles at various levels of detail, most of which have been validated against literature values. The AHE concept and the main components used to develop a dynamic AHE model, together with validation results based on laboratory measurements, are described in detail. The library is available at `https://github.com/carlesRT/Absolut`.

Keywords: Modelica library, Absorption thermodynamic cycles, Absorption heat exchanger, H_2O/LiBr

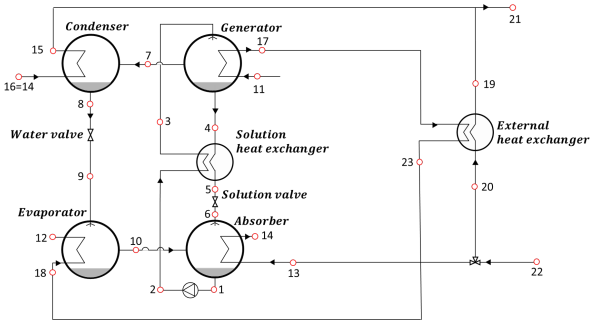


Figure 1. Schematics of an absorption heat exchanger. Numbers indicate the positions of referenced variables.

1 Introduction

1.1 The absorption heat exchanger

The absorption heat exchanger (AHE) concept involves the integration of an absorption heat pump (AHP) with a conventional heat exchanger (HEX), see Figure 1. The main advantage of this configuration is the lower return temperatures achieved on the primary (hot) side compared to standard solutions, such as water-water counterflow heat exchangers. The AHE concept was originally introduced and discussed in (Fu et al. 2010), (Li et al. 2011) and (Yin Zhang, Shi, and Yinping Zhang 2014).

The primary supply stream at temperature T_{11} first passes through the generator, where it provides heat to produce the refrigerant vapor. It then enters the external heat exchanger, and subsequently flows into the evaporator, where it can be cooled to a temperature lower than the secondary return temperature ($T_{12} < T_{22}$). On the secondary side, the return stream is heated by the external heat exchanger, the absorber, and the condenser. Various configurations are possible (e.g. absorber, condenser and HEX arranged in parallel). In the configuration shown in Figure 1, the secondary stream is split into two branches; one flows through the external heat exchanger, while the other passes through the absorber and then the condenser.

The AHE concept is applied mainly in China, with a notable project located in Taiyuan-Gujiao, China (Hu, Jiang, and Xie 2009). In this project, an AHE is integrated into a 50 km long district heating line connecting a CHP plant to the city center to increase heat transport capacity. Through this integration, the system can deliver 3.2 GW of heat at 130°C/20°C, demonstrating the significant reduction in return temperature achievable with this technology. Interest in the AHE concept within the district heating sector is also growing in Europe (G. Beckmann, F. Schittl, G. Piringer, D. Rixrath, R. Krotil, J. Krail 2019; Yang, Mirl, and Schmid 2020; Zotter, Eberhöfer, and Seidnitzer-Gallien 2023).

1.2 Open-Source Modelica models

The modeling language Modelica (Modelica Association n.d.) was selected to develop the necessary models for conducting various studies within the Absolut project and upcoming studies on future energy concepts within the TREASURE[1] project.

Before initiating the modeling process, a literature review was conducted, which included testing available open-source models, to assess their suitability for the planned work. Commercial libraries such as TIL (*TIL Suite* n.d.) and in-house models (e.g. (Cudok 2021; Corrales Ciganda et al. 2016; De La Calle et al. 2016; Wernhart et al. 2022)) were excluded from this review. The review identified three primary libraries. The key findings are summarized in Appendix A.

The review of open-source models for absorption machines yields the following conclusions.

- Compatibility with other Modelica libraries: The use of standardized interfaces (e.g. `fluidPort`) enables a modeling approach that leverages validated models from multiple sources and facilitate interoperability. This approach has been adopted by the authors to enhance the usability of their models within the broader Modelica ecosystem.

[1]TREASURE project: www.treasure-project.eu

- Object-oriented modeling approach: The DCOL library (Febres et al. 2017) adopts an object-oriented approach, with separate models for key components (e.g. absorber, generator), which supports modularity, ease of development, and maintainability. The same approach is followed in this work, with DCOL models serving as a reference for the implementation of the dynamic models.
- Initialization Challenges: Some evaluated models encountered difficulties during initialization and execution, highlighting the importance of a robust model structure and well-defined initial values to improve usability. The models developed in this work include runnable examples, which also serve as a reference for troubleshooting newly created or modified models.
- Fluid Property Implementation: Although fluid property correlations are available, inconsistencies in their implementation may lead to numerical instabilities. Existing libraries would benefit from a more consistent and reliable representation of fluid properties.

1.3 The Absolut library

A wide range of models at varying levels of complexity has been implemented. Given the number of Modelica models publicly available compared to those described in the literature (see section 1.2), we believe that the open-source community can benefit from this contribution. The provided library offers a reliable set of simplified models suitable for case studies or as a foundation for further development. However, it should be noted that the implementation work was primarily focused on the dynamic modeling of the absorption heat exchanger (AHE).

The Absolut library includes models for absorption heat pumps at three main levels of detail: basic (zero-order), static, and dynamic models. The left side of Figure 2 provides a view of the main package structure. Most of the models have been validated using examples from (Herold, Radermacher, and Klein 2016). The static and dynamic models use H_2O/LiBr as the working pair, for which thermophysical correlations have been implemented (see section 2.2). The use of alternative working pairs is subject to two main conditions: first, the compatibility between the models and the implemented fluid property correlations; and second, the validity of the model assumptions (e.g., the assumption of a pure refrigerant at the generator outlet). Mention that some modifications will be most likely required due to the use of some non-standard functions such as `temperature_Xp()` on the mixture side (H_2O/LiBr). These can be addressed by adding missing functions to the new working pair package or better adapting the models to use more standardized functions, e.g. `temperature(setDewState())`. Similar compatibility issues might also arise on components using pure-refrigerant.

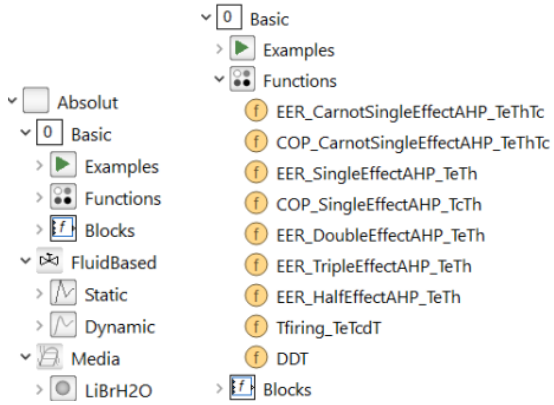


Figure 2. Overview of main packages structure (left) and Basic package (right).

Usability has been confirmed primarily in Dymola (*Dymola* 2024), with partial testing performed in OpenModelica (*OpenModelica* 2024) to ensure usability with purely open-source environment.

2 Description of models

2.1 Ideal models

The `Basic` package contains models based on an idealized representation of the thermodynamic cycle for absorption machines. The core models are implemented as functions, which are encapsulated in blocks with input/output interfaces to facilitate their use via the graphical user interface. All the available functions are shown in the right side of Figure 2. The application of these functions and blocks is demonstrated through several examples.

The provided examples are adapted from (Herold, Radermacher, and Klein 2016) and are used to verify the correctness of the implementation. In addition, `.mos` scripts are included with each example to facilitate automated simulation and visualization of results.

Most of the models included in the `Basic` package are based on the Carnot cycle (reversible process). By combining the Carnot cycle for power generation with a refrigeration cycle, the efficiency of a single effect absorption heat pump can be derived. This principle forms the basis of the functions `EER_CarnotSingleEffectAHP_TeThTc` and `COP_CarnotSingleEffectAHP_TeThTc`, as well as their corresponding blocks. The Carnot-based functions are further refined to account for the primary source of irreversibility in absorption cycles, which is associated with heat transfer between the cycle and its surroundings (Herold, Radermacher, and Klein 2016). This refinement is achieved by introducing a thermal resistance between the external and internal temperatures, resulting in a temperature difference. It is important to note that the internal cycle of the AHP is still assumed to be reversible. This assumption, along with the assumption that the tempera-

DOI
10.3384/ecp218949

ture difference between the condenser and the generator is equal to that between the condenser and the evaporator, forms the basis of the so-called zero-order model (Herold, Radermacher, and Klein 2016). This models is used to derive the functions `EER_SingleEffectAHP_TeTh` and `COP_SingleEffectAHP_TeTh` to calculate the cooling and heating efficiency respectively of a single effect absorption heat pump. Similar functions are implemented to calculate the cooling efficiency of double effect, triple effect and half effect absorption heat pumps.

The main practical application of the basic models is to quickly estimate the expected efficiency - and thereby the feasibility - of an absorption cycle for given operating temperatures. Conversely, they can be used to estimate the required driving temperature needed to deliver heat at a target temperature (e.g. `Absolut.Basic.Blocks.Tfiring`). Notice that the use of input/output interfaces is in line with the intended use of the models. However, this approach restricts the use of the blocks, as, for example, `EER_CarnotSingleEffectAHP` cannot be used to determine the driving temperature `Th` given the desired efficiency (`EER`) and the temperatures `Tc` and `Te`. Nevertheless, this result can be obtained by using the function `EER_CarnotSingleEffectAHP_TeThTc` directly, as both Dymola and OpenModelica are capable of interpreting and manipulating the function to solve for the input temperature `Th`.

2.2 Fluid properties

The static and dynamic models use pure water as a refrigerant, specifically employing the implementation based on the IAPWS/IF97 standard provided in the Modelica Standard Library (MSL) (Modelica Association 2024). The thermodynamic correlation for the working pair H_2O/LiBr is based on the formulation from (Yuan and Herold 2005) and has been implemented in Modelica.

This correlation follows a global approach to ensure improved consistency across properties. Rather than fitting individual thermodynamic properties separately, a fundamental function - in this case Gibbs free energy - is selected. All relevant properties are derived from this function using thermodynamic relationships (e.g. entropy is obtained from the partial derivative with respect to the temperature of the Gibbs free energy). The correlation expresses the Gibbs free energy as a function of temperature, pressure, and lithium bromide mass fraction. All available thermodynamic data are then used simultaneously to determine the main coefficients of the fundamental function. The correlation obtained gives good accuracy over the full range of liquid concentrations from pure water up to the crystallization line and from 5°C to 250°C (Yuan and Herold 2005). Examples verifying the implementation are included in the `Media.LiBrH2O.Validation` package.

2.3 Static models

The `...FluidBased.Static` package contains models for components such as valves, pumps, primary heat exchangers (absorber, generator, evaporator, and condenser) and solution heat exchangers. These component models can be combined to assemble various thermodynamic absorption cycles.

Several cycle configurations have already been implemented, including single effect and double effect absorption heat pumps (AHP), absorption heat transformer (Type II AHP), absorption heat exchanger (AHE) and resorption cycle. In some cases, multiple variants of a given cycle are provided, for example AHE model with the absorber and condenser arranged in parallel, and a version with all flows in parallel. These models can be used to roughly assess the required size, or better said the required UA value, for a given application (temperatures, heating capacity, configuration), while disregarding process dynamics. As a result, their use is limited when analyzing control strategies.

Most of the models implemented have been verified using examples from (Herold, Radermacher, and Klein 2016). The corresponding verification models are included in the respective `Validation` packages. Some results are included in Appendix B.

2.3.1 Heat exchangers

A detailed description of the main heat exchanger models (evaporator, etc.) is omitted here, and only a few remarks are provided on the available models.

All primary heat exchangers, and consequently the thermodynamic absorption cycles, are modeled using the standard `fluidPort` interface for external fluid connections. This allows the models to be integrated with their respective heat sources and sinks. However, for the single effect absorption heat pump, simplified versions of the models are also provided without a heat exchanger (`Static.SingleEffect_intern`). These omit the external fluid interfaces and are intended for simulations under prescribed operating conditions (e.g. given pressure and LiBr concentration at the generator and absorber) that can be used to easily evaluate a prescribed thermodynamic cycle and to provide reference values for initializing more complex models.

Models that consider the heat exchanger and have `fluidPort` interfaces calculate the total heat transfer using the commonly applied simplified method, which is based on the product of the overall heat transfer coefficient (UA) and the logarithmic mean temperature difference (LMTD), as described in (Herold, Radermacher, and Klein 2016). As the calculation of LMTD might be problematic, the `NoEvent` operator is included to avoid division by zero. Furthermore, the approximation of the LMTD proposed by Chen (Chen 1987) is also implemented, see `...Static.BaseClasses`. As reported in Petterson (Pettersson 2008) the deviation between the LMTD and Chen expression is very small. Either the LMTD method or Chen's approximation can be selected,

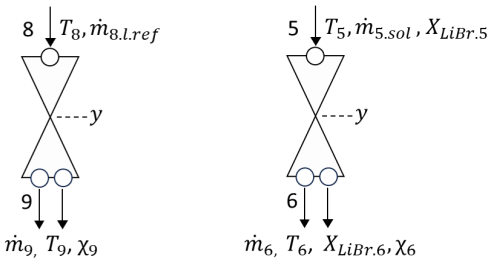


Figure 3. Schematics of the valve model: Refrigerant side (left) and solution side (right).

with LMTD used as the default.

Some relevant limitations and weak points of the current implementation, in addition to the fact that the models are static, includes the use of constant UA values (defined as parameters) and the limited use of `extends` clause, which leads to code duplication and consequently increases the risk of errors and maintenance effort.

Additionally, three heat exchanger models from the Buildings library, `PartialEffectivenessNTU`, `PlateHeatExchangerEffectivenessNTU` and `ConstantEffectiveness`, have been incorporated into the library using the `extends` clause (see package `...Static.Components.HEX`). An additional parameter has been introduced to later facilitate model substitution via the `redeclare` keyword. In the context of the static models, these models are intended to be used to model the solution heat exchanger. The simplest ones are `simpleHX`, which `extends PartialEffectivenessNTU` and fixes the UA value, and `ConstantEffectiveness` which assumes a constant effectiveness. `PlateHeatExchangerEffectivenessNTU` is a more complex model with a variable UA value.

2.3.2 Valves

Both the solution valve and the refrigerant valve are modeled assuming isenthalpic expansion and with no mass accumulation. The incoming fluid may undergo flashing, therefore the valve model calculates the outlet vapor quality, χ, defined as the mass fraction of vapor at the outlet. State points are numbered for ease specific location in the system, see Figure 3. Additionally, subscripts are used to further specify variables, as detailed in the nomenclature section. Subscripts are separated by a dot. For flow related variables (e.g. mass flow rate), a positive value indicates inflow into the component, while a negative value indicates outflow. This sign convention has already been incorporated into the governing equations, consequently, flow variables should be considered in terms of their absolute values. The common mass and energy balance equations for the refrigerant and solution valve are:

$$0 = \dot{m}_j - \dot{m}_k \quad \text{where } (j,k) \in \{(8,9),(5,6)\} \tag{1}$$

$$\chi_j = \frac{\dot{m}_{j.v}}{\dot{m}_j} \quad \text{where } j \in \{9,6\} \tag{2}$$

$$h_j = (1-\chi_j)\cdot h_{j.l} + \chi_j \cdot h_{j.v} \quad \text{where } j \in \{9,6\} \tag{3}$$

An additional mass balance equation for lithium bromide is required for the solution valve,

$$0 = \dot{m}_{5.l.sol} \cdot X_{LiBr.5} - \dot{m}_{6.l.sol} \cdot X_{LiBr.6} \tag{4}$$

The mass flow rate is determined by Equations (5) and (6). It is modeled as being proportional to the valve opening signal y, the pressure difference across the valve, and a constant based on nominal operating conditions. Specifically, the nominal mass flow rate $\dot{m}_{nom}$ is defined for a fully open valve ($y = 1$) and a specified nominal pressure difference dp_{nom}. The pressure levels correspond to the pressures in the respective vessels. For example, in the case of the refrigerant valve, the pressure difference is calculated using the pressure in the condenser p_c (position 8) and the pressure in the evaporator p_e (position 9).

$$\dot{m}_{8.l} = y \cdot (p_c - p_e) \cdot \frac{\dot{m}_{nom}}{dp_{nom}} \tag{5}$$

$$\dot{m}_{5.l} = y \cdot (p_g - p_a) \cdot \frac{\dot{m}_{nom}}{dp_{nom}} \tag{6}$$

2.3.3 Pumps

Equation (7) used to obtain the electrical power consumption of the solution pump W_{el} is obtained assuming incompressible flow and isentropic compression. It is defined as the ratio of the hydraulic work W_h, to the pump efficiency η. The hydraulic work is determined as the product of the mass flow rate at the pump inlet, the specific volume v, and the pressure difference between outlet and inlet. For a single effect AHP, this pressure difference corresponds to the pressure at the generator (high pressure) and the absorber (low pressure).

$$W_{el} = \frac{W_h}{\eta} = \frac{(p_g - p_a)\cdot v \cdot \dot{m}_{1.l}}{\eta} \tag{7}$$

The hydraulic work is added to the fluid as an enthalpy increase. Fluid properties, such as the specific volume, are evaluated at the pump inlet conditions. The same modeling approach is applied to the water pump, which is required in certain configurations, such as for simulating an absorption heat transformer.

2.4 Dynamic models

Up to this point, the dynamic models have been focus on the single effect absorption heat pump, which serves as the core component of the AHE. The dynamic model of the absorption heat pump is implemented in an object oriented manner, utilizing modular component models for pipes, valves, pumps, the solution heat exchanger, and the main

DOI 10.3384/ecp218949

heat exchangers (i.e. absorber, generator, evaporator and condenser) among others.

Key modeling assumptions are summarized below. These assumptions are commonly adopted in the literature (Herold, Radermacher, and Klein 2016).

- The refrigerant is assumed to be pure water.
- The condenser and generator operate at the same pressure, as do the evaporator and absorber, i.e. no pressure loss between them is considered.

Some of the component models such as valves, pumps and the solution heat exchanger have already been introduced (see section 2.3). Additional models, including pipes and various auxiliary components (e.g. sources), are taken from the MSL (Modelica Association 2024).

The primary work in the dynamic modelling involve the main heat exchangers. Several modeling approaches have been implemented, with a key distinction related to how heat exchange between the internal and external fluids and thermal inertia is handled.

The simplest approach utilizes an internal heat exchanger and it is analog to the static model formulation, where a fixed UA values is defined and the heat transfer rate is calculated using the logarithmic mean temperature difference. The process dynamics in this model are mainly limited to inertia related to the inner volumes of the vessels. The alternative approach models the heat exchanger externally, using a discrezited structure. For comparison, models ending in `_heatport` and `_internalHEX` within the `FluidBased.Dynamic.Components` package illustrate the effects of the difference of the modelling approach. More specifically, the models `AHPse_Dyn_internalHEX` and e.g. `AHPse_Dyn_heatport_nocp` within the package `...Components.Validation` provide a direct contrast between these two approaches.

Within the external heat exchanger modeling strategy, several versions have been developed. These include models that account for the thermal capacity of the heat exchanger itself, such as `...Dynamic.AHP.AHP`. Furthermore, some models are also capable of representing heat losses, whether radiative, convective, or both.

Some implementations incorporate a two-stage heat exchange process, with an additional heat exchanger between the external fluid and the entering solution in the absorber and generator; such models end in `_hex_extended`. This modification is motivated by the specific design of the AHP under study (WEGRACAL® SE 15), whose handbook indicates that the liquid solution coming from the absorber is evenly distributed within the generator and includes a sprayer in its schematics. A similar configuration is shown for the solution path from the absorber to the generator. In this context, incorporating an additional heat exchanger provides a straightforward way to reflect this design feature. However, it should be noted that the models do not include recirculation within the heat exchanger, which would be required to faithfully model the evaporator. This represents a limitation of the current models.

Further extensions include configurations that incorporate an additional solution pump transferring fluid from the generator back to the absorber, as implemented in `...Dynamic.AHP.AHP_pump`. This modification is again based on the WEGRACAL® SE 15 handbook, which explicitly states that the solution with a high lithium bromide concentration is pumped back to the absorber.

As with the static models, some code duplication exists in the current implementation, which complicates code review. This limitation will be addressed by introducing base classes and using the `extends` clause, by enabling feature toggling through if-conditioned component declarations. On the other hand, some of the dynamic models offer increased flexibility, compared to the static models, through the use of variable UA values in certain configurations, such as `...Dynamic.AHP.AHP_hex_extended`. In summary, the different versions result from a step-by-step implementation process that progressively increased model complexity and incorporated specific design features of WEGRACAL® SE 15, the device employed for model validation purposes. A clear conclusion about which model is preferable cannot yet be drawn, as no detailed analysis comparing the versions in terms of results and performance has been conducted.

2.4.1 Generator and Absorber

In general, the version with an externalized heat exchanger is preferred due to its greater flexibility. This modeling approach is applied to both the generator and the absorber, which are described in more detail below.

The same general modeling strategy is used to model the absorber and the generator. The main differences them lie in the direction of the vapor flow (compare the flow at position 7 and 10) and in the thermodynamic states of the entering fluid streams, see for example, the differences between the states at position 3 and 6 (refer to Figure 4).

The internal mass balance equations define the total mass within the component as the sum of the vapor and liquid phase masses. The total must equal the combined mass of refrigerant and lithium bromide solution, as formulated in Equations (8) to (10).

$$m = m_{ref} + m_{LiBr} = m_v + m_l \tag{8}$$

$$m_{ref} = m_v + m_l \cdot (1 - X_{LiBr}) \tag{9}$$

$$m_{LiBr} = m_l \cdot X_{LiBr} \tag{10}$$

The total mass is contained in a vessel of volume V, which is composed of contributions from both the vapor and liquid phases.

$$V = \frac{m_v}{\rho_v} + \frac{m_l}{\rho_l} \tag{11}$$

A vapor-liquid equilibrium is assumed between the liquid phase (H_2O/LiBr solution) and the vapor phase (pure

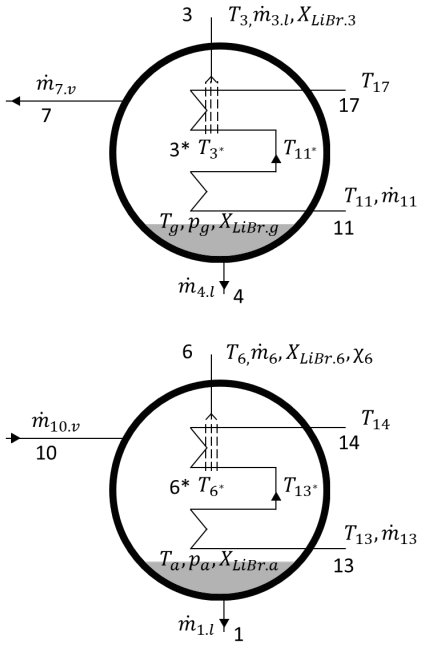


Figure 4. Schematics of the generator (top) and absorber (bottom).

refrigerant). Phase equilibrium is maintained by requiring the chemical potential of water to be equal in both phases, as expressed in Equation (12).

$$\mu_{l.ref} = \mu_v \tag{12}$$

The mass balance for the refrigerant in the generator and absorber is described by Equations (13) and (14) respectively. For example, the change in refrigerant mass within the generator is determined by the net mass flow: it is equal to the mass flow rate of the refrigerant entering from the absorber $\dot{m}_{3.ref}$, minus the mass flow rate of vaporized refrigerant exiting into the condenser $\dot{m}_{7.v}$, and minus the portion of liquid refrigerant returned from the generator to the absorber $\dot{m}_{4.l.ref}$.

$$\frac{d(m_{g.ref})}{dt} = \dot{m}_{3.l.ref} - \dot{m}_{7.v} - \dot{m}_{4.l.ref} \tag{13}$$

$$\frac{d(m_{a.ref})}{dt} = \dot{m}_{6.l.ref} + \dot{m}_{10.v} - \dot{m}_{1.l.ref} \tag{14}$$

Similarly, the mass balance for the solution in the generator and absorber is described by Equations (15) and (16) respectively.

$$\frac{d(m_{g.LiBr})}{dt} = \dot{m}_{3.l.LiBr} - \dot{m}_{4.l.LiBr} \tag{15}$$

$$\frac{d(m_{a.LiBr})}{dt} = \dot{m}_{6.l.LiBr} - \dot{m}_{1.l.LiBr} \tag{16}$$

With respect to the energy balance, the mass within the generator gains energy from three main sources: heat transfer from the external fluid, which provides the driving heat for refrigerant vaporization $\dot{Q}_g$, heat exchange with the surroundings, accounting for possible heat losses or gains $\dot{Q}_{g.loss}$ and enthalpy carried by the mass flows $\dot{H}_g$. The corresponding energy balance is defined by Equation (17).

$$\frac{d(U_j)}{dt} = \dot{H}_j + \dot{Q}_j - \dot{Q}_{loss.j} \quad \text{where } j \in \{g,a\} \tag{17}$$

The heat losses or gains to the environment are modelled using a constant thermal conductance and are assumed to be proportional to the temperature difference between the internal temperature of the vapor liquid equilibrium mixture, denoted as T_g for the generator and T_a for the absorber, and the ambient temperature T_{amb} (T_{amb} is now fixed to 25°C). This relationship is defined by Equation (18).

$$\dot{Q}_{loss.j} = UA_{j.amb} \cdot (T_j - T_{amb}) \quad \text{where } j \in \{g,a\} \tag{18}$$

The contribution of the enthalpy flow rate $\dot{H}$ to the energy balance in the generator and absorber is defined by Equations (19) and (20) respectively. It is expressed as the sum of enthalpy flows, i.e. the product of each entering (positive) and leaving (negative) mass streams and their respective associated specific enthalpies.

$$\dot{H}_g = \dot{m}_{3.l} \cdot h_{3^*.l} - \dot{m}_{7.v} \cdot h_{7.v} - \dot{m}_{4.l} \cdot h_{4.l} \tag{19}$$

$$\dot{H}_a = \dot{m}_6 \cdot h_{6^*} + \dot{m}_{10.v} \cdot h_{10.v} - \dot{m}_{1.l} \cdot h_{1.l} \tag{20}$$

Note that the enthalpies $h_{3^*.l}$ and $h_{6^*.l}$ correspond to the thermodynamic states at positions 3* and 6* respectively. These represent the "pre-heated" and "pre-cooled" streams, i.e. streams that have already exchanged heat with the external fluid. As a result, the heat transfer from the external fluid to the vessel contents is modeled in two stages.

The first stage involves heat transfer between the incoming external fluid (at position 11 and 13) and the internal vapor-liquid mixture within the generator or absorber, maintained at temperature T_g or T_a. The second stage involves heat exchange between the cooled external fluid (at positions 11* and 13*) and the fluid entering from the external circuit, initially at temperature T_3 or T_6, which is then heated and cooled to T_{3^*} and T_{6^*} respectively. This additional heat transfer step is specific to model versions identified by the suffix `hex_extended`.

The first stage corresponds to the main heat exchanger. In this model, the thermal capacity of the solid structure of the heat exchanger is explicitly considered. Both the external fluid and the solid parts are spatially discretized into n elements to capture transient behavior accurately (see Figure 5). The total heat flow entering the generator and absorber is defined by Equation (21).

$$\dot{Q}_j = \sum_{n=1}^{n} UA_j \cdot (T_{j.s.i} - T_j) \quad \text{where } j \in \{g,a\} \tag{21}$$

DOI
10.3384/ecp218949

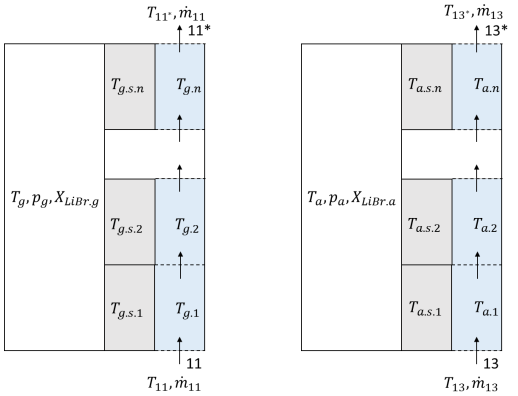


Figure 5. Schematics of internal heat exchanger model of generator (left) and absorber (right).

The energy balance of the solid part of the heat exchanger is defined by Equation (22) for each discretized section i. Here, C denotes the heat capacity of the solid in each section, expressed in Joules per Kelvin, and is calculated as $C = c \cdot m$, where c is the specific heat capacity and m is the mass of the solid in the corresponding section.

$$C\frac{d(T_{j.s.i})}{dt} = UA_j \cdot (T_j - T_{j.s.i}) + UA_j \cdot (T_{j.i} - T_{j.s.i}) \quad (22)$$

where $j \in \{g,a\}$ and $i \in \{1,2,\ldots,n\}$

The external fluid is modeled using the `DynamicPipe` model of the MSL. The static head is set to zero. The pressure loss coefficients are set to a negligible value. Changes in density are expected to be low. The energy balance equations for the external fluid are defined by Equations (23) and (24). $\dot{m}_{j.ext}$ and $h_{j.i}$ are used to indicate the entering (*in*) and leaving (*out*) mass flow rate and specific enthalpy at each volume i.

$$\frac{d(U_{j.i})}{dt} = \dot{H}_{j.i.ext} - UA_j \cdot (T_{j.i} - T_{j.s.i}) \quad (23)$$

$$\dot{H}_{j.ext} = \dot{m}_{j.ext.in} \cdot h_{j.i.in} - \dot{m}_{j.ext.out} \cdot h_{j.i.out} \quad (24)$$

where $j \in \{g,a\}$ and $i \in \{1,2,\ldots,n\}$

With regard to the second stage of heat transfer from the external fluid to the entering fluid at the main vessel, represented by $\dot{Q}_{hex}$, (corresponding to positions 11*, 17, 3 and 3* for the generator, and positions 13*, 14, 6 and 6* for the absorber), it is important to note that the thermal capacity of the solid structure is neglected in this part of the model. The model used is based on `PartialEffectivenessNTU`[1]. In this model, the overall heat transfer coefficient-area product (UA value) is assumed to be constant. The heat transfer is defined as the product of the maximum possible heat flow rate $\dot{Q}_{hex.max}$ and the heat exchanger effectiveness ε, as shown in Equation (25). The effectiveness ε is a function of the number of transfer units ($NTU = UA/\dot{C}_{min}$) and the thermal capacities of the internal and external streams, $\dot{C}_{int}$ and $\dot{C}_{ext}$, respectively, where $\dot{C} = \dot{m} \cdot c_p$. More specifically, ε depends on the ratio of the minimum to the maximum capacity rate, $\dot{C}_r = min(\dot{C}_{int}, \dot{C}_{ext})/max(\dot{C}_{int}, \dot{C}_{ext})$, as described in Equation (26). The maximum possible heat transfer rate $\dot{Q}_{hex.max}$ is defined as the product of the minimum thermal capacity rate and the temperature difference between the two entering fluid streams (e.g. temperatures at position 11* and 3 in the case of the generator), as shown in Equation (27).

$$\dot{Q}_{hex} = \dot{Q}_{hex.max} \cdot \varepsilon \quad (25)$$

$$\varepsilon = \frac{1 - e^{-NTU \cdot (1 - \dot{C}_r)}}{1 - \dot{C}_r \cdot e^{-NTU \cdot (1 - \dot{C}_r)}} \quad (26)$$

$$\dot{Q}_{hex.max} = \dot{C}_{min} \cdot (T_j - T_k) \quad (27)$$

where $(j,k) \in \{(11^*,3),(13^*,6)\}$

2.4.2 Evaporator and condenser

The primary difference in the modeling of the evaporator and condenser, compared to the generator and absorber, lies in the treatment of heat transfer and the working fluid. In the evaporator and condenser, only pure refrigerant is present, and the heat exchange between the vapor–liquid equilibrium mixture and the external fluid is modeled in a simplified manner. Specifically, only a single main heat exchanger is used in these components—no extended or two-stage heat exchanger configuration is applied.

The equations that describe the mass balance of the evaporator and condenser are equivalent to Equations (8) to (12) assuming $X_{LiBr} = 0$. Equations (13) and (14) need to be substituted by Equations (28) and (29).

$$\frac{d(m_{c.ref})}{dt} = \dot{m}_{7.v.ref} - \dot{m}_{8.l.ref} \quad (28)$$

$$\frac{d(m_{e.ref})}{dt} = \dot{m}_{9.l.ref} + \dot{m}_{9.v.ref} - \dot{m}_{10.v.ref} \quad (29)$$

Regarding the energy balance, Equations (18), (22), and (23) apply under the assumption that $j \in \{c,e\}$. Furthermore, Equations (19) and (20) must be replaced by Equations (30) and (31).

$$\dot{H}_c = \dot{m}_{7.v} \cdot h_{7.v} - \dot{m}_{8.l} \cdot h_{8.l} \quad (30)$$

$$\dot{H}_e = \dot{m}_9 \cdot h_9 - \dot{m}_{10.v} \cdot h_{10.v} \quad (31)$$

As mentioned above, the two-stage heat exchanger configuration is not applied in this case. Consequently, Equations (25) to (27) are not relevant to the evaporator and condenser models.

[1] Model Fluid.HeatExchangers.BaseClasses.PartialEffectivenessNTU of the Buildings library (Wetter et al. 2014)

3 Absorption heat exchanger (AHE) model validation

3.1 Model description

An AHE model has been developed based on the `PlateHeatExchangerEffectivenessNTU` model for the external heat exchanger and the `Dynamic.AHP.AHP_pump_hex_extended` model for the AHP. This model has been validated using experimental measurements obtained within the framework of the Absolut project (Eberhöfer 2022; Zotter, Eberhöfer, and Seidnitzer-Gallien 2023). For detailed information about the experimental test rig, the reader is referred to (Eberhöfer 2022).

The measurement data are used as input for the model, particularly the inlet mass flow rates and temperatures at the absorber, external heat exchanger, and generator. These measurements correspond to the sensor locations at positions 13, 20, and 11, respectively (see Figure 1).

In regard of the parametrization and setup,

- Based on estimation heat loss coefficients from (Eberhöfer 2022), the parameter `U_amb` is fixed to 3 W/K for each main heat exchanger. Heat losses due to radiation are not considered.
- Details on the internal control of the absorption heat pump (e.g. solution pump) are not known. The mass flow rate from the generator to the absorber has been fixed to 0.12 kg/s. The mass flow rate from the absorber to the generator is adjusted to keep the volume at the generator at a constant level.
- The volume of each vessel is estimated to be 70 liters.
- The sum of heat capacity per section C used in Equation 22 is set to 5 kJ/K for all main heat exchangers. The values in Table 1 of (Kohlenbach and Ziegler 2008) are used as reference values. The values of the heat exchanger tube bundles (assuming 100% weight) vary from 3.9 kJ/K to 6.1 kJ/K.
- The external heat exchanger SWEP B25Tx41/2P is parametrized directly based on expected performance obtained with the manufacturer software.

In regard of the model calibration, the main work is related to the estimation of appropriate values for the 6 unknown UA values of the AHP heat exchangers. The calibration is performed manually through stepwise adaptation of the parameters, using approximately 90 stationary points from various experiments as benchmarks. The UA value for each element n required for Equation (22), which are the same values used in Equation (23) are obtained with Equation (32) and are defined based on nominal operating conditions by two parameters ($\dot{m}_{nom}$ and UA_{nom}). The mass flow rates $\dot{m}$ and $\dot{m}_{nom}$ are referred to the external fluid, i.e. positions 11, 13, 16 and 18.

$$UA_j = \frac{UA_{nom}}{n} \left(\frac{\dot{m}}{\dot{m}_{nom}} \right)^{0.6} \quad \text{where } j \in \{g, a\} \tag{32}$$

The values after calibration are summarized in Table 1. Additionally, the fixed UA in W/K values used for the solution heat exchanger, and the second stage heat exchange at the generator and absorber, see Equations (25) to (27), are set respectively to 350, 100 and 400 W/K.

Table 1. Parametrization of main heat exchangers after calibration.

Heat exchanger	UA_{nom} **in kW/K**	$\dot{m}_{nom}$ **in kg/s**
Evaporator	1.2	0.045
Generator	1.5	0.045
Absorber	3.9	0.35
Condenser	3.9	0.35

To analyze the dynamic behavior of the model, simulation results covering 14 hours of experimental operation are compared with measurement data. The measurement uncertainty of the temperature sensors is estimated at $\pm$ 0.2 K as reported in (Zotter, Eberhöfer, and Seidnitzer-Gallien 2023). The comparison primarily focuses on the outlet temperatures of the main heat exchangers. To assess the model's accuracy under steady-state conditions, data points identified as quasi-stationary- according to the criteria outlined in (Eberhöfer 2022) are used. These points correspond to periods in which the system operation is assumed to have stabilized.

Results from a single experimental run are presented here. The boundary conditions used in the simulation are summarized in Figure 6. Points at which the operation is considered to have reached steady-state are marked with circles. For this analysis, the evaluation period spans from hour 1 to hour 15 of regular operation.

3.2 Validation results

Simulation results at positions 12, 14, 15, 17, 19 and 21 (see Figure 1) over the evaluation period of the experiment are presented in Figure 7 alongside the corresponding measurement data and the calculated deviations. The model predicts the outlet temperature behavior with generally good accuracy. A total of 841 values (one every minute) are used for calculate the deviation at the given instant and used for the evaluation, see Table 2. Most monitored temperatures, specifically T_{14}, T_{15}, T_{17}, and T_{21}, show low average deviations in the range of -0.4 K to 1.3 K, with deviations spanning from -2.7 K up to 3.2 K. In contrast, the outlet temperature of the external heat exchanger T_{19} and the outlet temperature of the evaporator T_{12} exhibit a higher average deviation of 2.7 K and 1.6 K with a maximum deviation (underestimated temperature) of 8.3 K and 3.1 K respectively.

When evaluating temperature deviations under steady-state operating conditions, the average deviations are comparable to those observed over the entire duration of the

DOI 10.3384/ecp218949

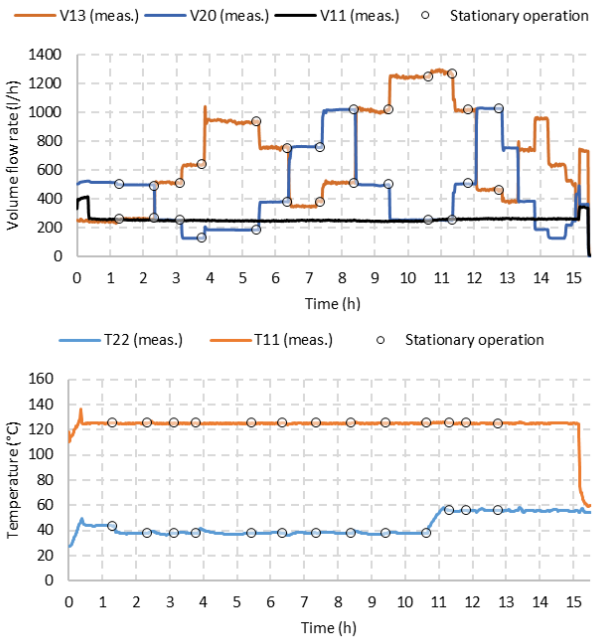


Figure 6. Measured data (meas.) used as boundary conditions for the model validation. Volume flow rates (top) and temperatures (bottom). Volume flow rate and temperature entering the generator V_{11} and T_{11}. Volume flow rate entering the absorber V_{13} and the external heat exchanger (secondary side) V_{20} and their temperature T_{22}.

experiment (cf. Table 2 and Table 3). It is noteworthy that the large temperature deviations previously observed at the external heat exchanger outlet (T_{19}) during transient conditions are reduced once the system reaches steady-state. Under these conditions, the deviations at T_{19} range from a minimum of 0.74 K to a maximum of 5.5 K, with an average deviation of 2.6 K across the thirteen steady-state data points - the highest average deviation among the six evaluated temperatures.

4 Conclusions

The presented Absolut Modelica library includes both static models, which have been verified against literature

Table 2. Average, maximum and minimum temperature deviations (in Kelvin) defined as the reference value (measurement) minus the calculated values (simulation result) for 14 hours evaluation period.

	Temperature deviation in K		
Temperature	**Average**	**Maximal**	**Minimal**
T_{14}	−0.26	1.42	−2.73
T_{15}	−0.39	1.1	−2.1
T_{12}	1.56	3.07	−0.59
T_{17}	1.28	3.16	−1.87
T_{19}	2.70	8.26	−0.33
T_{21}	−0.07	0.65	−1.38

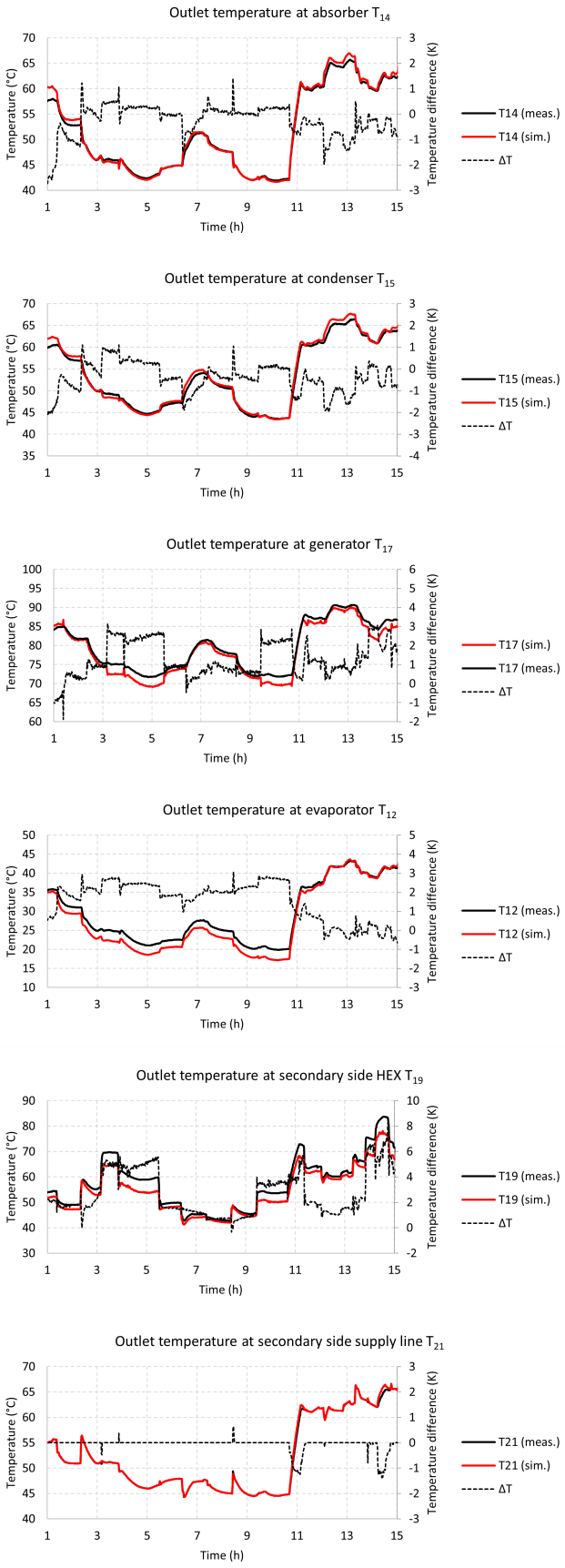


Figure 7. Temperature difference (ΔT) in K between measurements (meas.) and simulation results (sim.) at outlet of main heat exchangers.

Table 3. Average, maximum and minimum temperature deviations (in Kelvin) defined as the reference value (measurement) minus the calculated values (simulation result) for monitored positions under steady-sate operating conditions.

Temperature	Temperature deviation in K		
	Average	**Maximal**	**Minimal**
T_{14}	−0.29	0.53	−2.11
T_{15}	−0.43	0.7	−1.66
T_{12}	1.2	2.61	−0.79
T_{17}	1.72	2.74	−0.01
T_{19}	2.58	5.49	0.74
T_{21}	0.0	0.0	−0.05

values and are suitable for preliminary analysis of absorption thermodynamic cycles, and dynamic models. Despite the current limitations - such as the use of simplified heat exchanger models and uncertainties related to internal control - the simulation results of the dynamic models show good agreement with experimental data. Both the dynamic and static models utilize the implemented correlations for the H_2O/LiBr working pair. Overall, the library provides a solid foundation for the analysis of absorption thermodynamic cycles, offering basic yet reliable models for further study and development. As an open-source resource (available at `https://github.com/carlesRT/Absolut`), it is intended to be extended and refined by the community to address current limitations and support broader applications.

Acknowledgements

The project AbSolut (FFG-Nr.: 879433) is supported with the funds from the Climate and Energy Fund and implemented in the framework of the RTI-initiative “Flagship region Energy”. The TREASURE project has received funding from the European Union under Grant Agreement No. 101136095. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or CINEA. Neither the European Union nor CINEA can be held responsible for them.



Nomenclature

Table 4. Description of subscripts and symbols.

Subscript	Description
a	Absorber
amb	Ambient / Surroundings
c	Condenser
e	Evaporator
el	Electric
ext	External fluid
g	Generator
h	Hydraulic
hex	Heat exchanger
l	Liquid state
LiBr	Lithium bromide
Loss	Heat losses
max	Maximal
min	Minimal
p	constant pressure
r	Ratio between minimal and maximal
ref	Pure refrigerant
s	Solid state
sol	Solution, i.e. LiBr-water mixture/solution
v	Vapor state
$1, 2, 3, \ldots$	Position of the variable in the AHE

Symbols	Description
C	Heat capacity (J/K)
$\dot{C}$	Capacity rate (W/K)
c	Specific heat capacity (J/(kg.K))
$\dot{H}$	Enthalpy flow rate (W)
m	Mass (kg)
$\dot{m}$	Mass flow rate (kg/s)
NTU	Number of transfer units (-)
p	Pressure (Pa)
$\dot{Q}$	Heat flow rate (W)
T	Temperature (K)
U	Internal energy (J)
UA	Heat transfer coefficient times area (W/K)
V	Volume (m^3)
W	Work (W)
X	Mass fraction (kg/kg)
y	Valve opening (-)
ε	Heat exchanger effectiveness/efficiency (-)
η	Pump efficiency (-)
μ	Specific chemical potential (J/kg)
ν	Specific volume (kg/m^3)
ρ	Density (m^3/kg)
χ	Vapor quality (-)

References

Chen, J.J.J. (1987). “Comments on improvements on a replacement for the logarithmic mean”. In: *Chemical Engineering Science* 42.10. Publisher: Elsevier BV, pp. 2488–2489. ISSN: 0009-2509. DOI: 10.1016/0009-2509(87)80128-8. URL: https://linkinghub.elsevier.com/retrieve/pii/0009250987801288 (visited on 2025-05-01).

Corrales Ciganda, Jose Luis et al. (2016-06). “Thermodynamic cycle models for the development of absorption heat trans-

formers for the process industry". In: *Heat Powered Cycles Conference*. University of Nottingham.

Cudok, Falk (2021-03-15). "Gemeinsamkeiten und Unterschiede zwischen Wärmetransformator- und Wärmepumpenprozess". Doktor der Ingenieurwissenschaften. Berlin: TU Berlin. 147 pp. URL: https://doi.org/10.14279/depositonce-11730.

De La Calle, Alberto et al. (2016-12). "Dynamic modeling and simulation of a double-effect absorption heat pump". In: *International Journal of Refrigeration* 72, pp. 171–191. ISSN: 01407007. DOI: 10.1016/j.ijrefrig.2016.07.018. URL: https://linkinghub.elsevier.com/retrieve/pii/S0140700716302298 (visited on 2023-12-20).

Dymola (2024-04-19). Version 2024x Refresh 1. URL: https://www.3ds.com/products/catia/dymola.

Eberhöfer, Damian (2022-10). *Adaption und Untersuchung eines Absorptionswärmetauschers zur Rücklauftemperaturreduktion in Fernwärmenetzen*.

EDF LAB CHATOU PRISME Department (2024). *Thermosyspro*. Thermosyspro. URL: https://thermosyspro.com/index.html.

El Hefni, Baligh (2019). *Modeling and Simulation of Thermal Power Plants with ThermoSysPro: A Theoretical Introduction and a Practical Guide*. In collab. with Daniel Bouskela. Cham: Springer. 1 p. ISBN: 978-3-030-05105-1.

Febres, Jesus et al. (2017). *District Cooling Open Source Library (DCOL)*. Version 4.1. URL: https://zenodo.org/records/1215665.

Fu, Lin et al. (2010-03). "A district heating system based on absorption heat exchange with CHP systems". In: *Frontiers of Energy and Power Engineering in China* 4.1, pp. 77–83. ISSN: 1673-7393, 1673-7504. DOI: 10.1007/s11708-010-0022-0. URL: http://link.springer.com/10.1007/s11708-010-0022-0 (visited on 2024-02-09).

G. Beckmann, F. Schittl, G. Piringer, D. Rixrath, R. Krotil, J. Krail (2019-12-19). *TeTra - Thermische Energietransformation zur Wärme- und Kälteauskopplung sowie Effizienzsteigerung in Nah- und Fernwärmenetzen*. Final report, p. 125.

Herold, Keith E., Reinhard Radermacher, and Sanford A. Klein (2016). *Absorption Chillers and Heat Pumps*. Second edition. London New York; Boca Raton: CRC Press. 1 p. ISBN: 978-1-4987-1435-8. DOI: 10.1201/b19625.

Hu, Tianle, Yi Jiang, and Xiaoyun Xie (2009). "A Novel District Heating Solution Based on Absorption Heat Exchanger(AHE) for Different Types of Cogeneration Plants". In: *Proceedings of international sustainable energy conference ISEC 2018, Graz*.

Kohlenbach, P. and F. Ziegler (2008-03). "A dynamic simulation model for transient absorption chiller performance. Part I: The model". In: *International Journal of Refrigeration* 31.2, pp. 217–225. ISSN: 01407007. DOI: 10.1016/j.ijrefrig.2007.06.009. URL: https://linkinghub.elsevier.com/retrieve/pii/S0140700707001181 (visited on 2023-12-20).

Li, Yan et al. (2011-02). "A new type of district heating method with co-generation based on absorption heat exchange (co-ah cycle)". In: *Energy Conversion and Management* 52.2, pp. 1200–1207. ISSN: 01968904. DOI: 10.1016/j.enconman.2010.09.015. URL: https://linkinghub.elsevier.com/retrieve/pii/S0196890410004176 (visited on 2024-02-09).

Modelica Association (2024). *Modelica Standard Library*. URL: https://github.com/modelica/ModelicaStandardLibrary.

Modelica Association (n.d.). *Modelica*. URL: https://modelica.org/.

OpenModelica (2024-12-14). Version v.1.24.3. URL: https://openmodelica.org/.

Pettersson, F. (2008-08). "Heat exchanger network design using geometric mean temperature difference". In: *Computers & Chemical Engineering* 32.8. Publisher: Elsevier BV, pp. 1726–1734. ISSN: 0098-1354. DOI: 10.1016/j.compchemeng.2007.08.015. URL: https://linkinghub.elsevier.com/retrieve/pii/S0098135407002256 (visited on 2025-05-01).

TIL Suite (n.d.). Aachen. URL: https://tlk-energy.de/software/til-suite.

Wernhart, Michael et al. (2022). "Vereinfachung von Absorptionskälteanlagen-Modellen". In.

Wetter, Michael et al. (2014-07-04). "Modelica Buildings library". In: *Journal of Building Performance Simulation* 7.4, pp. 253–270. ISSN: 1940-1493, 1940-1507. DOI: 10.1080/19401493.2013.765506. URL: http://www.tandfonline.com/doi/abs/10.1080/19401493.2013.765506 (visited on 2024-12-06).

Yang, Fang, Nico Mirl, and Fabian Schmid (2020). *Potenziale von Absorptionswärmepumpe in zentralisierten Wärmeversorgungsnetzen*, p. 79.

Yuan, Zhe and Keith E. Herold (2005-07). "Thermodynamic Properties of Aqueous Lithium Bromide Using a Multiproperty Free Energy Correlation". In: *HVAC&R Research* 11.3, pp. 377–393. ISSN: 1078-9669, 1938-5587. DOI: 10.1080/10789669.2005.10391144. URL: https://www.tandfonline.com/doi/full/10.1080/10789669.2005.10391144 (visited on 2023-12-20).

Zhang, Yin, Wenxing Shi, and Yinping Zhang (2014-02). "From heat exchanger to heat adaptor: Concept, analysis and application". In: *Applied Energy* 115, pp. 272–279. ISSN: 03062619. DOI: 10.1016/j.apenergy.2013.11.015. (Visited on 2024-12-14).

Zotter, Gerald, Damian Eberhöfer, and Carina Seidnitzer-Gallien (2023). "An energetical, exergetical and experimental analysis of an absorption heat exchanger used as transfer substation in an already existing district heating grid". In: 14th IEA Heat Pump Conference. Chicago.

Appendix A: Review of open source models

The ThermoSysPro library (EDF LAB CHATOU PRISME Department 2024; El Hefni 2019) includes a model for an absorption chiller (`AbsorptionRefrigeratorSystem`). This model employs non-standardized interfaces rather than the standard Modelica fluid interface `FluidPort`, limiting compatibility with other Modelica libraries. The model provides interfaces for the external fluids of the evaporator and generator, while the condenser and absorber are connected in series using idealized internal sources and sinks. This modeling approach restricts the flexibility of the model, requiring modifications for more generic applications. The H_2O/LiBr fluid functions in `Properties.WaterSolution` are sufficient for using the provided model but are constrained to a few equations with ambiguous references to the underlying data.

The District Cooling Open-Source Library (DCOL) is an open-source library that provides parametric thermo-fluid dynamic models for components of District Cooling Systems (Febres et al. 2017). It includes two absorption chiller models: a standard model and a simplified version that eliminates the need for control signals for internal valves. Both models share core components, namely the absorber, evaporator, generator, condenser, and pump, and utilize standard interfaces based on the Modelica Standard Library (Modelica Association 2024). The models are dynamic, accounting for vessel volumes and the mass of heat exchangers. Several examples simulations are provided, the simplified model runs successfully, while the standard model fails to initialize. The library also implements fluid property functions for the H_2O/LiBr working pair. However, inconsistencies have been observed in the definition of some thermodynamic functions, particularly a mismatch that can lead to numerical issues. For example, given a pressure `p`, temperature `T` and concentration `X` the result of `temperature(setState_phX(p,h,X))`, where the enthalpy `h` is obtained via `specificEnthalpy(setState_pTX(p,T,X))` will differ slightly from the original input temperature `T`.

The Buildings library (Wetter et al. 2014) includes a model of an indirect steam-heated absorption chiller, that uses performance curves and is adapted from the EnergyPlus Absorption:Indirect model[1]. It employs standard interfaces from the Modelica Standard Library (Modelica Association 2024). The model uses six functions to predict key performance metrics, including cooling capacity, pump power consumption, generator heat flow rate, and condenser heat flow. These functions rely on performance data stored in a `record`. Several example simulations are provided to demonstrate the model's functionality.

Appendix B: Additional validation results

In this section validation results of selected static models are presented. The results are used to confirm the correct implementation of the models and, indirectly, that of the fluid property correlations as well.

Table 5. Validation results for a single-effect absorption heat pump static model. Deviation in kW and % between reference values from Table 6.1 in (Herold, Radermacher, and Klein 2016) and the simulation results from `AHP_se_UAfix_eps`.

	Deviation	
Heat flow rate at...	**in kW**	**in %**
Absorber	0.09	0.62
Generator	0.09	0.62
Evaporator	0.07	0.61
Condenser	0.07	0.60
Solution heat exchanger	−0.03	−0.84

[1]Chiller:Absorption:Indirect, https://designbuilder.co.uk

Table 6. Validation results for a single-effect Type II absorption heat pump static model. Deviation in kW and % between reference values from Table 6.6 in (Herold, Radermacher, and Klein 2016) and the simulation results from `AHPTypeII_UAfix_HXeps`.

	Deviation	
Heat flow rate at...	**in kW**	**in %**
Absorber	−0.41	−0.22
Generator	−0.94	−0.51
Evaporator	0.10	0.05
Condenser	1.31	0.70
Solution heat exchanger	−0.33	−0.47

Table 7. Validation results of a static model for a parallel flow double-effect absorption heat pump. Deviation in kW and % between reference values from Table 7.4 in (Herold, Radermacher, and Klein 2016) and the simulation results from `AHP_pf_de_UAfix`.

	Deviation	
Heat flow rate at...	**in kW**	**in %**
Evaporator	−1.02	−0.28
Absorber	0.91	0.21
Condenser 1 (low pressure)	0.61	0.33
Condenser 2 / Generator 1	0.46	0.23
Generator 2 (high pressure)	−0.43	−0.16
Solution heat exchanger 1	−0.08	−0.25
Solution heat exchanger 2	0.04	0.06

Table 8. Validation results of a static model for a series flow double-effect absorption heat pump. Deviation in kW and % between reference values from Table 7.5 in (Herold, Radermacher, and Klein 2016) and the simulation results from `AHP_sf_de_UAfix`.

	Deviation	
Heat flow rate at...	**in kW**	**in %**
Evaporator	−17.02	−4.57
Absorber	21.21	4.62
Condenser 1 (low pressure)	6.08	2.84
Condenser 2 / Generator 1	10.91	5.88
Generator 2 (high pressure)	−10.22	−3.41
Solution heat exchanger 1	1.14	3.20
Solution heat exchanger 2	−1.24	−1.36

Table 9. Validation results of a static model for a resorption cycle. Deviation in kW and % between reference values from Table 8.6 in (Herold, Radermacher, and Klein 2016) and the simulation results from `Resorption`.

	Deviation	
Heat flow rate at...	**in W**	**in %**
Generator 1 (high pressure)	1.3	0.01
Generator 2 (low pressure)	−1.2	−0.02
Absorber 2 (high pressure)	0.1	0.00
Absorber 1 (low pressure)	0.6	0.00
Solution heat exchanger 1	−3.6	−0.10
Solution heat exchanger 2	−1.6	−0.05

DOI 10.3384/ecp218949

Identification and Elimination of Instabilities During Simulation of Highly Stiff Vehicle Electrical Power System Models

Robert Weber[1] Staša Gejo[1] Rainer Gehring[1] Lars Mikelsons[2]

[1]BMW Group, Germany, {robert.we.weber, stasa.gejo, rainer.gehring}@bmw.de
[2]Chair for Mechatronics, University of Augsburg, Germany, lars.mikelsons@informatik.uni-augsburg.de

Abstract

In vehicle electrical power system simulations, reliable models are required, especially for critical use cases such as inrush currents, short circuits, and load dump scenarios. These switching scenarios involve rapid changes in resistance and result in stiff models due to the fast dynamics. Instabilities are observed immediately after switching, highlighting significant numerical challenges. In this study, we investigate the origins of these inaccuracies and propose robust solutions through improved electrical models and advanced numerical integration techniques with optimized error tolerance settings. The suggested decoupling approach employs physical and signal filtering techniques to replace the variable resistor used to simulate the fuse switching process. The nominal value strategy involves establishing an appropriate baseline value. The methods discussed yield stable and accurate simulations when implemented with appropriately adapted model-specific parameterizations.

Keywords: stiff model simulation, vehicle electrical system simulation, numerical instabilities, decoupling method, nominal value

1 Introduction

The importance of the electrical power system for vehicle operation and safety is ever-increasing, due to the growing complexity and reliance on integrated assistance and safety functions in modern vehicles. As the automotive industry moves towards increased automated and autonomous driving functions, new challenges arise in achieving functional safety (Schwimmbeck, Buchner, and Herzog 2018).

New electronic fuses play a vital role in ensuring the safe operation of power distribution units by implementing protection mechanisms like overcurrent, overtemperature, undervoltage, and wire protection algorithms. These functions guarantee the reliable supply of safety-critical consumers even in error conditions, through the quick and selective isolation of faults (Baumann, Abouzari, et al. 2023). To ensure the reliability and availability of these safety-critical systems, it is crucial to perform highly dynamic simulation tests, such as inrush current, short circuit, and load dump scenarios, visualized in Figure 1. Short circuit tests are performed adjacent to the load to evaluate the fuse's fault isolation capability. Moreover, inrush current tests assess the immediate electrical response upon fuse closure, while load dump scenarios simulate the transient overvoltage conditions, resulting from the abrupt load disconnection via the fuse.

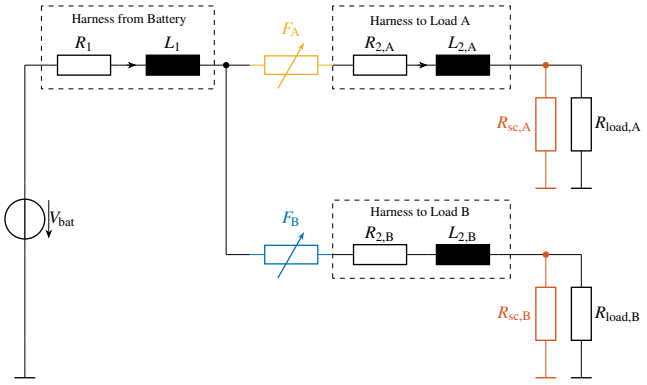


Figure 1. Simplified model of the vehicle electrical power system visualizing the different test cases: short circuit scenarios via small resistors $R_{sc,A}$ and $R_{sc,B}$ (marked in orange) connected to ground, load dump scenario by opening fuse F_A (marked in yellow) and closing fuse F_B for the inrush current scenario (marked in blue).

Evaluating the vehicle electrical power system functionality and simulation results is challenging, as discontinuities and steep gradients can lead to numerical artifacts. The time constants of the circuit elements can vary by several orders of magnitude, so the effective time constant changes as the resistance varies during switching. This change in resistance alters the dynamic behavior of the system, which is why it is considered *stiff*. Highly stiff or hybrid stiff systems – i.e., systems with discontinuities and Boolean functions that alter the mathematical model description – can be computationally expensive to solve. Moreover, their results are sensitive to solver settings, which poses a challenge for safety-related investigations (Liu, Felgner, and Frey 2010).

One of the common problems encountered during the simulation of electronic fuses in vehicle electrical systems is the appearance of deviations from a physically correct behavior. These deviations manifest as uneven sawtooth disturbances in the voltage across the fuse or as a single exaggerated spike caused by integration instability, rather

than by physical phenomena. Such voltage behavior can be critical, as it serves as an input for additional components – such as the diodes connected in parallel with the fuse. Diodes alone may pose numerical challenges, and an unstable voltage input can introduce further complications in the numerical simulation. They are employed as a protective measure to limit the voltage across the fuse. Furthermore, the fuse trigger limits can be incorrectly breached, leading to erroneous simulation model behavior (e.g. undervoltage). All these aspects can have a significant negative impact on the numerical stability and functionality of simulation models. The goal of this work was to find the sources of numerical instabilities in these models and to resolve them by means of physical remodeling or by using more advanced integrator settings.

Chapter 2 discusses the properties of the vehicle electrical power system. Chapter 3 characterizes the non-physical numerical artifacts observed in the simulation, examining their causes and consequences. The practical remodeling approaches are presented and discussed in Chapter 4. Integrator settings as well as the error estimation and tolerance, and their effect on the simulation results are covered in Chapter 5. Lastly, Chapter 6 compares the presented methods applied in two different models and sets the ground for future research and improvements.

2 Vehicular Electrical Power System Simulation Models

The models examined in this chapter represent key elements of vehicular power systems, consisting of voltage and current sources, variable resistors, and signal conditioning circuits, used to represent the behavior of switched electronic fuses and other protective devices. These models can be used to capture transient effects previously mentioned under inrush currents, short-circuit events, and load dump conditions, covering a broad range of time constants.

Such highly dynamic systems are usually also *stiff*. There is no unique definition of stiffness; however, according to Lambert (1992) and Cellier and Kofman (2006), there are several criteria which can be used to recognize or define a stiff system. The two criteria important for this work are:

- Statement A: "A linear constant coefficient system is stiff if all of its eigenvalues have negative real part and the stiffness ratio is large."
- Statement B: "A system is said to be stiff in a given interval of time if in that interval the neighboring solution curves approach the solution curve at a rate which is very large in comparison with the rate at which the solution varies in that interval."

According to Statement A, a linear electrical system is considered stiff if the real parts of its eigenvalues differ by several orders of magnitude.

Let us look at an example of an interconnected load model, consisting of a line, load input impedance, and closed switch found in Baumann, Weissinger, and Herzog (2019), shown in Figure 2. Its eigenvalues are $\lambda = \{-1.43 \cdot 10^{10}, -5.88 \cdot 10^{4} \pm j\, 4.01 \cdot 10^{4}, -4.45 \cdot 10^{-3}\}$, whose ratios of real parts go from 10^{6} up to 10^{13}. This type of circuit is omnipresent in the vehicle electrical power system, though in slightly varying input impedance topologies but with similar order of magnitude of its parameters. These circuits connect numerous consumers (small motorized components, heaters, wipers, and ventilators) with the supply, converters, and power distributors. System stiffness arises from the input impedance, cable, and parasitic effects in complex components which can create states with vastly different time constants (referring to Statement B). Additionally, fuse switching causes the total resistance value to change, amplifying variations in time constants and eigenvalues over time. Consequently, this alteration introduces a change in the stiffness of the system. Another cause of numerical issues and instabilities can be discontinuities, as well as nonlinear components and signals.

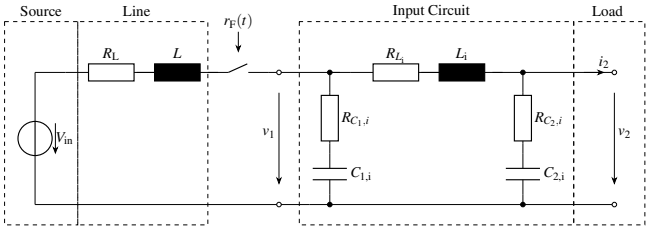


Figure 2. Interconnected load model adapted from Baumann, Weissinger, and Herzog (2019).

The problem of numerical integration for stiff problems has persisted for decades, and no integration method is able to solve every stiff problem. Various integrators can be used for stiff applications, such as Radau, LSODAR, CVODE, and DASSL, as well as a whole range of ESDIRK methods (Liu, Felgner, and Frey 2010; Cohen, Hindmarsh, and Dubois 1996; Blom et al. 2016). These integrators can be classified as either single-step methods, which rely only on the current integration step, or multi-step methods, which use several previous steps to compute the next time step. In addition, the methods differ in the order of integration. Four solvers, including DASSL and RADAU IIA, were compared in a stiff system simulation in Liu, Felgner, and Frey (2010), which showed that DASSL had a higher number of calculated points and a higher solver efficiency for a stiff continuous and stiff hybrid model. However, the study showed a lack of accuracy in all compared solvers for the stiff hybrid system. It is important to note that accuracy and efficiency always depend on the simulated model and cannot be generalized. This work will use DASSL for all the models, as it is well-suited for stiff systems due to its implicit multi-step properties.

DOI 10.3384/ecp218961

3 Model Analysis

"Stability" is a broad concept used to define the behavior of a system. *Analytical* stability, in simplified terms, refers to the property of the system to achieve a steady state after a transient or to remain asymptotically bound. On the other hand, the lack of *numerical* stability (Cellier and Kofman 2006), or rather a *discretization* instability (Ardourel and Jebeile 2021), refers to the lack of ability of the system to numerically reproduce the exact solution. The reason for it lies in the small differences between the original differential equation and its implemented discretized version in the applied numerical method. In other words, a system is numerically unstable if, however small the step size is, a significant deviation arises between the exact solution and the numerical solution (Ardourel and Jebeile 2021).

An additional problem of numerical integration is inaccuracy due to round-off, truncation, and accumulation error. Reducing the step size improves the accuracy but also increases the computational cost. A good balance between error and cost can be achieved by the use of variable-step integration algorithms which utilize specific step-size control algorithms. (Cellier and Kofman 2006)

If a variable-step solver is used, the estimated local error at a given simulation step is controlled to satisfy the following condition (*Dymola Full User Manual* 2022):

$$|\text{local error}| < \text{tol}_{\text{rel}} \cdot |x_i| + \text{tol}_{\text{abs}} \tag{1}$$

By default in Dymola, the relative and absolute tolerances of the simulation are equal, reshaping the Equation 1 into Equation 2, where "tol" denotes the available tolerance input in the setup.

$$|\text{local error}| < \text{tol} \cdot (|x_i| + 1) \tag{2}$$

Dymola Full User Manual (2022) roughly defines the relative tolerance as the approximate definition of the "number of expected true digits in the solution". Meaning, if the value of the state x at a time step i is close to zero, the value of absolute tolerance will determine the limits of the local error. The role of the $+1$ operation in the brackets in Equation 2 and the settings that change its value, affecting the error calculation, will be discussed in Chapter 5.

Consider the example from Figure 3, which is used to examine the instability in the fuse voltage when inductance is present. The resistor R_{load} represents a simple resistive load with no input circuit. The fuse is connected to the source and load via wiring harness models. The fuse is modeled as a simplified component that switches off by acting as a variable resistor, with its resistance changing according to the adapted sigmoid function defined in Equation 3. A key element of these simulations is the detailed representation of the resistance change dynamics of the fuse, closely approximating the behavior observed in the physical fuse. Such modeling provides a more accurate insight into the fuse's effect on the vehicle electrical system.

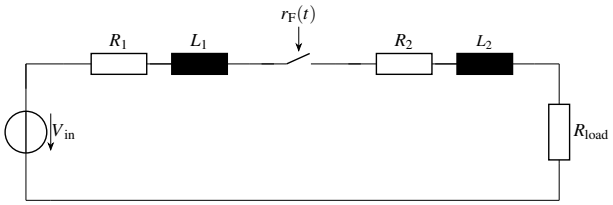


Figure 3. A simple circuit used to investigate the fuse voltage instability in the presence of inductance. The circuit comprises a source, a fuse with upstream and downstream wiring harness models, and a simplified load.

$$r_F(t) = -\frac{R_{F,off}}{1 + e^{k\cdot(t - T_{sw})}} + R_{F,off} + R_{F,on} \tag{3}$$

The switching time T_{sw} is the defined time instant where the switch between the "on" and "off" states is at a halfway point; k is the rate of change of the exponential element. The "on" and "off" state resistances are $R_{F,on} = 1\,\text{m}\Omega$ and $R_{F,off} = 10\,\text{M}\Omega$. The off state is chosen to approximate "open-circuit" conditions. The simulation is run with the DASSL solver, using different tolerance values ranging from 10^{-5} to 10^{-8}. The artifact visible in fuse voltage, as illustrated in Figure 4, indicates a numerical error rather than an actual system response.

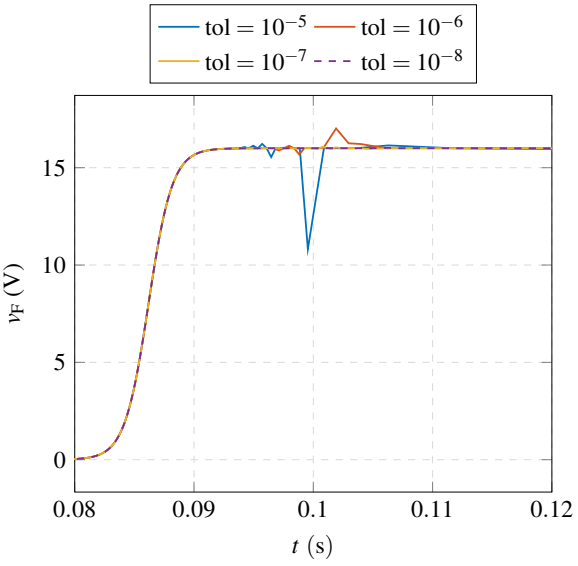


Figure 4. Instability in fuse voltage across different relative error tolerance values.

Such a sharp, triangular deviation is not characteristic of the passive electrical elements under a continuous switching function. It is shown to be a consequence of numerical inaccuracy, as reducing the relative error tolerance by one order of magnitude to 10^{-8} eliminates any visible disturbance in the signal. In order to see what causes such

a spike after the voltage transition seemingly ends, an additional plot was generated that marks every simulation step, as depicted in Figure 5.

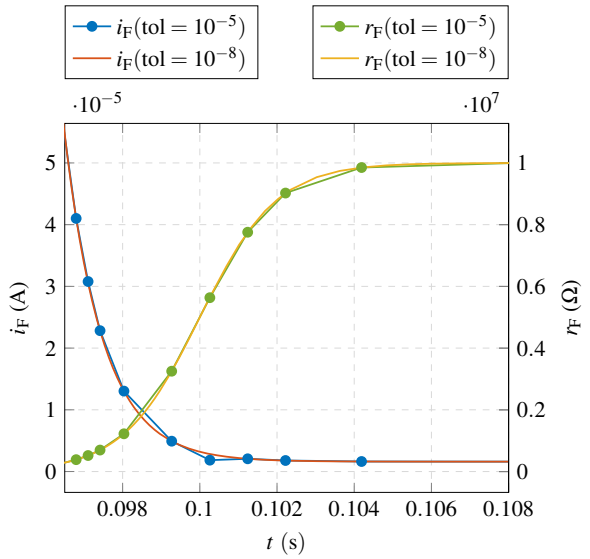


Figure 5. Current and resistance of the variable resistor across different relative tolerance values are plotted. Additionally, the solver steps for tol $= 10^{-5}$ are marked.

As the fuse resistance keeps rising and reaching 5 MΩ, the voltage approaches the value of the source voltage $V_{\text{in}} = 16\,\text{V}$. The "off" current is already in the order of microamperes, which means that the relative error tolerance is insufficient to maintain calculation accuracy. Further calculations in the integration algorithm for the ODE problem involve both very small (µA) and very large quantities (MΩ). However, the disparity in the orders of magnitude between these quantities induces a significant error amplification, ultimately resulting in voltage deviations in the order of several volts. This deviation can be seen in the fuse current plot, where i_F is a result of simulation using relative tolerances of tol $= 10^{-5}$ and tol $= 10^{-8}$. At time $t = 0.100256\,\text{s}$, the current value at tol $= 10^{-5}$ is deviating from the "expected" current at tol $= 10^{-8}$, leading to an error magnification to the voltage level. Using a smaller value for the relative tolerance helps keep the step size small enough to avoid any error escalation. Only at tol $= 10^{-8}$ is the inaccuracy not present.

Equations 4 to 11 present the causal system of equations for the model generated by Dymola. The integration inaccuracy is evident in the voltage signals of both inductors (v_{L_1}, v_{L_2}) as well as in the voltage across the fuse (v_{r_F}) in the circuit shown in Figure 3.

When the error occurs in the integration of the differential equation of the current in Equation 9, it is propagated through the calculations of all the voltages, including the negative potential on R_1 in Equation 7. These voltage values are subsequently fed back into the derivative of the current in Equation 9, which can further exacerbate the error.

$$v_{R_2} = L_2 \cdot i \tag{4}$$
$$v_{r_F} = r_F \cdot i \tag{5}$$
$$v_{R_1} = L_1 \cdot i \tag{6}$$
$$v_{R_1}^{-} = V_{in} - v_{R_1} \tag{7}$$
$$v_{R_{\text{load}}} = R_{\text{load}} \cdot i \tag{8}$$
$$\frac{\mathrm{d}i}{\mathrm{d}t} = -\frac{1}{L_1 + L_2}\left(v_{R_2} + v_{r_F} - v_{R_1}^{-} + R_{\text{load}}\right) \tag{9}$$
$$v_{L_1} = L_1 \cdot \frac{\mathrm{d}i}{\mathrm{d}t} \tag{10}$$
$$v_{L_2} = L_2 \cdot \frac{\mathrm{d}i}{\mathrm{d}t} \tag{11}$$

It shall be noted that, although this is a simple example, the same effect can be observed in more complex circuits, e.g. in the model below and generally in the vehicular power system simulations as well. The model developed by Baumann, Abouzari, et al. (2023) is a slightly expanded version of the previously shown model in Figure 3. In this model, additional consumers are added to the single supply line (with one fuse), each equipped with individual input circuits and resistive load elements, as illustrated in Figure 6.

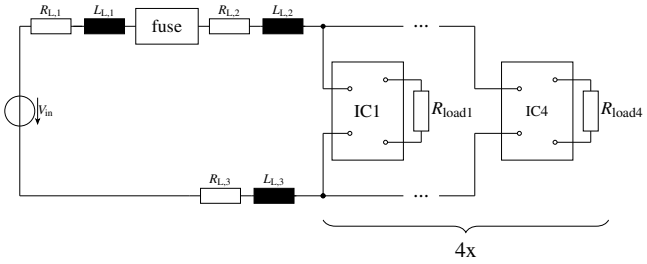


Figure 6. Simple multi-load model with a source and one resistive fuse element, interconnected with cables. Each resistive load has an input circuit (IC) block.

This time the fuse variable resistor is controlled with a more advanced dynamic switching function, developed by Baumann, Abouzari, et al. (2023). The fuse is switching the load "off" from the supply at $t = 0.1\,\text{ms}$. As shown in Figure 7, the numerical accuracy is present even after the significant change in fuse voltage. A very small current and a very large changing resistance are unfavorable for the integration algorithm. When the relative and, consequently, the absolute tolerance of the simulation are reduced, the errors observed in the previous case are resolved.

However, such a low relative tolerance value is too low for the highly stiff large-scale systems required in the design of vehicular electrical systems. It can cause a complete simulation failure during switching or even at the very start of the simulation. Hence, an alternative solution was explored that does not involve a direct limitation of the global relative tolerance.

DOI
10.3384/ecp218961

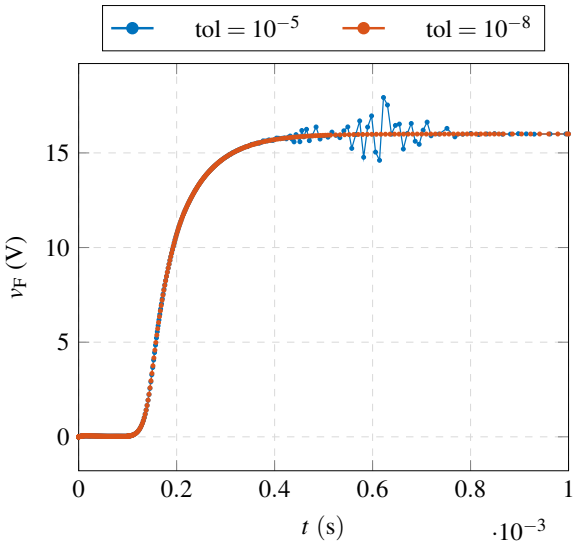


Figure 7. Voltage response for v_F across the fuse at $\text{tol}_{\text{rel}} = 10^{-5}$ (default) and $\text{tol}_{\text{rel}} = 10^{-8}$.

4 Simulation Approaches for Switched Electronic Fuses

In the following, a signal and a physical filtering approach is proposed to mitigate instability. The variable resistor is remodeled and the circuit is split, yielding an equivalent model referred to as decoupling. This serves as a necessary preparatory step for the implementation of the filtering approaches.

4.1 Decoupling

The main idea behind the decoupling is to use controllable current and voltage sources. This allows for the implementation of a custom control function rather than using a variable resistor. Its resistance directly affects the system matrix and variations in its value modify the system eigenvalues. Therefore, an indirect way of securing the same voltage behavior was required to construct an equivalent system with the use of the closely available physical quantities. Among the decoupled attempts with different source combinations, the best implementation was found in the combination of the controlled current source on the primary (supply) side and a controlled voltage source on the secondary (output or load) side. This configuration is presented in Figure 8.

Equations (12) and (13) were implemented to control the respective sources using the mathematical expression for the desired resistance change of the fuse, the measured secondary current $\hat{i}_y$ and the measured primary voltage $\hat{v}_x$. The measured current from the secondary circuit is fed to the current source on the primary side, ensuring identical conditions as in the original model, whereas the secondary voltage was controlled indirectly via the shown expressions. Voltage was measured between the positive nodes of the two decoupled sides v_{xy} to obtain the equivalent of

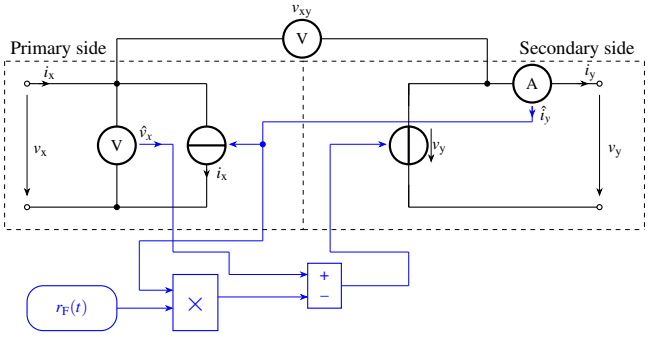


Figure 8. Decoupling implemented with a combination of a controlled current and voltage source (CV).

the fuse voltage. The notation CV will be used to denote this implementation in the subsequent text.

$$v_y(t) = \hat{v}_x(t) - \hat{i}_y(t) \cdot r_F(t) \tag{12}$$

$$i_x(t) = \hat{i}_y(t) \tag{13}$$

An option that was explored as well included two controlled voltage sources (further denoted as VV), whose inputs are shown in Equations (14) to (16).

$$v_y(t) = \hat{v}_x(t) - \hat{i}_x(t) \cdot r_F(t) \tag{14}$$

$$v_x(t) = \hat{v}_y(t) + \hat{i}_y(t) \cdot r_F(t) \tag{15}$$

$$i_x(t) = i_y(t) \tag{16}$$

This approach has certain disadvantages due to its structure. The most significant problem is the occurrence of a division by zero in one of the differential equations of the simulation model. Therefore, to avoid the division by zero a correction factor is defined to only slightly differ from the original factor in the addition block. However, this very small value in the denominator had a negative impact on the stability of the model. Apart from these VV drawbacks, no other significant differences between the two approaches that could affect the stability of the model were observed.

Since the secondary current is measured and fed to the primary, the sensitive state can be accessed directly. Due to the previous comparison of the two approaches, the decoupled CV was subsequently used for further investigation.

4.2 Signal Filtering

Decoupling the system offers a significant advantage of transforming a purely electrical model into a signal-based switching system. This approach enables modifying the signals for both the controlled signal current on the primary side and the controlled signal voltage on the secondary side. Since it is known that instabilities arise from even minimal inaccuracies in the current, which are amplified by large resistance values, the idea was to smoothen the entry to the controlled signal voltage. This can be achieved by employing a filter, as shown in Figure 9.

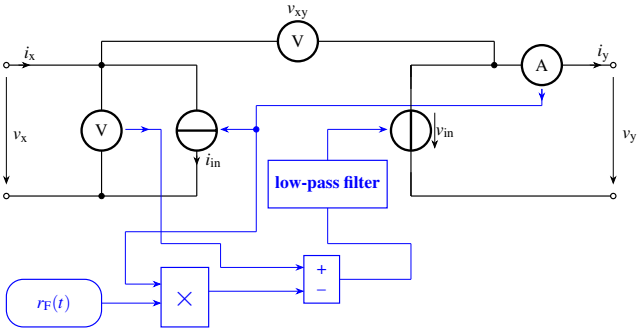


Figure 9. Decoupled model with an integrated low-pass filter.

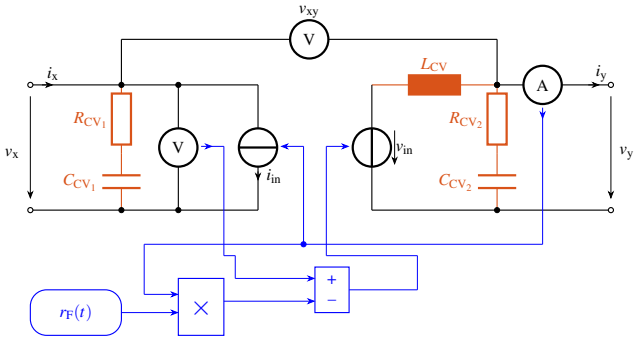


Figure 10. Decoupled model with an implemented physical filter approach. The components needed for the filter are highlighted in orange.

Such an implementation can achieve a smoother transition during switching operations. When it comes to the primary side, filtering the current directly is neither necessary nor effective, as the current response itself has no evident sawtooth disturbances. Additionally, primary and secondary currents must remain equal, in order to retain the equivalence to the original circuit (Equation 13).

Various strategies can be employed to smoothen the signal input for the secondary voltage source, with a first-order element serving as the simplest form of signal-based filtering (Equation 17).

$$\tau \cdot \frac{y_{\text{filter}}(t)}{\mathrm{d}t} + y_{\text{filter}}(t) = K \cdot u_{\text{filter}}(t) \tag{17}$$

It is crucial to have a precise understanding of the expected voltage behavior in the circuit before implementing any filters to avoid suppressing significant electrical effects. Additionally, it must be noted that such components influence the system dynamics by introducing a delay and altering the signal's response based on the chosen time constant. It is advantageous to select a very small time constant (e.g., $\tau < 10^{-7}$ s) to minimize switching delays and preserve the original electrical characteristics of the signal. The effects of different time constants are discussed in more detail in Chapter 6. The preset gain factor should be set to $K = 1$.

4.3 Physical Filtering

In addition to the previously introduced signal filter, a physical filter approach is presented. The most promising variant included the RC branch parallel to the controlled sources and an additional inductor on the secondary side. This configuration is illustrated in Figure 10.

Table 1. Parameter values used in the decoupled CV with inaccuracy compensation.

Primary side		**Secondary side**		
R_{CV_1}	C_{CV_1}	R_{CV_2}	C_{CV_2}	L_{CV}
$10^4\,\Omega$	$10^{-9}\,\mathrm{F}$	$10^4\,\Omega$	$10^{-9}\,\mathrm{F}$	$10^{-9}\,\mathrm{H}$

This circuit has shown the ability to compensate the numerical error and provide accurate results for the example in Figure 11. The parameters of the additional circuit elements are set to very high resistance values and very low inductance and capacitance values, so as to avoid introducing undesired dynamics into the system. A parametrization example is shown in Table 1.

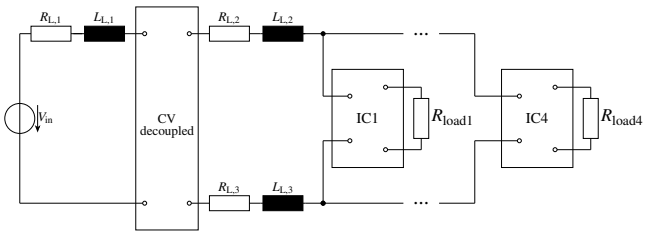


Figure 11. Power system model with decoupled block.

When this implementation is applied to the simplified system with multiple loads with their input circuits, the behavior was stable and accurate. However, if the circuit tends to be predominantly inductive, the shown solution still exhibits inaccuracies. One such example is the small model from Figure 3. Its inductive characteristic causes numerical instability with every possible RC and RLC combination, except in the case with the RC branch only on the secondary side, without an inductor, where the instability is only reduced but not eliminated. Since the RLC components of the physical approach may also represent parasitic effects of the wiring harness, it is possible to investigate whether a more detailed modeling of the wiring harness could lead to a more accurate simulation and reduce instabilities. This could be addressed in future research.

5 Elimination of Instabilities by Focusing on the Absolute Tolerance

A *nominal* value indicates the magnitude of a specific state and influences the absolute tolerance. This nominal value is specified in the textual description of the component (*Modelica - Language Specification* 2023).

Listing 1 shows an example in which the current i of an inductor is assigned a nominal value of 10^{-4}.

Listing 1. Nominal value definition for a component

```
L2(i(nominal=1e-4), L=0.75e-6)
```

Equation 18 demonstrates how error tolerance is effectively distributed using this setting. If the value of the state itself is higher than the defined nominal value, the relative tolerance will dominate the error calculation. Conversely, a higher nominal value shifts the dominance toward the absolute tolerance. But what does it mean in practice?

$$|\text{local error}| < \text{tol} \cdot (|x_i| + \text{nominal}(x_i)) \tag{18}$$

"A simple way of understanding tolerance is that the relative tolerance sets the number of significant digits to be calculated using:

$$\text{significant digits} = -\log_{10}(\text{tol}),$$

(when $|x_i|$ is larger than the nominal value). And the absolute tolerance sets the number of decimal places:

$$\text{decimal places} = -\log_{10}(\text{tol} \cdot \text{nominal}(x_i)),$$

(when $|x_i|$ is less than the nominal value)" (Fish 2019).

Therefore, it is desirable to use a nominal value of the same order of magnitude as the lower value of the state in which the inaccuracy occurs. Decimal places are used for large values and significant figures for very small ones. It ensures that the step size remains sufficiently small for precise calculations when the state falls below the nominal. This is a crucial point in these switching models because exactly when the current becomes too small for the integrator to keep the accuracy, the large resistance comes into play. This way, the strict accuracy mode is "turned on" only when the state is below the nominal value. Otherwise, the system follows the relative tolerance constraints.

An investigation of the small model in Figure 3 with preset relative tolerance of 10^{-5} reveals that defining an appropriate nominal value for the state leads to immense improvements in the voltage characteristics. This change is illustrated at the bottom of Figure 12.

The simulation analysis performed in Dymola provided access to information on integrator order, step size, and various error indicators and markers. In the upper plot of Figure 12, this small example shows that using the nominal setting (dashed red) results in an extended period of reduced step size compared to the default (dashed blue), thereby enabling more accurate calculations at the end of the switching period. This change is reflected in the integration order remaining at higher values over that extended duration. The step size is reduced as a consequence of more frequent step rejections compared to the simulation with the default settings, as shown by the dots in Figure 12.

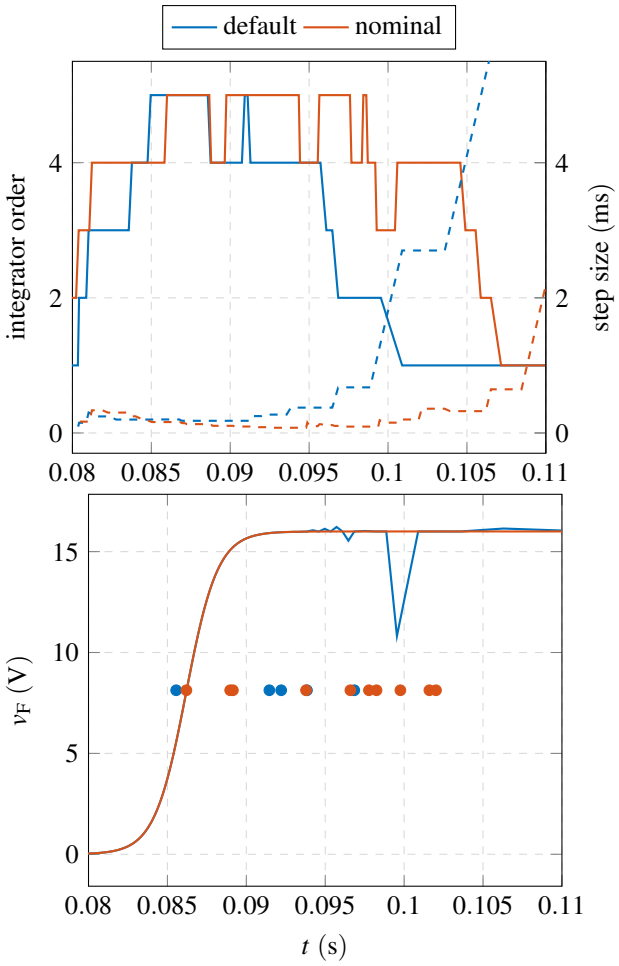


Figure 12. Numerical analysis of the small model without (default, nominal = 1) and with a nominal value of 10^{-4}. The upper plot shows the integration order in full line and step size comparison in dashed. The plot below depicts the change in fuse voltage as well as the solver rejected steps as full dots.

6 Results and Discussion

To evaluate the performance of these approaches, two simulations are presented. Firstly, the results from the small-scale power system model are shown, followed by those from a large-scale simulation. Finally, the similarities and differences between the results of these approaches are compared.

6.1 Small-Scale Power System Simulation Model

This vehicle electrical system model was previously introduced in Figure 6. The model consists of four loads with integrated input circuits, one fuse, and a corresponding wiring harness. The fuse is switched off at $t = 0.1\,\text{ms}$ changing its resistance from 1 mΩ to 1 MΩ. The function describing the change in resistance during switching off was developed by Baumann, Abouzari, et al. (2023). As a consequence of switching the fuse, the voltage over the resistor changes from 28 mV to 16 V. Preset tolerance of the simulation is $\text{tol}_{\text{rel}} = 10^{-5}$.

Figure 13 displays the voltage across the fuse resistor for the default simulation (shown in blue). Additionally, the previously introduced methods for physical filtering (yellow) and signal filtering (violet), as well as the nominal value (orange), are visible. The parameters for the physical filtering are the same as in Table 1. For signal filtering, a time constant of $\tau = 10^{-8}$ s is used. The trace of the default simulation shows instabilities occurring between 0.4 ms and 0.8 ms. These sawtooth disturbances exhibit a maximum voltage deviation from the equilibrium state of 10.71%. As visualized in the zoomed-in area (**A**) of Figure 13, the simulation results indicate that both filtering approaches and the nominal setting produce identically smooth voltage outputs.

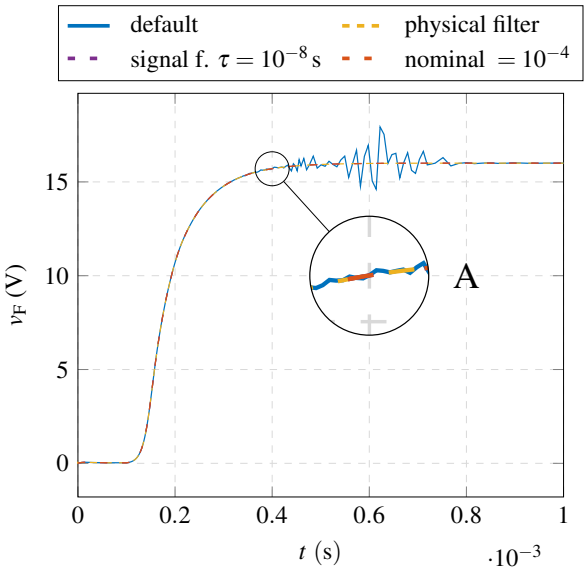


Figure 13. Differences in voltage response v_F across the fuse of the models from Figure 6 and Figure 11 at $\text{tol}_{\text{rel}} = 10^{-5}$ (default).

Further tests have shown that a sufficiently small time constant is $\tau < 10^{-6}$ s. Since the fuse switches within a time interval of up to 10^{-5} s, the first-order filter can effectively resolve numerical inaccuracies without introducing significant delays in the voltage value. In addition, no negative side effects of the approach can be identified. For the physical approach, adding the RLC and RC branches before and after the variable resistor is also a viable option, yielding results similar to those of the decoupled system with a signal filter. The filter CV models typically do not exhibit a reduced step size compared to the original model, as the filtering elements do not dominate the solver's error estimation. Defining the nominal state value (highlighted in orange) further improves stability at minimal implementation cost, without requiring detailed knowledge of the circuit parameters. The only information necessary to properly set the nominal value is the state that causes numerical disturbances and the precision required to mitigate them. The nominal attribute employs a stricter error tolerance for very small currents. In this example, the approach results in more rejected steps and a smaller step size, as demonstrated in the previously shown example, illustrated in Figure 12.

6.2 Large-Scale Power System Simulation Model

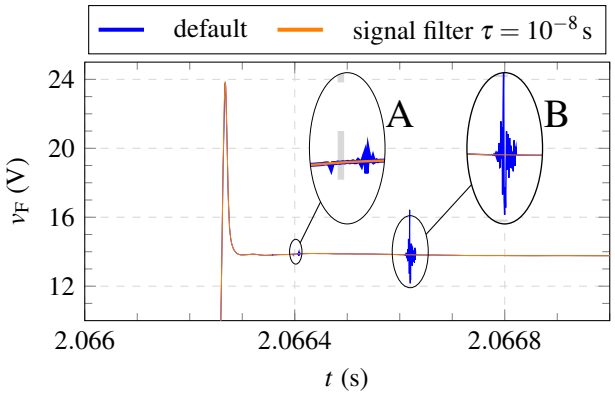
In the following, the stability of the approaches is evaluated using an electrical power system simulation model developed at BMW. This full model includes all components of the vehicle's electrical consumers, a battery model, the entire wiring harness, and the above-mentioned fuses. In addition, each component contains complex control and monitoring logic. The simulation model is comprised of approximately 1,800 states and about 230,000 equations. In this simulation at $t = 2$ s, a short circuit is initiated, causing the fuse to switch off in accordance with the characteristics described in Chapter 6.1. Despite the preset tolerance ($\text{tol}_{\text{rel}} = 10^{-7}$) being relatively strict, instabilities are still observed. The fact that we observe such disturbances, even with the tight tolerance, indicates that they occur in almost every power system simulation after switching the fuse. Vice-versa, this contradicts the findings in Figure 5 that instabilities are no longer visible at a preset tolerance of $\text{tol}_{\text{rel}} = 10^{-7}$ or stricter. Investigations show that the potential of the deviations is reduced with a tighter tolerance, but it is not guaranteed.

Figure 14 shows the voltage across the fuse resistor, where the same signal filtering technique is applied using two different time constants. Furthermore, the physical approach is presented using the same parameter set as in the initial model described in Table 1, while the nominal approach again employs a value of 10^{-4}. During the switching of the fuse, the voltage across it leads to a voltage peak due to the up- and downstream inductors at time 2.066 26 s. After the voltage peak, the instabilities occur at 2.066 32 s and 2.0668 s. The maximum deviation is about 15.89 % at 2.066 62 s. As visualized in the zoomed-in areas (**A**) and (**B**) in Figure 14a, the characteristic sawtooth disturbances are consistent with those observed in Figure 4.

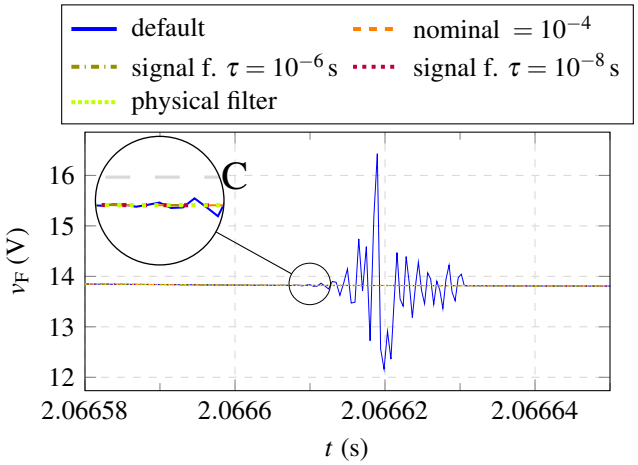
The signal filter curve ($\tau = 10^{-8}$ s) demonstrates significant improvements in stability at the time when the inaccuracies in the default simulation model arise. These enhancements occur without any observable change in the electrical system dynamics. Figure 14b presents the zoomed voltage traces and illustrates both the instability characteristics and the results of the implemented approaches. Overall, these simulation results indicate that all introduced and applied methods have effectively eliminated the observed instabilities. Compared to the small model filtering result ($\tau = 10^{-6}$ s), the large model result ($\tau = 10^{-8}$ s) also does not exhibit any observable differences after the voltage peak. The effects of different signal filter time constants throughout the complete simulation duration are explored next.

Figure 15 illustrates the same simulation with default settings alongside the signal filtering applied with two dif-

DOI
10.3384/ecp218961



(a) Voltage across the fuse resistor in the large-scale simulation model, comparing the original result with that obtained after applying the filtering approach.



(b) Enlarged illustration of the voltage instability across the fuse resistor in a large-scale simulation model and the results for all the approaches presented to eliminate it.

Figure 14. Voltage v_F across the fuse resistor in a large-scale simulation model and the results of the previous introduced approaches.

ferent time constants at $t = 2\,\text{s}$, where a maneuver is initiated. This occurs before the fuse is switched off. In this context, maneuvers refer to vehicle-specific events that change the load on consumers, such as electric power steering. The large-scale simulation shows that a large time constant (definition of "large" relative to the simulation model) leads to oscillations starting at $t = 2\,\text{s}$ with a maximum amplitude of 0.21 V. These oscillations can reduce the step size and affect simulation stability. Therefore, the time constant for the filtering approach must be carefully selected.

6.3 Discussion of the Approaches

The implementation of the previously introduced approaches in the two simulation models, which differ significantly in numerical complexity, revealed notable variations in the simulation results. The following discusses these similarities and differences in detail.

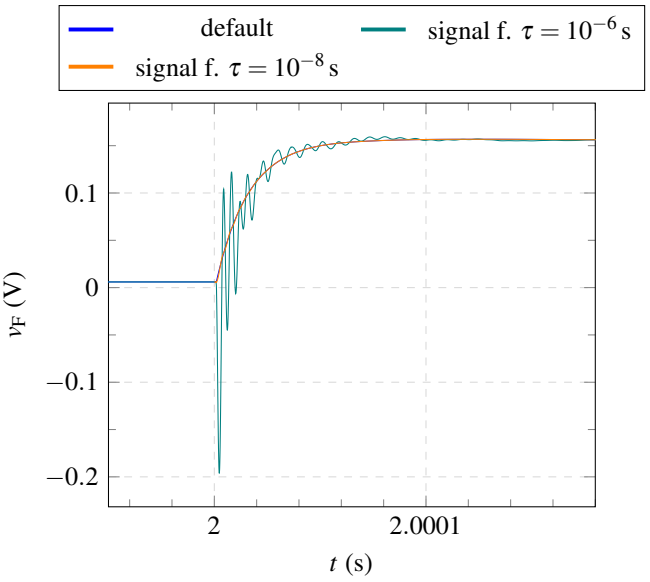


Figure 15. Effects of the signal filter with different time constants on the voltage across the fuse resistor at the start of the maneuver in the power system simulation.

6.3.1 Nominal Approach

The nominal approach is attractive due to its straightforward implementation. The simulation results indicate that the instability issues were minimized. Residual ripples may persist, but they typically do not affect simulation stability. The models presented above generally exhibit different behavior with regard to step size. While in the first small-scale simulation the step size is reduced – thereby increasing the simulation time – the large-scale model demonstrates a reduction in simulation time. This effect in the large-scale model is due to the constantly low step size, caused by the remaining states, as well as state and time events of the overall simulation model. As a result, the step size is controlled in a different manner compared to the small-scale model, which can positively affect the simulation time. However, this relationship cannot be determined in advance, as it depends on the individual model and its dynamics. As already mentioned in Chapter 5, the nominal value should be chosen to be of the same order of magnitude as the low state value at which the inaccuracy occurs.

6.3.2 Signal Filtering Approach

By decoupling the system, continuous controllability of the voltage was achieved throughout the simulation. In both models, the implementation of a first-order filter in the voltage path resulted in significant improvements by smoothening the voltage. Importantly, overall simulation performance remained unchanged compared to the original simulation, regardless of the time constant employed. However, large time constants that depend on the simulation model lead to oscillations in both voltage and current. In particular, the current as a state variable further compromises simulation performance and stability. These ob-

servations underscore the critical importance of carefully selecting the time constant in the filtering approach. For the demonstrated simulation scenarios, it is recommended to select time constants τ between 10^{-7} s and 10^{-9} s. Previous investigations have shown that these values do not cause any noticeable negative effects.

6.3.3 Physical Filtering Approach

Physical filtering is another approach in which different states are integrated before and after the switching element. By using a high resistance and a low capacitance and inductance, instabilities can be eliminated without significantly changing the dominant error or the step size. Both simulation models presented earlier, despite their differences in complexity, have demonstrated that this approach reliably eliminates instabilities. However, the natural delay in the RC circuit in an electrical system can lead to oscillations if the parameters of the components are chosen improperly. When using this approach, it is recommended to employ the previously used parameter values for inductance, resistance, and capacitance. If necessary, these parameters may be reduced by an order of magnitude without compromising stability.

7 Conclusion

A decoupling approach with filters has been introduced as a substitute for the challenging variable resistor element under highly dynamic switching in stiff electrical power system models. On the other hand, an alternative error tolerance setting – nominal – has been explored and compared to the default settings. Decoupling is essential for the signal filtering approach, as it enables the removal of numerical disturbances in a manner analogous to physical filtering. This strategy provides direct access to the perturbed voltage value, which can then be stabilized. Both filter variants successfully achieve their intended purpose without reducing the global relative tolerance. However, the first-order block in the signal variant is considerably simpler to implement. With a sufficiently small time constant, it is also more robust against unwanted voltage instabilities. The application of the physical filtering solution is the more complex method, which involves tuning of the RLC and RC branches to achieve the desired output.

It can be concluded that the fastest and most reliable solution to the problem is the use of the nominal setting in the problematic state, in the magnitude of its low value. However, this method still exhibits a minor error, although in the order of less than a percent of the expected value. The authors advise a thorough understanding of the circuit characteristics, as it is essential to identify the problematic state and subsequently implement a suitable approach.

For future studies, it is recommended to investigate how the various approaches with different parameter sets affect the runtime of the models. Furthermore, it is important to understand the performance implications for different fuses in the power system simulation, given its complexity and size.

An alternative approach to achieve the desired voltage-switching characteristic is to use a series-connected controllable voltage source instead of a variable resistor. This configuration replicates the voltage-switching behavior of the electronic fuse rather than relying solely on resistance. Moreover, this approach avoids dependency on the current and high resistance by applying a direct voltage requirement. This solution requires further research and testing to determine potential disadvantages.

References

Ardourel, Vincent and Julie Jebeile (2021). "Numerical instability and dynamical systems". In: *European Journal for Philosophy of Science* 11. DOI: 10.1007/s13194-021-00372-7.

Baumann, Martin, Ali Shoar Abouzari, et al. (2023). "Resource-Saving Modeling of an Electronic Fuse in Vehicular Power Systems". English. In: *2023 IEEE Vehicle Power and Propulsion Conference, VPPC 2023 - Proceedings*. 2023 IEEE Vehicle Power and Propulsion Conference, VPPC 2023 - Proceedings. Institute of Electrical and Electronics Engineers Inc. DOI: 10.1109/VPPC60535.2023.10403182.

Baumann, Martin, Christoph Weissinger, and Hans-Georg Herzog (2019). "Reducing Transient Disturbances within Automotive Power Systems Through Adapting of Input Circuits". In: *2019 International Conference on Computing, Electronics & Communications Engineering (iCCECE)*, pp. 214–218. DOI: 10.1109/iCCECE46942.2019.8941977. (Visited on 2024-11-28).

Blom, David S. et al. (2016). "A comparison of Rosenbrock and ESDIRK methods combined with iterative solvers for unsteady compressible flows". In: *Advances in Computational Mathematics* 42.6, pp. 1401–1426. ISSN: 1572-9044. DOI: 10.1007/s10444-016-9468-x.

Cellier, Francois E. and Ernesto Kofman (2006). *Continuous System Simulation*. Springer. ISBN: 978-0-387-26102-7.

Cohen, Scott D., Alan C. Hindmarsh, and Paul F. Dubois (1996). "CVODE, A Stiff/Nonstiff ODE Solver in C". In: *Computer in Physics* 10.2, pp. 138–143. ISSN: 0894-1866. DOI: 10.1063/1.4822377.

Dymola Full User Manual (2022). Dassault Systèmes AB. Lund, Sweden. URL: https://www.dymola.com/ (visited on 2025-01-05).

Fish, Garron (2019). *A closer look at solver tolerance*. URL: https://www.claytex.com/tech-blog/a-closer-look-at-solver-tolerance/ (visited on 2024-12-04).

Lambert, J.D. (1992). *Numerical Methods for Ordinary Differential Systems: The Initial Value Problem*. Wiley. ISBN: 978-0-471-92990-1.

Liu, Liu, Felix Felgner, and Georg Frey (2010). "Comparison of 4 numerical solvers for stiff and hybrid systems simulation". In: *2010 IEEE 15th Conference on Emerging Technologies & Factory Automation (ETFA 2010)*, pp. 1–8. DOI: 10.1109/ETFA.2010.5641330.

Modelica - Language Specification (2023). Modelica Association. URL: https://modelica.org (visited on 2025-10-04).

Schwimmbeck, Stefan, Quirin Buchner, and Hans-Georg Herzog (2018). "Evaluation of Short-Circuits in Automotive Power Nets with Different Wire Inductances". In: pp. 1–6. DOI: 10.1109/ESARS-ITEC.2018.8607795.

Requirement Verification with CRML and OpenModelica

Lena Buffoni[1] Adrian Pop[1] Audrey Jardin[2]

[1]Department of Computer and Information Science, Linköping University, Sweden,
{lena.buffoni,adrian.pop}@liu.se
[2]R&D, Electricité de France, France, audrey.jardin@edf.fr

Abstract

Common Requirement Modeling Language (CRML) is a language designed to express requirements in an intuitive manner, in this paper we present the recent advancements in tool support for the requirement modeling and verification workflow in OpenModelica and illustrate this on the Traffic Light use-case.
Keywords: CRML, Requirements, Verification, Complex Systems, Digital Engineering

1 Introduction

Systems Engineering is *a transdisciplinary and integrative approach to enable the successful realization, use and retirement of engineered systems* (INCOSE Fellows 2019).

Thanks to an approach centered on requirements, its main benefits lie in better consideration of the system's environment (in terms of coordination of the various stakeholders, support of associated constraints and interfaces with other systems). It also provides an iterative approach making it possible to guide choices towards solutions "just needed" throughout the life cycle of the system. Impact analyses of some changes during the system development cycle (e.g. coming from regulatory tightening, partner entry and/or withdrawal, etc.) can be performed early in the design process and knowledge can be better capitalized to refine the product strategy in its whole from one implementation project to another.

Despite numerous success stories, in particular, in the most advanced systems engineering industries such as defense and aerospace, many industries still struggle to implement systems engineering practically (Bretz, Kaiser, and Dumitrescu 2019; Kiniry et al. 2022).

In addition to budgetary and socio-organizational difficulties linked to change management, one of the major pitfalls often encountered is to limit the management of requirements to the documentary aspect. Technical contributors then perceive requirement management as an additional task to their usual engineering activities and with a relatively poor added value at the very limited time scale of their responsibility in the project compared to the lifespan of the system or even the company. In the worst case, this can generate a certain disillusionment and at least partial disengagement of some of the contributors.

When verification and validation of requirements are taken into account correctly in the organization and in the projects, the second flaw lies in the difficulty of finding "off-the-shelf" software solutions to handle requirements "from end to end" during the system lifecycle. In particular, beyond documenting the traceability of requirements and their theoretical correspondence to design elements (such as requirement management tools like IBM Doors, Polarion, etc.), using requirements as a guide for the development process requires the ability to capitalize this kind of industrial constraints in the form of models to make this knowledge usable by a computer (i.e. calculable) and to allow the automation of a certain number of tasks such as checking the consistency of a set of requirements, verifying the compatibility of a solution with a set of regulations, studying the impact of changes in the specifications or in the design, etc.

As defined by INCOSE in (SEBoK v2.11 2024), a requirement is a "*statement that identifies a system, product, or process characteristic or constraint, which is unambiguous, clear, unique, consistent, stand-alone (not grouped), and verifiable, and is deemed necessary for stakeholder acceptability*" and six verification methods are usually acknowledged:

1. *Inspection* requires a human sense or a simple measurement method to examine the object studied once it exists.
2. *Test* consists in performing some verifications on the existing object, often with dedicated instrumentation, and under specific controlled conditions.
3. *Demonstration* checks the desired characteristics of the existing object under its *classical* operating conditions.
4. *Sampling* tests also characteristics, but not only on one realization of the object studied but on a sample of several realizations.
5. *Analogy* uses the similarity of the object studied with elements for which the desired characteristics have already been proven.
6. *Analysis* shows theoretical compliance of the object studied using analytical evidence based on mathematical reasoning (calculation, modeling and / or simulation).

The article focuses here on the verification method based on analysis, which is the only option one can have for checking objects early in their design process or when testing or demonstration in real-life conditions is not possible (e.g. for cost reasons) or even not desirable (e.g. for safety reasons). Several approaches already exist in the literature to formalize requirements and their verification such as LTL (Linear Temporal Logic) and CTL (Computation Tree Logic) (Baier and Katoen 2008) , timed (Alur 1999) and hybrid (Henzinger 2000) automata or UML/SysML state behavioral diagrams (OMG 2017; OMG 2025b). (Bouskela, Buffoni, et al. 2023) explains why existing solutions are not really suitable for complex systems with strong physical aspects, the origin of the Common Requirement Modeling Language (CRML) from ITEA MODRIO project previous work's on the FORM-L language and how such a new language separate from Modelica could be a possible solution. In short, the existing formal requirements modeling methods have the following disadvantages: they lack of object-orientation, have difficult mathematical syntax, often do not deal with probabilistic aspects and, most importantly, tend to use abstractions of the system in the form of state machines, which already express a kind of a behavior and is not appropriate to correctly deal with systems with strong physical aspects that evolve continuously over time.

The novelty of the current paper is to show the recent progress made for supporting CRML in OpenModelica and how this new feature can practically help newcomers automating the verification of some temporal requirements with the use of Modelica/FMI (or more largely black box) models.

Section 2 briefly presents the CRML language and the associated methodology to verify requirements through the simulation of solution models. Section 3 provides details on how the continuous- and discrete-time domains are handled to express dynamic requirements in CRML. Section 4 introduces the Traffic Light System (TLS) used as a common thread example to illustrate the CRML approach. Section 5 describes the current tooling in OpenModelica. Section 6 concludes with some perspectives for improving the overall toolchain and its potential integration into a larger (Model-Based Systems Engineering) MBSE framework.

2 CRML in a Nutshell

CRML is a language for verifying temporal requirements on systems with strong physical concerns and for which the dynamical interactions with its environment and interfaced systems are of most importance.

The main goal of CRML is to be a high-level language enabling the definition of requirements in quite readable but computable expressions and serving as a pivotal format capable of targeting different verification engines either by simulation (as documented here with its integration in OpenModelica) or by formal proof (future work not documented here about connection with model checking approaches).

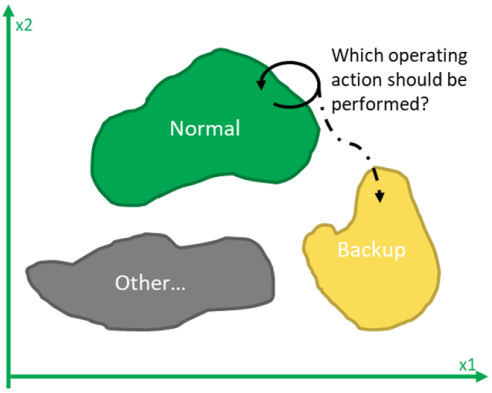


Figure 1. A typical realistic dynamic requirement.

The following paragraphs briefly recap how a requirement could be expressed and verified using CRML. Appendix gives a quick overview of the CRML built-in elements and most useful operators. Interested readers can refer to the online tutorial (Buffoni et al. 2023) and the language specifications (CRML v1.2 2023) for more details.

2.1 Requirement Expression

A requirement in CRML is defined as an expression combining up to 4 items:

- a **condition** to be checked;
- a **spatial locator** indicating on which object the condition has to be verified;
- a **time locator** defining when the condition has to be satisfied;
- (optionally) a **statistical target** to indicate with which performance the condition has to be satisfied.

A CRML requirement is therefore of a "special" Boolean built-in type, here called Boolean4 stating that a requirement value at instant `t` can vary among the set `{true, false, undefined, undecided}`. This is illustrated with the Traffic Light System example of Section 4. The theoretical rationale for introducing these 4 values could be found in (Bouskela and Jardin 2018) where a new temporal logic algebra is defined to rigorously compute the value of requirement over continuous-time periods and to enable the combination of requirements together.

One may consider that checking requirements for physical systems always amounts more or less to checking whether main variables of interest do not exceed thresholds. The main difficulties that make the handling of requirements more complex are as follows:

DOI 10.3384/ecp218971

Table 1. CRML pattern for a typical realistic dynamic requirement.

Req.	*Natural Language*	*CRML Formalization*
r1	During operation, the system should stay within its normal domain.	`Requirement r1 is 'during' inOperation 'ensure' inNormalDomain;`

r2	If the system fails to stay within its operating domain, then it should not stay outside of its normal domain for more than x minutes.	`Requirement r2 is 'during' inOperation 'ensure' (not inNormalDomain 'implies' r2_outside);`
		`Requirement r2_outside is 'during' [inNormalDomain 'becomes false', inNormalDomain 'becomes false' + x mn] 'check at end' b;`
		`Boolean b is inOperation 'implies' inNormalDomain;`

r3	The system should not go outside its normal domain more than n times per year.	`Requirement r3 is 'count' ((b 'becomes false') on [b 'becomes false', b 'becomes false' + 1 year]) <= n;`
r4	If (r1 and r2 and r3) fail, then the system should go to its backup domain within y minutes as soon as the failure is detected.	`Requirement r4 is not (r1 and r2 and r3) 'implies' 'during' [(r1 and r2 and r3) 'becomes false', (r1 and r2 and r3) 'becomes false' + y mn] 'check at end' inBackupDomain;`
R	(**The "real" complete requirement:**) During system operating life, r1 and r2 and r3 and r4 should be satisfied with a probability of success of p.	`Real prob is estimator Probability (r1 and r2 and r3 and r4) at inSystemOperatingLife 'becomes false';` `Requirement R is 'during' inSystemOperatingLife 'check at end prob' > p;`

- The authorized values vary dynamically depending on the system operating modes and the periods of time.
- Time response and/or probabilistic criteria should be added to make the requirements achievable by real systems.
- The number of requirements is rapidly growing (due to the number of stakeholders, the complexity of the system/subsystems/components, and the possible evolution over the system lifespan).

The CRML language enables alleviating these problems by allowing the formal expression (Table 1) of realistic dynamical requirements (Figure 1).

Listing 1. Example of user-defined operator in CRML

```
Operator [ Periods ] 'during' Boolean b =
    Periods [ new Clock b, new Clock not b
    ];
```

The user can enhance the readability of CRML statements by defining his own operators with a chosen syntax simply by combining built-in operators. For example, the `during` operator in Table 1 has been defined as a time period constructor from a Boolean input (Listing 1). This `during` operator is part of a library known as FORM-L used in CRML to define time periods and conditions constructors that are frequently encountered (see Appendix for more examples of custom operators).

2.2 Requirement Verification

As stated above, a CRML requirement is a Boolean4 variable. Its value is: `true` (resp. `false`) if the condition is satisfied (resp. violated) for the defined time locator, `undefined` if the time locator has not been tested over the test scenario, `undecided` if the test scenario has finished before a decision could be made (i.e., before the condition has been violated or before the end of the time locator).

The value of a CRML requirement is computed from verification models made of four different kinds of model (Figure 2):

- the *behavioral model* capturing the behavior of design solutions;
- the *requirement model* defining envelopes of acceptable behaviors;
- the *binding model* mapping the variables of the two previous models and processing the required conversions (the functional variables mentioned in the requirements could have different naming convention and units than the physical quantities computed in the behavioral model);
- the *test model* specifying the operating scenarios to be studied.

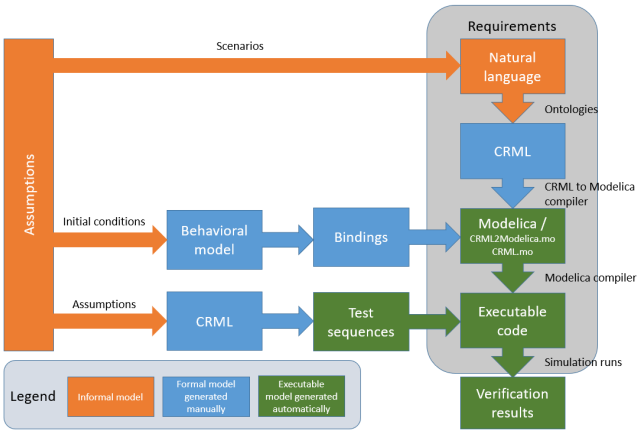


Figure 2. Requirement verification workflow.

3 Time Management in CRML

The main temporal constructs in CRML are Clock, Event, Period and Periods.

The CRML2Modelica library contains the definitions of basic types in CRML, including events and periods. These definitions are split into two parts, a record structure that can then be passed along an argument in subsequent equations, and a model containing the equations for the record.

In Modelica Events are represented by a trigger (a 4-valued boolean) and a time of occurrence.

A single period, as shown in Listing 2 is defined by opening and closing events, while a set of periods is defined by clocks or sets of opening and closing events (leading to possibly overlapping time periods).

For example, a new Period instance is generated and instantiated as follows:

Listing 2. Period instance creation in Modelica

```
record CRMLPeriod
 Boolean isLeftBoundaryIncluded "If true,
     the left boundaries of the time
     periods are included";
 Boolean isRightBoundaryIncluded "If true,
     the right boundaries of the time
     periods are included";
 public
  Types.Event start_event;
  Types.Event close_event;
  Boolean is_open;
end CRMLPeriod;

model CRMLPeriod_build
 CRMLPeriod P;
 equation
 P.is_open =
  if( (CRMLtoModelica.Functions.
      Event2Boolean (P.start_event) ==
      Boolean4.true4) and not (
      CRMLtoModelica.Functions.
      Event2Boolean (P.close_event)==
      Boolean4.true4))
  then true else false;
```


DOI
10.3384/ecp218971

```
end CRMLPeriod_build;
```

4 Traffic Light Use-Case

The purpose of the Traffic Light example is to illustrate how CRML could be used in the OpenModelica framework from the requirement formalization phase to verification by simulation and analysis phases.

This example was chosen to show, on a system known to all, how some realistic requirements could be handled in a very practical way with CRML and OpenModelica. It is deliberately simplified to focus on the software tooling part.

The mission of the Traffic Light System (TLS) is to avoid accidents and fluidize traffic by minimizing congestion. The main functions are, respectively: F1) give clear instructions to each vehicle and F2) prioritize crossing durations based on road traffic. The corresponding requirements could therefore check that *"at most only one light should be on"* and *"the time spent in a light mode must be between two limits"*. A possible formalization in CRML is given in Listing 3.

Listing 3. Requirements on the Traffic Light System

```
model TrafficLightSpecification is
 // Import of libraries
 flatten {ETL, FORM_L}
 union {
 // List of external variables
 Boolean red is external;
 Boolean yellow is external;
 Boolean green is external;

 Boolean operation is external;
 Boolean night_mode is external;
 Boolean day_mode is not night_mode;

 // Definition of requirements
 // req0: "During operation, no more than
    one light should be on (a flashing
    mode could be specified later)"
 Boolean{} lights is {red, yellow, green};
 Integer n_lights_on is card { filter
    lights (element == true) };

 Requirement req0 is
  ('during' operation) ('check anytime' (
     n_lights_on <= 1) );

 // Day_mode
 // req1: "After green, next step is yellow
    "
 Requirement req1 is
  (( 'after' ( green 'becomes true' ) '
     before' ( yellow 'becomes true' ))
     while day_mode)
  ('check count' (red 'becomes true') '=='
     0);

 // req2: "Step green should stay active
    for at least 30 seconds"
 Requirement req2 is
  (( 'after' (green 'becomes true') 'for'
     30) while day_mode)
  ('ensure' green);

 // req3: "After green becomes active + 30
    seconds, next step should turn yellow
    within 0.2 seconds"
 Requirement req3 is
  (( 'after' (green 'becomes true' + 30) '
     for' 0.2) while day_mode)
  ('check at end' yellow);

 // Night_mode
 // req4: "During night, yellow should only
    be used"
 Requirement req4 is
  ('during' night_mode)
  ('check count' ( (red or green) 'becomes
     true') '==' 0);
 // req5: "During night, yellow should
    flash every 2 seconds"
 Requirement req5a is
  ('during' night_mode)
  ('check count' ( yellow 'becomes true') '
     <>' 0);
 Requirement req5b is
  ('after' ( yellow 'becomes true') 'within
     ' 2) while night_mode)
  ('ensure' yellow);
 Requirement req5c is
  ('after' ( yellow 'becomes true') 'for'
     2) while night_mode)
  ('check at end' not yellow);
 Requirement req5 is req5a and req5b and
    req5c;
};
```

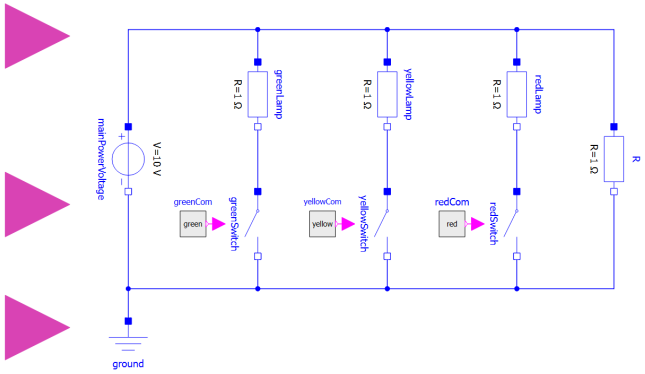


Figure 3. Traffic Light implementation as a circuit.

The physical behavior of TLS could be modeled as an electronic circuit that activates three different lights (Figure 3). The control part of TLS could be modeled in two different ways: either as a grafcet or as a finite-state machine using the synchronous built-in operators of the Modelica language as shown in Figure 4.

4.1 CRML Compilation

The CRML2Modelica compiler is a proof of concept compiler written in Java. It is open source and available on

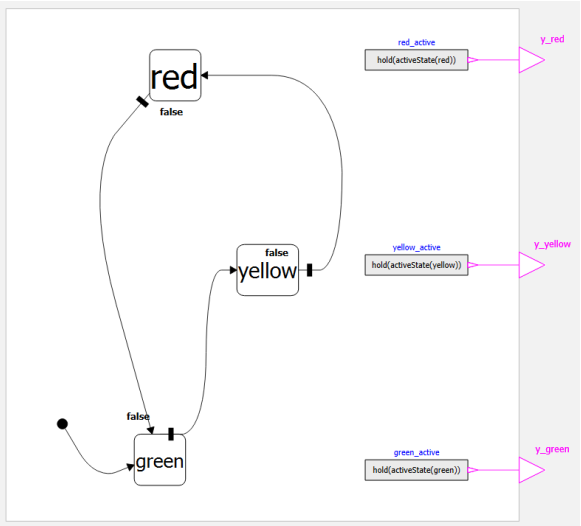


Figure 4. Traffic Light control implemented as a state machine.

GitHub [1]. Currently it supports almost all the Extended Temporal Library (ETL) library blocks which define all the operators required for computing Boolean4-status of a requirement.

The CRML2Modelica compiler goes through the CRML file once first, to retrieve all signatures of user-defined operators and category's definitions and then a second time translating all the definitions. In addition to temporal aspects, three CRML features require special handling: Boolean4, categories and sets.

Boolean4

Boolean4 is defined in Modelica as an enumeration as shown in Listing 4.

Listing 4. Boolean4 translation in Modelica

```
type Boolean4 = enumeration(undefined,
    undecided, false4, true4);
```

Categories

Categories are applied directly to the operator definitions, and the corresponding operators are replaced. For example, the operator `set to false` in Listing 5 is translated to Modelica as shown in Listing 6 and the operator `id` is replaced by `cte_false` as specified by the category `varying1`.

Listing 5. Category in CRML

```
Operator [ Boolean ]  'id'  Boolean b = b;
Operator [ Boolean ]  'cte_false'  Boolean
    b = false;
Category varying1 = { ('id', 'cte_false')
    };
Operator [ Boolean ] 'set to false' Boolean
     b = apply varying1 on ( 'id' b );
        ...
Boolean b_varying1_on_id_true is 'set to
    false' b_true; // value should be false
```

[1] https://github.com/OpenModelica/CRML

Listing 6. Category translation to Modelica

```
model 'set to false'
 output CRMLtoModelica.Types.Boolean4 out;
 input CRMLtoModelica.Types.Boolean4 b;
 'cte_false' 'cte_false0'(b=b);
 equation
 out ='cte_false0'.out;
end 'set to false';
```

Sets

Sets are handled as fixed size arrays in the current implementation, for sets with a known number of elements, the array size is set accordingly, for sets with an unknown size, the array size is set to a predefined buffer size. Periods are a special case of this and are currently mapped to a record with an array with a static size, but if a larger number of events is expected, the buffer size can be increased with a flag to the compiler.

In our Julia-based OpenModelica framework (Tinnerholm, Pop, and Sjölund 2022) we support automatic recompilation during simulation and one can increase/shrink the size of an array or change any other aspect of the model. An example of a `when` equation triggering such a change is given in Listing 7. Using this functionality in the future, we can make the translated Modelica code more flexible and more efficient as resizing can be tailored to the needs of each variable.

Listing 7. Dynamic Array Resize

```
when requestedSize > N then
  // Recompilation on parameter change
  recompilation(N /* what to change */,
      requestedSize + 100 /* value */);
end when;
```

4.2 Requirement Verification

A requirement model is verified for a behavior model and a scenario. In this paper we focus on Modelica, but as a requirement model can be exported as an FMU, it could be verified against any other FMU representing a behavioral model.

In order to identify the inputs from the behavioural model, CRML uses `external` variables, that are variables that do not need to be instantiated within the model. A list of external variables can be exported during compilation into a file. This list can then be fed into a semi-automated bindings algorithm for mapping requirements to the behavioural model.

A verification scenario that instantiates the requirement and behavioural models and connects them together can then be simulated and the results analysed.

The code generated for requirements is composed together with the unmodified Modelica model and the external variables in the CRML are bound to the Modelica variables. Usually this results in a balanced Modelica model but is certain cases, usually with badly specified CRML it might be possible to build invalid models.

DOI
10.3384/ecp218971

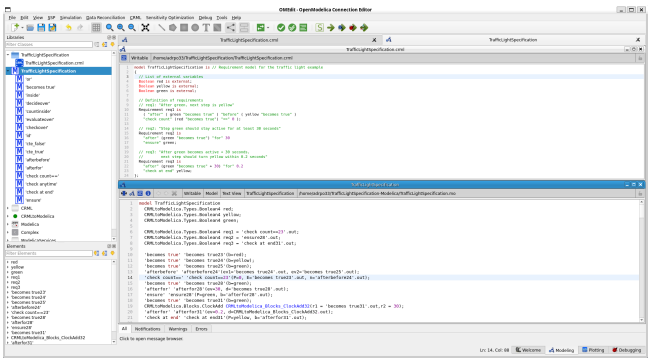


Figure 5. CRML file display in OMEdit and its Modelica translation.

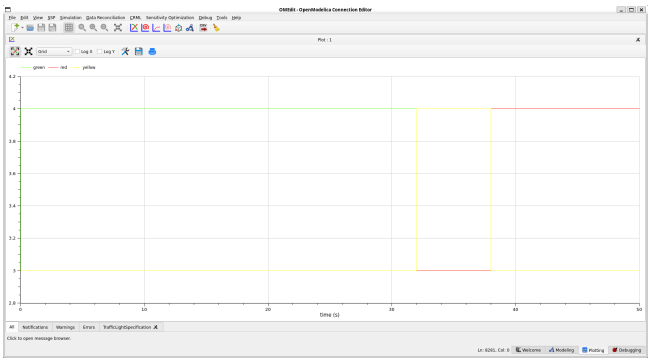


Figure 6. Simulation of the generated Modelica file corresponding to the original CRML file.

5 CRML Support in M&S tools

We have started integrating the CRML tools into OpenModelica. The CRML files can be loaded in the OpenModelica Connection Editor (OMEdit) and the text displayed with syntax highlighting, see Figure 5, top part. It is also possible to load entire directories containing CRML files.

The CRML files can be translated to Modelica from OMEdit via menu actions (via right click) and the translated .mo files are automatically loaded into OMEdit, see Figure 5, bottom part.

As expected, the generated Modelica files can also be simulated, as in Figure 6.

To tell OMEdit where it can find the tools and required Modelica libraries for the CRML translation a settings dialog is used, see Figure 7. If additional Modelica libraries are needed, one can specify them here.

The CRML compiler also specifies a test suite with about 150 test cases based on the ETL and FORML library blocks as well as more complex use-cases. This test suite can be run command line or from OMEdit and generates a test report that allows to evaluate coverage progress. Currently around 30% of the tests are successful as the test suite also includes features that are not yet supported in the compiler.

Using CRML files as text is only the first integration step in OpenModelica. Work is ongoing to support a

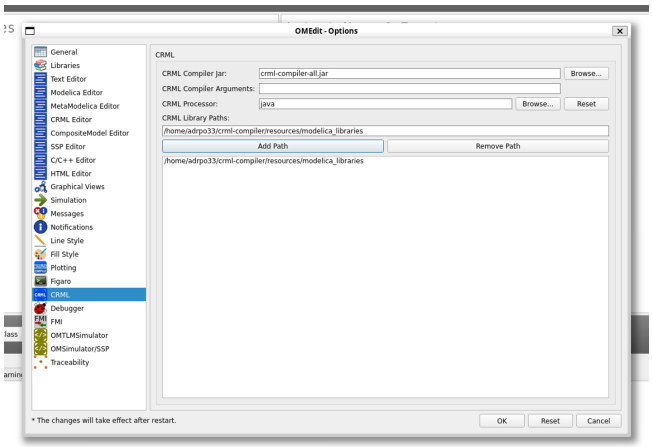


Figure 7. CRML tools settings in OMEdit.

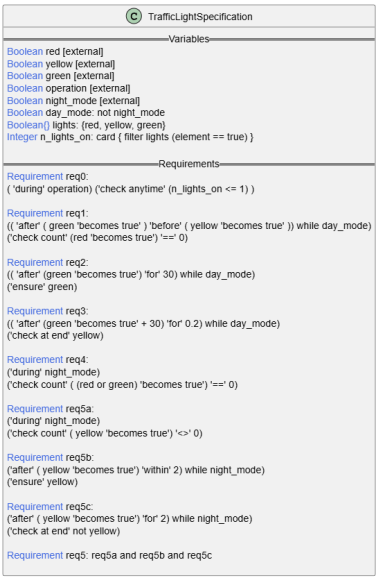


Figure 8. A graphical representation for CRML for the given example.

graphical representation language for CRML similar to UML as proposed by (Mazurié 2023), see Figure 8.

6 Conclusion and Future Work

CRML tooling is progressing to enable the formalization of realistic dynamic requirements of complex systems. Integration into OpenModelica enables a first step towards an easy access for the Modelica/FMI community to check requirements with simulation means. In order to be applicable to industrial-level use cases, future work plans to improve the toolset in several areas:

- Improvement of the user interface to visualise requirements simulation results as dashboards and structured html pages.
- Integration with the ReqIF standard (OMG 2025a) to provide a native gateway to the requirements management and traceability tools. This is in particular

needed to handle the metadata associated with the lifecycle of each requirement, coordinate the teams issuing requirements and deduce which set of requirements is valid and must be verified at a given point in time during the project.

- Alignment with the emerging SysML v2 standard (OMG 2025b) to structure and better visualise a large volume of requirements.

Other work is also planned, notably on the use of structural analysis techniques to statically verify consistency between CRML requirements and the introduction of the concept of Assume/Guarantee contract to facilitate the generation of test scenarios from CRML constraints.

7 Acknowledgements

The CRML initiative has initially been supported by the ITEA3 EMBrACE project, as well as National Research, Development and Innovation Fund of Hungary, financed under the [2019-2.1.1-EUREKA-2019-00001] funding scheme. It is continued and improved within the Open Source Modelica Consortium partly through EDF funding.

References

Alur, Rajeev (1999). "Timed automata". In: *International Conference on Computer Aided Verification*. Springer, pp. 8–22.

Baier, Christel and Joost-Pieter Katoen (2008). *Principles of model checking*. MIT press.

Bouskela, Daniel, Lena Buffoni, et al. (2023). "The Common Requirement Modeling Language". In: *Proceedings of the Modelica Conference 2023*.

Bouskela, Daniel and Audrey Jardin (2018). "ETL: A New temporal language for the verification of cyber-physical systems". In: *Proceedings of the Annual IEEE International Systems Conference 2018*. DOI: 10.1109/SYSCON.2018.8369502.

Bretz, Lukas, Lydia Kaiser, and Roman Dumitrescu (2019). "An analysis of barriers for the introduction of Systems Engineering". In: *Proceedings of the 29th CIRP Design*. Elsevier, pp. 783–789.

Buffoni, Lena et al. (2023). *Tutorial: CRML A Language for Verifying Realistic Dynamic Requirements*. Tech. rep. MODPROD. URL: https://github.com/OpenModelica/CRML/tree/main/resources/crml_tutorial.

CRML v1.2 (2023). *Specification v1.2 of the Common Requirement Modeling Language*. Tech. rep. ITEA EMBrACE Project. URL: https://github.com/OpenModelica/CRML/blob/main/language_specification/CRML%20specification_v1.2.pdf.

Henzinger, Thomas A (2000). "The theory of hybrid automata". In: *Verification of digital and hybrid systems*. Springer, pp. 265–292.

INCOSE Fellows (2019). *Briefing to INCOSE Board of Directors*. Tech. rep. INCOSE.

Kiniry, Joseph et al. (2022). *High Assurance Rigorous Digital Engineering for Nuclear Safety (HARDENS) Final Report*. Tech. rep. Galois.

Mazurié, Baptiste (2023). "Start of a new design method for a competitive Small Modular Reactor (SMR) adaptable to future uses". MA thesis. KTH, School of Engineering Sciences (SCI).

OMG (2017). *Unified Modeling Language (UML) version 2.5.1*. URL: https://www.omg.org/spec/UML/2.5.1/PDF (visited on 2025-07-29).

OMG (2025a). *Requirements Interchange Format (ReqIF)*. URL: https://www.omg.org/reqif/ (visited on 2025-07-31).

OMG (2025b). *Systems Modeling Language (SysML) version 2*. URL: https://github.com/Systems-Modeling/SysML-v2-Release/blob/master/doc/2a-OMG_Systems_Modeling_Language.pdf (visited on 2025-07-29).

SEBoK v2.11 (2024). *Guide to the Systems Engineering Body of Knowledge*. https://sebokwiki.org/wiki/.

Tinnerholm, John, Adrian Pop, and Martin Sjölund (2022). "A Modular, Extensible, and Modelica-Standard-Compliant OpenModelica Compiler Framework in Julia Supporting Structural Variability". In: *Electronics* 11.11, 1772. DOI: 10.3390/electronics11111772.

Appendix: CRML Cheat Sheet

Key Concepts

Requirements

A requirement is a `Boolean4` combining up to 4 items:

```
R = (Where) (When) (What) [How well]
```

Multiple requirements can be combined according to the algebra defined on the `Boolean4` type.

b	true	false	undecided	undefined
not b	false	true	undecided	undefined

b1 and b2	**true**	**false**	**undecided**	**undefined**
true	true	false	undecided	true
false	false	false	false	false
undecided	undecided	false	undecided	undecided
undefined	true	false	undecided	undefined

Syntax

Notation

```
[ expr ] ..... optional expression
{ expr } ..... expression repeated
expr_1 | ... | expr_n .... n expressions
'char' ..... character
"string" ..... string
```

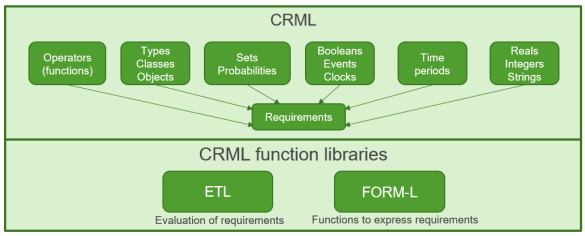
Expressions

```
[ [type] ident is] [ value |
external ] [ ; | , ] ..... expression
// This is single-line ..... comment
/* This is a multi-line */ .. comment
```

Keywords

Types: Boolean, Category, class, Clock, Event, Integer, library, model, Operator, package, Period, Periods, Probability, Real, String, Template, type
Special values: false, true, undecided, undefined, time
Special characters: (,),[,], {, }, ,, ;, ., ", ', E, e, //, /*, */, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Operators: =, +, -, *, /, <, <=, >, >=, ==, <>, ^, acos, alias, and, asin, associate, at, card, constant, cos, duration, element, else, end, estimator, exp, extends, external, filter, flatten, forbid, if, integrate, is, log, log10, mod, new, not, on, or, parameter, partial, proj, redeclare, sin, start, then, tick, time from, union, variance, while, with

Architecture



Real Operators

`Real x is decimal_value ....` constructor
`Real x is new Real n ...` constructor from *Integer n*
`x1 + | - x2 .....` binary addition | subtraction
`+ | -x1 ..........` unary addition | subtraction
`x1 * x2 .....` multiplication
`x1 / x2 .....` division
`x1^x2 .....` exponentiation

`if b then x1 else x2 ..........` if-clause
`x1 at c .....` value at *Clock c*
`duration b on P .....` duration during which *Boolean b* is true over a *Period P*
`e2 - e1 ...` elapsed duration between two *Event*
`time frome e1 ...` elapsed time from an *Event*

Integer Operators

`Integer n is integer_value` constructor
`Integer n is new Integer x` constructor from *Real x*
`n1 + | - n2 .....` binary addition | subtraction
`+ | - n1 .....` unary addition | subtraction
`n1 * n2 .....` multiplication
`n1 / n2 .....` division
`n1^n2 .....` exponentiation

`if b then n1 else n2 ..........` if-clause
`n1 at c .....` value of *Integer n1* at *Clock c*
`card c .....` number of ticks of *Clock c*
`card S .....` number of elements of *set S*

String Operators

`String s is string_value ...` constructor
`String s is new String x ...` constructor from *Real* | *Integer* | *Boolean* x
`s1 + s2 .....` concatenation

Boolean Operators

`Boolean b is true | false | undecided | undefined .....` constructor
`Boolean b is new Boolean c` constructor from *Clock c*
`b1 and b2 .....` conjunction
`b1 or b2 .....` disjunction
`not b .....` negation
`b1 * b2 .....` filter
`b1 + b2 .....` accumulation
`integrate b1 on P` integration over a *Period*
`b1 == b2 .....` equality
`if b then b1 else b2 ..........` if-clause
`b1 at c .....` value at *Clock c*
`x1 > x2` strictly greater than for *Real* | *Integer xi*
`x1 < x2 ...` strictly less than for *Real* | *Integer xi*
`n1 >= | <= n2 .....` comparison of *Integer ni*
`n1 <> n2 .....` different from for *Integer ni*
`x1 == x2 .....` equality for *Integer* | *Type xi*
`e1 < | <= e2 ....` (strictly) before for *Event ei*
`e1 > | >= e2 .....` (strictly) after for *Event ei*

Event Operators

`Event e is new Event b .` constructor from first occurence of *Boolean b*
`e1 proj c .....` projection on ticks of *Clock c*
`e1 proj(d) c ..` projection for duration *Real d*
`e1 + d .....` delay of *Real d*
`tick c .....` current tick of *Clock c*
`p start .....` opening event of *Period p*
`p end .....` closing event of *Period p*

Clock Operators

`Clock c is new Clock b .` constructor from *Boolean b*
`c1 proj c2 ....` projection on ticks of *Clock c2*
`c1 proj(d) c2 .` projection for duration *Real d*
`c1 + d .....` delay of *Real d*
`c1 filter cond tick .....` filter

`c1 and c2` ..... conjunction
`c1 or c2` ..... disjunction

Period and Periods Operators

`Period P is [|] e1, e2 [|]` constructor from *Event*
`Periods P is {P1, P2, ..., Pn}` . constructor from a set of *Period*
`Periods P is [ | ] c1, c2 [ | ]` constructor from *Clock*
`P2 while P1` ..... filter

Probability Operators

`Probability px is new Probability b` ..... constructor from *Boolean b*
`Probability px is new Probability b at c` ..... constructor at ticks of *Clock c*
`estimator px` ..... estimator
`estimator variance px` . variance estimator

Sets Operators

`[T{} S is]{u1,...,un}` .. non-empty typed set
`[T{} S is]{}` ..... empty typed set of type T
`T{} S is {expr}` special set with elements of different types (=*class, model, library, or package*)
`S1 union S2` ..... union of sets
`flatten S1` ..... flattening of a set
`filter(S1 cond (element))` ..... filter

Customisation

Type Constructors

`type T2 is (T1 x is value(a1,..., an)){Type D1, ...Dn; D1 a1 [is value];...}` ..... custom type
`partial type T is value` .... partial type
`T1 alias s` .... renaming type T1 with *String s*
`T1 forbid f1,...,fn` forbidding operators *fi*

Operator Constructors

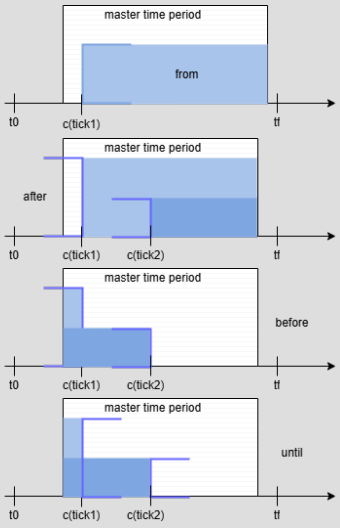
`Operator [T] 'w1' T1 u1 ... 'wn' Tn un = expr` .. custom operator 'w1w2wn' on inputs ui of type Ti
`Template w1 u1 ... wn un = expr` .... custom operator 'w1w2wn' on *Boolean*

Libraries

ETL Operators for Evaluating Booleans

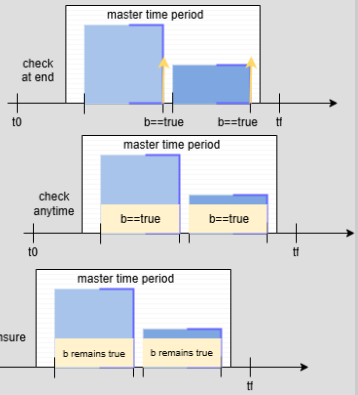
`c 'inside' p` .....
`b 'becomes true'` .....
`b 'becomes false'` .....
`b 'becomes true inside' p` .....
`b 'becomes false inside' p` .....
`'count' c 'inside' p` .....
`'decide' b 'over' p` . decision event, could be violation of *Boolean b* or end of *Period p*
`'evaluate' b 'over' p` accumulated value over a *Period p*
`'check' b 'over' P` accumulated value over a set of *Periods*

FORML Operators for Expressing Periods



`'from' c` .....
`'after' c` ....
`'before' c` ..
`'until' c` ....
`'during' b` .....
`'when' c` .....

FORML Operators for Expressing Conditions



`'check at end'`
`'check anytime'`
`P 'ensure' b`
`P 'check count' c '<=' n` .....

DOI 10.3384/ecp218971

A Thermal Digital Twin of Asphalt Pavements: Implementation and Application to an Instrumented Pavement in Costa Rica

Gustavo Canon Falla, [1] Volodymyr Prokopets, [2] Fabian Elizondo Arrieta[3] Eliécer Arias[3] Alexander Zeißler[1]

[1]Institute of Pavement Engineering, Technische Universität Dresden, Germany, {gustavo_adolfo.canon_falla, alexander.zeissler}@tu-dresden.de

[2]Chair of Software Technology, Technische Universität Dresden, Germany, volodymyr.prokopets@tu-dresden.de

[3]National Laboratory of Materials and Structural Models, University of Costa Rica, Costa Rica, {fabian.elizondo,eliecer.arias}@ucr.ac.cr

Abstract

This article presents the development of a synthetic thermal digital twin designed to reproduce the temperature distribution within the asphalt layers of a pavement system. At the core of the digital twin is a numerical model that solves Fourier's heat conduction equation using a finite difference method, enabling the simulation of transient heat transfer across multiple material layers.

The digital twin architecture is organized into three modular layers: the physical twin, which includes real-time temperature measurements from embedded sensors (synthetically simulated in this study); the twinning layer, which synchronizes physical and virtual data through feedback control using proportional-integral (PI) controllers; and the virtual twin, which integrates material properties, heat fluxes, and environmental conditions to replicate pavement thermal behavior.

The numerical model was validated using climate data and surface temperature measurements collected over a one-year period from a reference instrumented pavement in Costa Rica. In addition to radiative, convective, and conductive heat fluxes, the model accounts for the cooling effects of rainfall and subsequent evaporation—factors that are particularly relevant in tropical climates. The results demonstrate the capability of the digital twin to capture both long-term thermal trends and short-term responses to environmental events.

Keywords: Digital Twin, Asphalt Concrete, Pavement, Heat transfer

1 Introduction

1.1 Background

Digital transformation has changed the way infrastructures are designed, built and maintained (Barricelli, Casiraghi, et al. 2021). This transformation has been made possible by the integration of cyber-physical systems, which have paved the way for the development and use of digital twins (DTs). DTs are virtual replicas of physical assets that facilitate real-time data monitoring and analysis, thus enabling performance optimization and early detection of potential problems before they become critical issues. This transformative process has been enhanced by the rapid advancement of cutting-edge technologies, including advanced sensors, Internet of Things devices, artificial intelligence and big data analytics. These technologies empower DTs to provide highly accurate and predictive information on critical infrastructure performance and maintenance requirements.

The digitization of physical transportation infrastructures is currently in its early stages compared to other civil engineering infrastructures (Torzoni et al. 2023), facing several challenges that hinder its progress. One of the main obstacles encountered in the digitization process is the scarcity of comprehensive sensor data available for existing road network. This lack of data poses a major barrier to the creation of accurate DTs that can faithfully reproduce real road conditions. In addition, the absence of accurate numerical models further compounds the problem, as it can lead to erroneous predictions and impede the overall effectiveness of digitization efforts.

However, despite these challenges, digitization of transportation infrastructure holds immense potential to revolutionize the entire lifecycle of road systems, from design and construction to ongoing maintenance. The implementation of digital twins and the use of real-time data can enable proactive maintenance strategies, allowing early detection and resolution of potential problems before they escalate. This can improve safety, reduce downtime and increase operational efficiency. In addition, the digital transformation of roads facilitates optimized resource allocation through data-driven decision making, resulting in cost savings and improved sustainability.

1.2 Digital twin concept for road infractructures

The concept of a DT for road infrastructure is centered around the physical asset, which in this case is the pavement structure. Pavement structures can be classified into

two main categories: flexible pavements, which are constructed using asphalt concrete (AC) materials, and rigid pavements, which are made with cement-based materials. The present work, however, focuses solely on flexible pavements.

The structural integrity of flexible pavements is significantly influenced by the dynamic stresses and strains induced by vehicular traffic and environmental loads, such as temperature fluctuations, moisture, and precipitation. Furthermore, the materials utilized in flexible pavement construction undergo aging processes that can lead to a modification of their mechanical properties over time. Hence, the physical characteristics of AC pavements are time-dependent and can experience substantial variations in response to changes in external factors. Therefore, to ensure the integration of the pavement structure into a DT, it is crucial to incorporate embedded sensors capable of continuously monitoring and detecting changes in critical parameters. These parameters may include vibrations, strains, humidity, temperature, and other relevant variables that affect the structural performance of the pavement.

The data stream derived from the sensors provides a fundamental foundation for creating a physics-based computational model, commonly referred to as the digital twin model (DTM) or virtual twin. The DTM closely replicates the real-time state of the physical structure. The concept of DT involves creating a dynamic feedback loop between the physical twin and its corresponding virtual twin.

In order to facilitate the continuous feedback process between the physical and virtual systems in a DT, it is necessary to establish a physical-to-virtual connection. This connection utilizes the data collected by sensors and applies it to update the parameters of the DTM. This process enables the DT to remain in synchronization with the physical system, allowing for a deeper understanding of the system's behavior and the ability to predict and prevent potential issues.

Figure 1 illustrates the digital twin concept for asphalt pavements. The physical twin (i.e., physical asset) on the left represents the real-world pavement infrastructure equipped with sensing technologies, while the virtual twin on the right corresponds to the computational model, developed using data-driven, numerical, or hybrid modeling approaches. The cyclic interaction—referred to as twinning—establishes the physical-to-virtual connection, enabling the virtual model to continuously reflect changes in the physical system.

1.3 Objective and Methodology

The objective of this paper is to demonstrate the feasibility of implementing a thermal digital twin for asphalt pavements using the Modelica language. The focus lies on the development of a numerical model capable of accurately reproducing the transient thermal behavior of a multi-layered pavement structure. This model is a key component of a broader modular digital twin framework

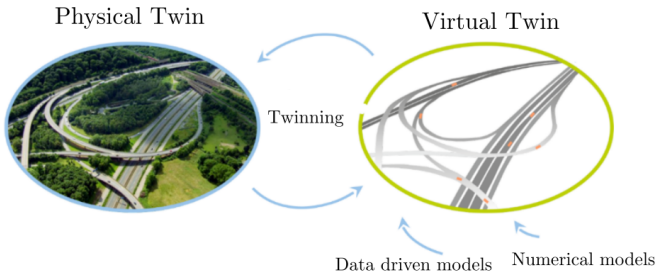


Figure 1. Digital twin concept of an asphalt pavement.

designed to integrate both thermal and mechanical aspects of pavement systems. In the present study, only the thermal subsystem is considered, which has been fully implemented in OpenModelica. The mechanical component—developed separately in Fortran and coupled to the thermal model via a Python wrapper—is acknowledged but remains outside the scope of this paper.

The thermal model is based on the one-dimensional form of Fourier's heat equation and is solved using a finite difference scheme with small time steps to capture transient heat flow through the pavement layers. The model receives surface and environmental heat fluxes as input and outputs the internal temperature distribution.

A proportional-integral (PI) controller forms the core of the twinning mechanism. It dynamically adjusts the surface albedo to minimize the error between the measured and simulated surface temperatures. This feedback process enables real-time synchronization between the physical twin—comprising embedded surface sensors—and the virtual thermal model.

Model validation was performed using data from a reference pavement located in the central region of Costa Rica. Surface temperature measurements collected over a one-year period were compared against simulation outputs. The model was further applied to forecast temperature distributions and to assess the thermal response of the pavement under typical tropical conditions. Special attention was given to the cooling effects produced by rainfall events, which were modeled through the inclusion of advective and evaporative heat transfer mechanisms.

2 Thermal digital twin architecture

The proposed architecture consists of three interconnected layers: the *physical twin*, the *twinning layer*, and the *virtual twin*. This multi-layer structure enables real-time synchronization between physical measurements and numerical simulations. Figure 2 illustrates the high-level system layout, highlighting the interaction among the three layers and their respective components.

The Digital Twin is implemented herein as a purely synthetic system. Both the Physical and Virtual Twins are modeled within the same simulation environment. Since no real hardware is involved in this implementation, communication protocols and latency are not relevant to the

DOI
10.3384/ecp218981

current setup. However, in a future deployment involving real pavement sensors, the communication interface between the physical system and the Modelica-based Virtual Twin would become crucial. Although thermal systems generally evolve slowly, latency could still impact the effectiveness of control recommendations or predictive maintenance alerts derived from the twin.

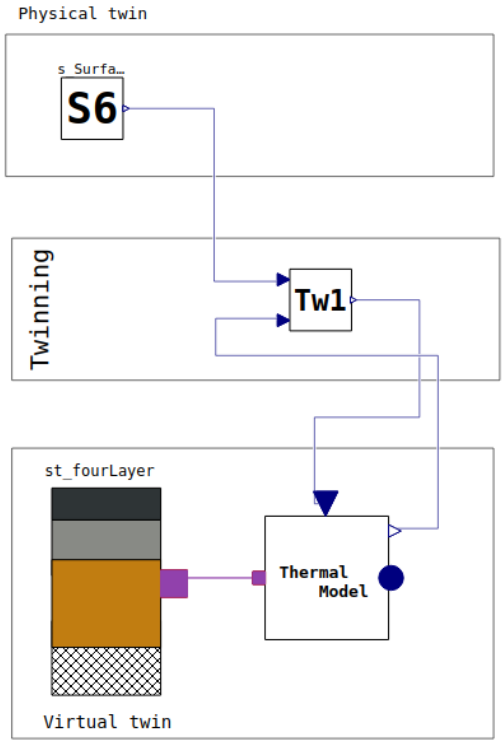


Figure 2. High-level architecture showing the physical twin, twinning layer, and virtual twin.

The model architecture has been developed in a modular fashion, enabling users to easily configure or extend the framework with their own use cases, as demonstrated in the package structure shown in Figure 3. It can be seen that the top-level package `DigitalTwinRoad` contains several subpackages, each dedicated to a specific aspect of the modeling framework, including sensors, numerical models, pavement structure, materials, traffic, and twinning logic.

This modular organization allows users to define and manage each component independently. For instance, the present example utilizes elements from the `Sensors` package to create six different environmental and pavement-related input components, including ambient temperature, solar radiation, vapor pressure, wind speed, cloud coverage, and surface temperature sensors. This flexibility supports a broad spectrum of applications, including academic experimentation, applied research, and real-world deployment scenarios.

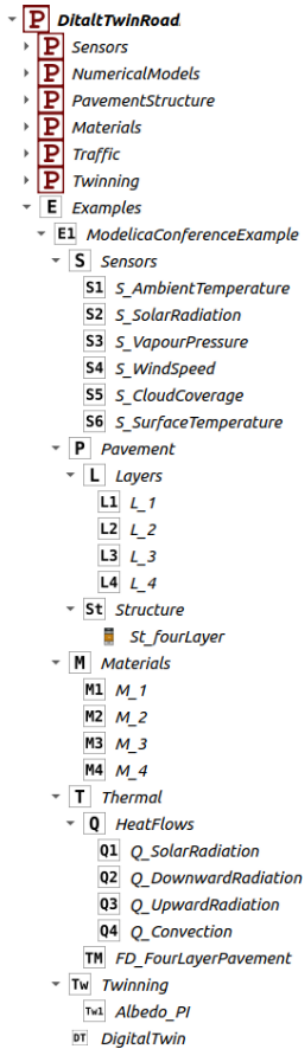


Figure 3. Modelica package structure of the digital twin road framework.

2.1 Physical Twin

The physical twin comprises sensors embedded in or positioned on the pavement structure. In this study, the primary physical input is provided by the `S6` component, which may be either a surface-mounted temperature sensor or a thermal camera. This device measures the real-time surface temperature of the pavement and transmits the data to the digital twin system for synchronization.

2.2 Twinning Layer

The twinning layer serves as the intermediary interface between the physical and virtual twins, enabling closed-loop communication and real-time synchronization. This layer is implemented using a Proportional-Integral (PI) controller. The PI controller receives two inputs: the measured surface temperature from the physical sensor (`S6`) and the corresponding temperature calculated by the virtual twin. It computes the error between these two values and generates an output signal (`outputSignal`) that is used to dynamically adjust the surface albedo, which di-

rectly influences the calculation of solar heat input to the pavement surface and controls the energy absorbed in the thermal model. This feedback mechanism minimizes discrepancies and ensures that the virtual twin consistently aligns with the behavior of the real pavement system.

2.3 Virtual Twin

The virtual twin is responsible for simulating the internal thermal behavior of the pavement system. It consists of two primary components: (1) the pavement structure model and (2) the numerical model for heat transfer.

2.3.1 Pavement Structure

The pavement structure is represented using a layered configuration defined in the `PavementStructure` and `Materials` packages. Each layer corresponds to a physical segment of the pavement—such as surface, binder, base, and sub-base—and is characterized by specific material properties including thermal conductivity, density, specific heat capacity, and thickness. These parameters define the thermal inertia and conduction capacity of each layer, forming the physical foundation for the thermal simulation.

2.3.2 Numerical Thermal Model

The second component of the virtual twin is the thermal model, which numerically solves the one-dimensional transient heat conduction equation using a finite difference method. The model is based on the Fourier differential heat equation without internal heat generation (Adam et al. 2023):

$$\frac{\partial^2 T}{\partial z^2} = \frac{\rho \cdot c}{k} \frac{\partial T}{\partial t} \tag{1}$$

where T is the temperature within the pavement (K), z is the pavement depth (m), t is time (s), k is the thermal conductivity ($\mathrm{W \cdot m^{-1} \cdot K^{-1}}$), ρ is the density ($\mathrm{kg \cdot m^{-3}}$), and c is the specific heat capacity ($\mathrm{J \cdot kg^{-1} \cdot K^{-1}}$).

Figure 4 shows the Modelica implementation of this thermal model. The core numerical solver is encapsulated in the `FD4` block, which receives as inputs four distinct surface heat fluxes:

- `Q1`: Solar radiation (Alavi, Pouranian, and Hajj 2014)
- `Q2`: Downward longwave radiation (Hall et al. 2012; Bliss 1961)
- `Q3`: Upward longwave radiation
- `Q4`: Convective heat transfer (Wolfe, Heath, and Colony 1983; Vehrencamp 1953; Bentz 2000)

Owing to the modular design of the model architecture, additional heat flux components—such as latent heat losses due to evaporation or heat exchange caused by rainwater infiltration—can be seamlessly integrated into the thermal model.

These heat fluxes are summed and applied at the surface boundary of the finite difference grid. The model outputs the surface temperature (`T_surface`), which is compared against physical sensor data to drive the feedback mechanism in the twinning layer.

Additionally, the model computes the coefficients of a fifth-degree polynomial used to approximate the in-depth temperature distribution within the pavement. This polynomial expression takes the form:

$$T(z) = a_0 + a_1 z + a_2 z^2 + a_3 z^3 + a_4 z^4 + a_5 z^5 \tag{2}$$

where z is the depth (m) and a_0 through a_6 are coefficients determined dynamically during simulation, provided as output via the `aCoeffs[5]` port.

The fifth-degree polynomial was chosen to ensure proper fitting of the in-depth temperature distribution within the pavement structure. This level of polynomial provides sufficient flexibility to capture the nonlinear temperature gradients that develop due to surface heating and thermal diffusion through layered materials.

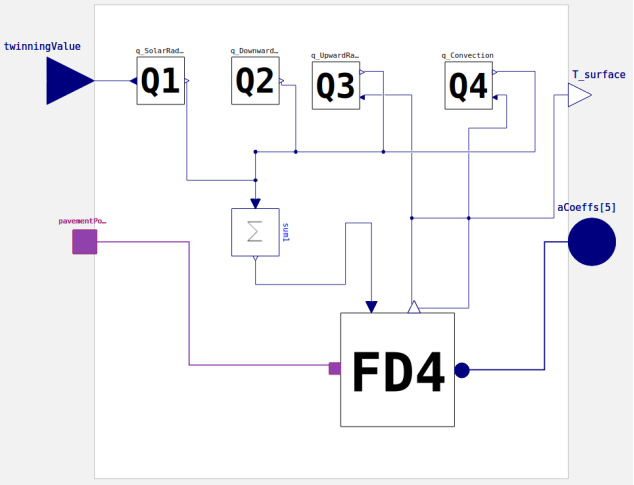


Figure 4. Modelica implementation of the thermal model using finite difference approximation in the `FD4` component.

3 Example Application: Instrumented pavement in Costa Rica

This case study presents the application of the proposed digital twin model to a real asphalt pavement structure located along Route 35 in the canton of San Carlos, Alajuela province, Costa Rica (Figure 5).

3.1 Pavement structures, climatic conditions and material properties

The pavement structure consists of four layers: the top layer is a polymer-modified asphalt concrete with a thickness of $d_1 = 60$ mm. The second layer is a conventional (unmodified) asphalt concrete with a thickness of $d_2 = 70$

DOI 10.3384/ecp218981

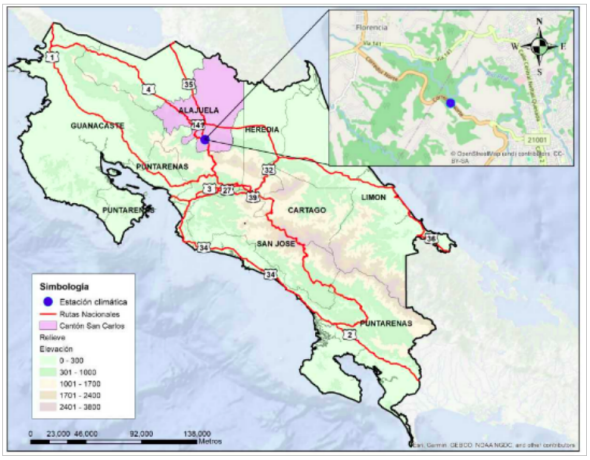


Figure 5. Location of the reference pavement along Route 35, San Carlos, Alajuela, Costa Rica.

mm. The third layer is a hydraulically stabilized granular base with a thickness of 240 mm, and the fourth is a granular sub-base with a thickness of 300 mm.

Meteorological data required for the numerical thermal model were obtained from a weather station installed on the roadside adjacent to the test section (Figure 6). The station is equipped with a data acquisition system capable of managing multiple sensors, configuring capture frequencies, and handling data storage and access.Additionally, the station supports cellular network communication, enabling remote configuration management and real-time data retrieval. This setup ensures that the digital twin model receives continuous and reliable boundary condition data for accurate thermal simulation.

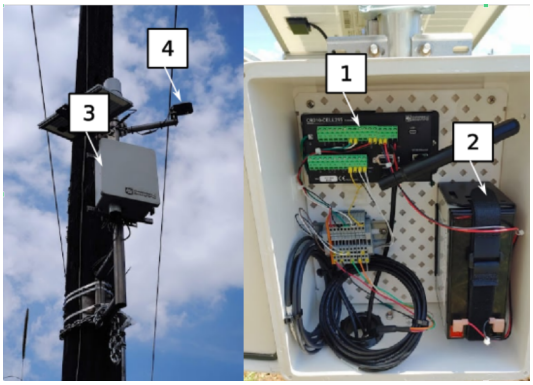


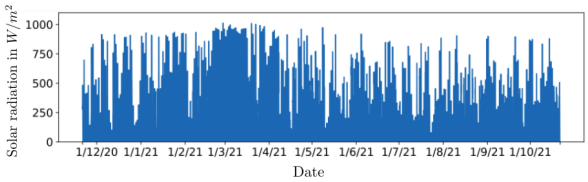
Figure 6. Meteorological station setup for data acquisition. (1) CR310 datalogger with 3G remote connectivity, (2) backup battery, (3) solar panel for system power, (4) CLIMA VUE 50 multiparameter sensor.

The collected data included hourly measurements of solar radiation, wind speed, air temperature, relative humidity, and precipitation. These values were linearly interpolated to generate input data at 10 minutes intervals.

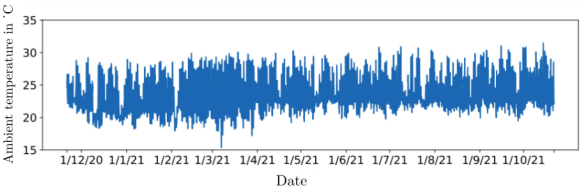
The heat transfer due to advection, which in this case refers to the thermal exchange caused by rainfall and its temperature difference with the pavement surface, was calculated according to the method described by Yavuzturk and Ksaibati (Yavuzturk and Ksaibati 2006).

Additionally, it was assumed that rainwater drains immediately from the pavement surface, forming a thin film from which evaporation occurs. The heat transfer due to evaporation was determined following the approach proposed by Van Buren (Van Buren et al. 2000).

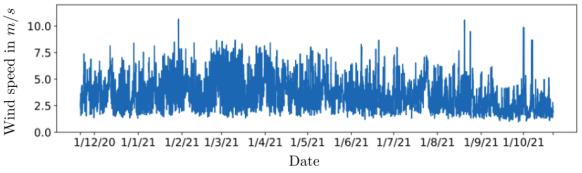
Climatic data from an 11-month period (from November 20, 2020, to October 20, 2021) are presented in Figure 7. As shown, air temperature varied approximately between 15°C and 32°C. Solar radiation reached a maximum of 1012.0 W·m^{-2}, while wind speed peaked at 10.7 m/s with an average of 3.5 m/s.



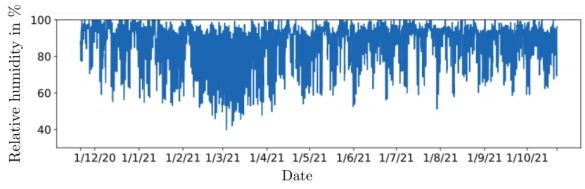
(a) Solar Radiation



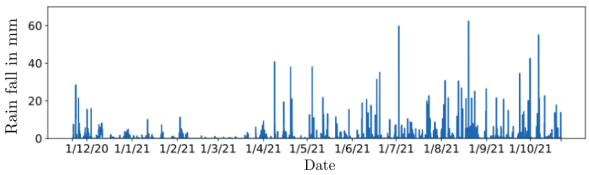
(b) Air Temperature



(c) Wind Speed



(d) Relative Humidity



(e) Precipitation

Figure 7. Hourly climatic variables measured at the roadside weather station over the 11-month period.

The thermal properties assigned to each pavement layer in the model include density, thermal conductivity, and specific heat capacity. For the asphalt layers (Layers 1 and 2), these properties were experimentally measured at

Technische Universität Dresden using an ISOMET 2114 device, applied to materials similar to those used in the Costa Rica test section. The granular layers (Layers 3 and 4) were characterized using values obtained from published literature.

Layer 1 has a density of 2530 kg/m^3, thermal conductivity of 1.46 W/m·K, and specific heat capacity of 654.13 J/kg·K. Layer 2 has a slightly higher density of 2560 kg/m^3, thermal conductivity of 1.33 W/m·K, and specific heat capacity of 648.61 J/kg·K. For the granular layers, Layer 3 has a density of 1800 kg/m^3, thermal conductivity of 1.0 W/m·K, and specific heat capacity of 900 J/kg·K, while Layer 4 has a density of 1700 kg/m^3, thermal conductivity of 0.8 W/m·K, and the same specific heat capacity of 900 J/kg·K.

3.2 Thermal Model Validation

To validate the thermal model, simulated temperature values were compared with field measurements obtained from a thermometer installed at a depth of 40 cm along the roadside. Figure 8 shows the results of the numerical simulation overlaid on the field-measured temperatures. As observed, the model is capable of replicating both the overall shape and magnitude of the thermal history with reasonable accuracy.

The average absolute difference between the simulated and measured temperatures was 1.08 °C. While this is a modest error, it should be considered in the context of the relatively narrow temperature range (∼20 °C to 28 °C). Furthermore, localized differences greater than 4 °C were observed during some transient periods, such as around January 9, 2021.

These discrepancies are attributed to two main factors: (1) the thermal properties used in the model were not site-specific, and (2) the field measurements were taken at a depth of 40 cm, beneath the surface and at the pavement edge. At this depth, thermal variations are attenuated and phase-shifted relative to the surface, making direct alignment with surface-driven simulations inherently challenging. Nevertheless, the model captures the overall thermal trends and magnitudes with sufficient fidelity for Digital Twin calibration purposes.

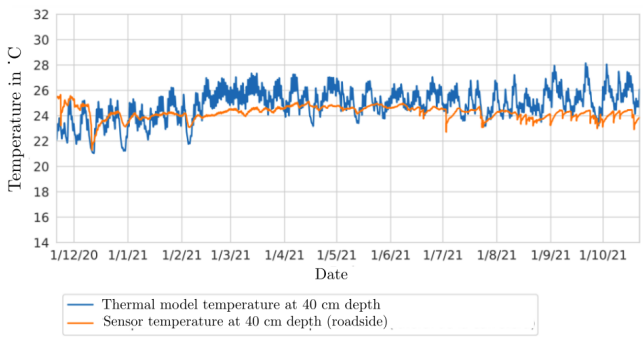


Figure 8. Comparison between measured and simulated temperatures at 40 cm depth over the validation period.

4 Numerical Results

4.1 Surface Temperature of the Asphalt

Figure 9 shows the simulated surface temperature of the asphalt pavement in comparison with the ambient air temperature over the 11-month period. As expected, surface temperature variations are significantly larger than those of the ambient air due to the combined effects of solar radiation, convective heat transfer, and thermal inertia of the pavement structure.

While ambient temperatures remained within a relatively narrow range (typically between 18° C and 32° C), surface temperatures frequently exceeded 40° C, with peaks reaching nearly 50° C under high solar radiation. This highlights the importance of modeling radiative and conductive mechanisms when predicting thermal loads on pavement surfaces, particularly in tropical environments.

Such elevated surface temperatures can increase the risk of temperature-related distresses such as rutting and fatigue cracking, even though low-temperature cracking is unlikely to occur in this climate.

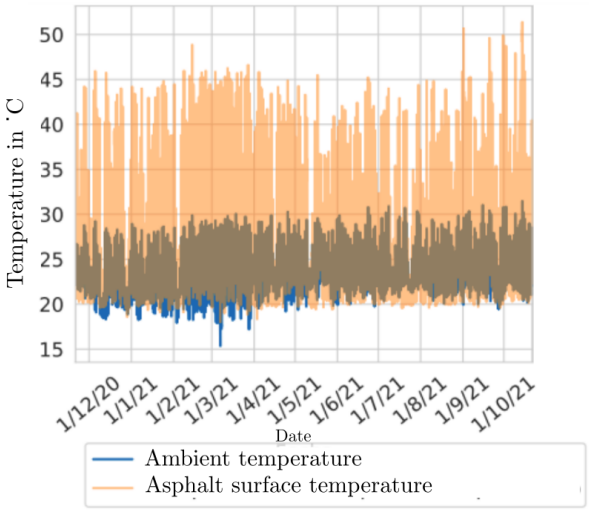


Figure 9. Simulated asphalt surface temperature compared to ambient air temperature from November 2020 to October 2021.

4.2 Cooling Effect of Rainfall on Pavement Surface

For a country like Costa Rica, incorporating the effects of rainfall into the thermal model is essential. Due to its tropical climate, Costa Rica experiences substantial precipitation throughout the year. In this context, the evaporation of rainwater serves as a key cooling mechanism that significantly influences pavement surface temperature.

Figure 10 shows the simulated surface temperature distribution over a two-day period, including a rainfall event during the night of January 2, 2021. The figure clearly illustrates that the onset of precipitation, followed by evaporation, produces a noticeable cooling effect on the asphalt surface.

DOI
10.3384/ecp218981

With the inclusion of rainfall effects in the model, surface temperatures are approximately 2–3°C lower than they would be without accounting for this phenomenon. This result highlights the importance of modeling transient environmental events to improve the realism and predictive capability of digital twin systems for pavements in tropical climates.

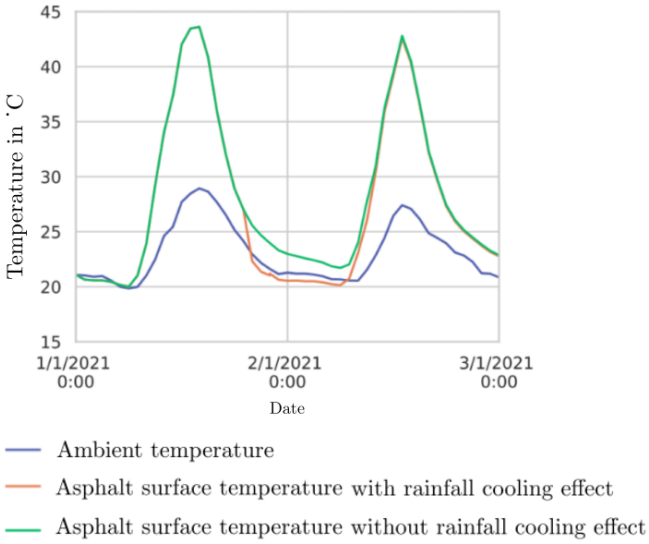


Figure 10. Simulated effect of rainfall on asphalt surface temperature during a two-day period including a nighttime rain event on 02.01.2021.

4.3 Temperature Distribution Within the Asphalt Layers

Figure 11 presents the simulated temperature distribution across the asphalt layers (Layers 1 and 2) over a 48-hour period, from January 1, 2021 at 00:00 to January 3, 2021 at 00:00. As expected, asphalt temperatures peak around midday, coinciding with the maximum solar radiation.

During the daytime, a clear thermal gradient can be observed between the top and bottom of the asphalt layers, with surface temperatures significantly higher due to direct solar exposure. Conversely, nighttime conditions lead to more uniform temperatures across the asphalt layers, typically close to the ambient air temperature of around 20° C.

This thermal behavior reflects the dominant heat transfer mechanisms, particularly conduction from the surface downward. A noticeable thermal lag is observed, with a delay of approximately 1.5 hours between the peak surface temperature and the temperature observed at the bottom of the asphalt layers. This lag is characteristic of heat diffusion in materials with thermal mass and indicates the importance of time-resolved simulation in capturing pavement response to environmental loads.

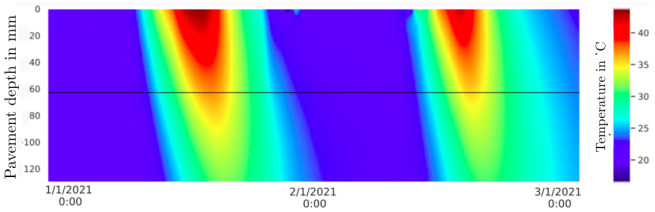


Figure 11. Simulated temperature distribution in asphalt layers (1 and 2) over a 48-hour period from 01.01.2021 to 03.01.2021.

5 Conclusions

This study demonstrates the development and application of a synthetic thermal digital twin for asphalt pavements using the Modelica language and the OpenModelica platform. The model captures transient heat conduction through a multilayer pavement structure by solving Fourier's heat equation with a finite difference scheme. Real-time synchronization between the physical and virtual twins is achieved through a PI controller that adjusts surface albedo to minimize temperature discrepancies.

The digital twin was applied to a case study in Costa Rica, where the model was validated using field temperature data collected over an 11-month period. The model showed good agreement with measured values, with an average absolute error of 1.08° C. The inclusion of rainfall and evaporation effects proved critical in accurately simulating surface cooling events, demonstrating the value of incorporating site-specific environmental conditions.

Results also highlighted the thermal behavior of asphalt layers under daily and seasonal cycles, revealing surface temperature peaks exceeding 50° C and internal thermal lags of about 1.5 hours. These conditions, typical in tropical climates, suggest that pavement distress mechanisms such as rutting and fatigue cracking may be more relevant than thermal cracking.

Finally, the modular architecture of the Modelica-based framework enables easy integration of additional environmental inputs and control strategies. This flexibility supports further development of comprehensive digital twins capable of coupling thermal and mechanical behavior for improved pavement performance prediction and infrastructure management.

In a real-world implementation, the Digital Twin would transmit predicted in-pavement temperature data from the virtual model to the physical road system. This information flow is essential for enabling proactive pavement management strategies. Forecasts of temperature at various depths can inform decisions such as optimal timing for de-icing, preventative maintenance to reduce thermal cracking, and adaptive load management during extreme heat or cold events. By delivering this predictive insight, the Virtual Twin supports data-driven interventions that can reduce operational costs, extend pavement lifespan, and enhance roadway safety. Although this study focuses on a synthetic case without direct communication to a

physical system, the proposed architecture is designed to support such bidirectional data exchange in future deployments.

Acknowledgements

The authors gratefully acknowledge the financial support provided by DFG SFB/TRR 339, Project ID 45359608.

References

Adam, Q. F. et al. (2023). "Modeling the use of an electrical heating system to actively protect asphalt pavements against low-temperature cracking". In: *Cold Regions Science and Technology* 205, p. 103681. DOI: 10.1016/j.coldregions.2022.103681. URL: https://doi.org/10.1016/j.coldregions.2022.103681.

Alavi, M. Z., M. R. Pouranian, and E. Y. Hajj (2014). "Prediction of asphalt pavement temperature profile with finite control volume method". In: *Transportation Research Record* 2456, pp. 96–106. DOI: 10.3141/2456-10. URL: https://doi.org/10.3141/2456-10.

Barricelli, M., E. Casiraghi, et al. (2021). "Digital twins in infrastructure: definitions, current practices, challenges, and opportunities". In: *International Journal of Construction Management* 21.9, pp. 823–838. DOI: 10.1080/15623599.2021.1966980.

Bentz, D. P. (2000). *A computer model to predict the surface temperature and time-of-wetness of concrete pavements and bridge decks*. Tech. rep. NISTIR 6551. National Institute of Standards and Technology. URL: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=860286.

Bliss, R. W. (1961). "Atmospheric radiation near the surface of the ground: A summary for engineers". In: *Solar Energy* 5, pp. 103–120. DOI: 10.1016/0038-092X(61)90002-9. URL: https://doi.org/10.1016/0038-092X(61)90002-9.

Hall, M. R. et al. (2012). "Influence of the thermophysical properties of pavement materials on the evolution of temperature depth profiles in different climatic regions". In: *Journal of Materials in Civil Engineering* 24.1, pp. 32–47. DOI: 10.1061/(ASCE)MT.1943-5533.0000357. URL: https://doi.org/10.1061/(ASCE)MT.1943-5533.0000357.

Torzoni, Matteo et al. (2023). "A digital twin framework for civil engineering structures". In: *arXiv preprint arXiv:2308.01445*. DOI: 10.48550/arXiv.2308.01445.

Van Buren, M. A. et al. (2000). "Thermal enhancement of stormwater runoff by paved surfaces". In: *Water Research* 34.4, pp. 1359–1371. DOI: 10.1016/S0043-1354(99)00250-2. URL: https://doi.org/10.1016/S0043-1354(99)00250-2.

Vehrencamp, J. E. (1953). "Experimental investigation of heat transfer at an air-earth interface". In: *Transactions of the American Geophysical Union* 34, pp. 22–30. DOI: 10.1029/TR034i001p00022. URL: https://doi.org/10.1029/TR034i001p00022.

Wolfe, R. K., G. L. Heath, and D. C. Colony (1983). "Cooling curve prediction of asphaltic concrete". In: *Journal of Transportation and Pipeline Engineering* 109, pp. 137–147. DOI: 10.1061/(ASCE)0733-947X(1983)109:1(137). URL: https://ascelibrary.org/doi/10.1061/(ASCE)0733-947X(1983)109:1(137).

Yavuzturk, C. and K. Ksaibati (2006). *Assessment of Thermal Stresses in Asphalt Pavements Due to Environmental Conditions*. Tech. rep. MPC-06-181. University of Wyoming. URL: https://www.ugpti.org/resources/reports/downloads/mpc06-181.pdf.

Status of the TransiEnt Library: Transient Simulation of Complex Integrated Energy Systems

Markus Gillner[1] Jan Westphal[1] Béla Wiegel[2] Tom Steffen[2] Julian Urbansky[3] Anne Hagemeier[3] Stefanie Ruppert[4] Annika Heyer[4] Jörn Benthin[4] Tim Hanke[5] Johannes Brunnemann[5] Christian Becker[2] Arne Speerforck[1]

[1]Institute of Engineering Thermodynamics, Hamburg University of Technology, Germany, {markus.gillner, j.westphal, arne.speerforck}@tuhh.de
[2]Institute of Electrical Power and Energy Technology, Hamburg University of Technology, Germany,{bela.wiegel, tom.steffen, c.becker}@tuhh.de
[3]Fraunhofer UMSICHT, Germany,{julian.urbansky, anne.hagemeier}@umsicht.fraunhofer.de
[4]Gas- und Wärme-Institut Essen e.V., Germany,{stefanie.ruppert, annika.heyer, joern.benthin}@gwi-essen.de
[5]XRG Simulation GmbH, Germany,{hanke, brunnemann}@xrg-simulation.de

Abstract

The TransiEnt Library is an open-source Modelica Library originally developed at the Hamburg University of Technology. It is a flexible framework for modelling and analysing the dynamic behaviour of coupled energy systems under current and future scenarios. With the addition of three new members to the TransiEnt Library consortium, namely Fraunhofer UMSICHT, Gas- und Wärme-Institut Essen e.V. and XRG Simulation GmbH, the TransiEnt Library has expanded its portfolio of models and methods for investigating the challenges in energy systems. Building upon the previous status report, this article presents the latest developments in the TransiEnt library, highlithing its extended capabilities to model and simulate large, complex energy systems. The recent developments include the automatic generation of aggregated models at district and regionals levels, as well as the modelling of medium- and low-voltage electrical distribution networks. In addition, new concepts for the representation of large-scale heating networks have been developed and are presented alongside illustrative use cases.

Keywords:District Heating Network, Large-scale Simulation, Modelica, Sector coupling

1 Introduction

Due to climate change, energy systems are in the midst of a transformation from using fossil fuels to a sustainable energy production. Large, centralized plants are being replaced by many small, decentralized ones. As energy production becomes predominantly renewable, various energy systems formerly based on heat or gas are increasingly electrified, resulting in enhanced interconnectivity and rising electricity demand through power to x technologies. In addition to the growing demand for electricity, the decentralized and small-scale systems in the electricity grid are leading to bi-directional power flows, which are putting a particular strain on the distribution grids (Greve et al. 2021). The intermittent nature of renewable energy generation from wind and solar on hourly and seasonal time scales leads to a growing need for short-term and long-term storage with different capacity and performance requirements. The digitalization of energy systems also makes it possible to control generation and consumption in real time.

This transformation process is leading to significantly more complex and interconnected energy systems that need to be assessed in terms of economic efficiency, sustainability, reliability and resilience. The economic design of storage systems, the investigation of flexibility through increased interconnection, and the resilience of grids require the investigation of dynamic processes.

The modelling language Modelica is ideally suited for simulating the dynamic behaviour of multi-domain systems. There are several open-source Modelica libraries that specialize in different areas of energy system modelling (Modelica Association 2025). The AixLib (Maier et al. 2024) and the buildings library (Wetter et al. 2014), for example, are two very comprehensive libraries that focus on the modeling of building energy systems at different levels of detail, from individual buildings to district level. The DisHeatLib library (Leitner et al. 2019) is a useful tool for simulating heating networks. However, to the authors' knowledge, no large-scale heating networks have been simulated with it yet. The library also does not take coupled networks into account. The ClaRa library (Vojacek et al. 2023) enables the analysis of power plant processes. Nevertheless, owing to its level of detail, it is not appropriate for simulating large networks. However, the TransiEnt library is designed for analyzing large-scale, coupled energy systems. Building upon the previous status paper (Senkel, Bode, Heckel, et al. 2021), this paper

focuses on simulations of large-scale energy networks. In the following, the context of the library, its scope and structure are described, before three more detailed considerations of larger energy systems are given, that are enabled by the new models in the library. At first, the modeling of large energy systems at regional level using the superstructure model is explained. Then, the modeling of large medium and low voltage distribution grids is presented. Finally, the modeling of large district heating networks is discussed. The paper concludes with a summary and an outlook on future developments.

2 Overview of Library

2.1 Context of TransiEnt Library

Experience from the development of the libraries *HKSim* (Wischhusen, Lüdemann, and Schmitz 2003) and *ACLib* (Pfafferott and Schmitz 2003) as well as a research focus on climate-friendly energy and environmental technology at the Hamburg University of Technology (TUHH) led to the idea of a Modelica library for the simulation of integrated energy systems, which resulted in the research project TransiEnt.EE from 2013 to 2017. First results and the relevance of dynamic consideration of integrated energy systems in the calculation of CO_2 emissions for different conversion paths were presented at the Modelica conference in 2015 (Andresen, Dubucq, et al. 2015) and led to the first release (version 1.0.0) of the Transient Library in 2017. The main focus was on the analysis of costs and CO_2 emissions of energy systems, facilitating the evaluation of efficiency measures.

The library was further developed in the follow-up project ResiliEntEE from 2017 to 2021. The focus here was on applying the Transient Library for large-scale simulations through numerically efficient modelling and evaluating the resilience of energy systems. As a new partner, the Institute of Electrical Power and Energy Technology at the TUHH made a significant contribution to the development of electrical power systems and grids (Heckel and Becker 2019). The resulting changes to the library were presented at the Modelica conference in 2021 (Senkel, Bode, Heckel, et al. 2021).

At the same time, the TransiEnt Library was also used and further developed outside the TUHH, in particular in the IntegraNet project (2016 - 2019). The project partners Fraunhofer UMSICHT, Gas- und Wärme-Institut Essen e.V. investigated sector coupling at the distribution system level, supported by XRG Simulation GmbH. Consequently, new models for gas and heating networks were developed, as well as the GridConstructor for complex district modelling (Benthin, Heyer, et al. 2019).

The follow-up project IntegraNet II (2019-2023) has led to new developments, including the introduction of an electric vehicle model to represent electromobility in local energy systems. Moreover, the coupling of the electricity and transport sectors has been facilitated, and heating network models have been revised to reduce computation time.

Due to many years of cooperation and parallel development of the TransiEnt library, a consortium was formed with TUHH and the Integranet project partners in April 2021 to combine their activities and support further developments. Together they released version 2.0.0 with extensions in the domains of large-scale simulations and resilience at the transport system level, in addition to novel models for the underlying distribution system level. After the alliance was founded, the development of the library was moved to GitHub in favor of better access and community collaboration (TransiEnt-official/transient-lib).

Components of the TransiEnt library have also been used in the project CyEntEE (2021 – 2024) to develop novel dynamic integrated low and medium voltage grid models with the goal to optimize flexibility and develop congestion management strategies. Additionally, the ongoing EffiziEntEE project (2022 - 2025) further utilizes and enhanced those models. Furthermore in EffiziEntEE heat models, which enable the numerically efficient simulation of large-scale district heating networks, were developed. The models from CyEntEE and EffiziEntEE are part of the release for this publication. The new release also contains the latest improvements in the automation of model creation at district and regional level. The next chapter provides a detailed insight into the improvements and innovations of the TransiEnt library.

2.2 Scope of TransiEnt Library

The TransiEnt Library serves as a comprehensive Modelica library specifically designed for the dynamic modeling of integrated energy systems with a high share of renewable energy (RE). It facilitates the representation of interactions among the domains of electricity, heat, and gas, thereby addressing the complexities inherent in present and future energy systems.

The primary objective of the TransiEnt Library is to provide a versatile framework that can be applied across various areas of energy research and analysis. Through documentation and the provision of check models, the library aims to ensure accessibility for researchers and engineers engaged in the dynamic modeling of integrated energy systems.

The applications of the TransiEnt library extend from district-level analysis to regional investigations, comprising industrial, commercial and residential sectors. This integration facilitates the interactions between producers (power plants and decentralized units), consumers, storage systems, and the infrastructures of the gas, heat, and electricity sectors.

The existing models of the TransiEnt library allow the modeling and simulation of complex integrated energy systems and can therefore be used to study the effects and opportunities of increased integration of electricity, heat and gas from RE into existing structures, such as the deployment planning of combined heat and power plants (cf. Peniche and Kather 2015) or the use of conven-

DOI 10.3384/ecp218989

tional power plants per se (cf. Dubucq and Ackermann 2015). The library also ensures the investigation of the potential and integration of green hydrogen in industry (cf. Andresen and Schmitz 2016; Andresen, Bode, and Schmitz 2018) or the volatile nature of RE on the stability of the electricity grid and possible flexibility options (cf. Dubucq and Ackermann 2017; Heckel, Möws, and Becker 2022). By providing models for the interconnection of the sectors, i.e. power-to-heat (PtH) and power-to-gas (PtG) technologies, issues related to flexibility between the individual sectors, such as the use of surplus electricity in PtH units (Bode and Schmitz 2019) or voltage stability in coupled grids (cf. Heckel and Becker 2020; Heckel, Steffen, and Becker 2022), can be investigated. In particular, the introduction of a resilience index in coupled grids (Senkel, Bode, and Schmitz 2021; Heckel and Becker 2019) can be used to assess the resilience of energy grids to disturbances (cf. Bode, Senkel, and Schmitz 2020; Heckel, Senkel, et al. 2022). Furthermore, uncertainties regarding the economic evaluation of future investments in building energy systems can be addressed through real options analysis using simulated energy flows (Glombik and Fromme 2024).
Moreover, the TransiEnt Library offers aggregation models, such as `GridConstructor` and `Superstructure`. These models empower users to model entire districts or large energy systems at the transport grid level with simple parameterizations. For instance, the `GridConstructor` can be used to analyze the physical impact of vehicle-to-grid communication attacks on low-voltage grids (Zhdanova et al. 2022).

While the TransiEnt Library offers robust modeling capabilities and ensures large-scale simulations, it is important to note its inherent limitations. The optimization of energy systems cannot be performed with the models in Dymola itself, but must be implemented via interfaces such as Matlab or Python. Furthermore, the library is not designed as a high-level designing tool; rather, it allows dynamic simulations of given system designs to analyze their impacts on the energy system.

2.3 Structure of TransiEnt Library

The TransiEnt Library provides models for components of gas, heat and electrical systems (generators, storage facilities, infrastructure, consumers) as well as various system models. Some basic models, particularly in the area of heat and gas, as well as basic modeling principles have been adopted from the ClaRa Library. This freely available model library was developed by TLK-Thermo GmbH and XRG Simulation GmbH and contains models of power plant components for the simulation of power plant systems.
As shown in Figure 1, the current version of the TransiEnt library consists of eight top level packages. Since the last status paper, the package `SystemGeneration` has been added, which supports the automated generation of energy systems on a city and regional level: `GridConstructor` and `Superstructure`. Please refer to Benthin, Heyer, et al. (2019) for information on how the `GridConstructor` works. The modeling of regional energy systems with the `Superstructure` is presented in subsection 3.1.

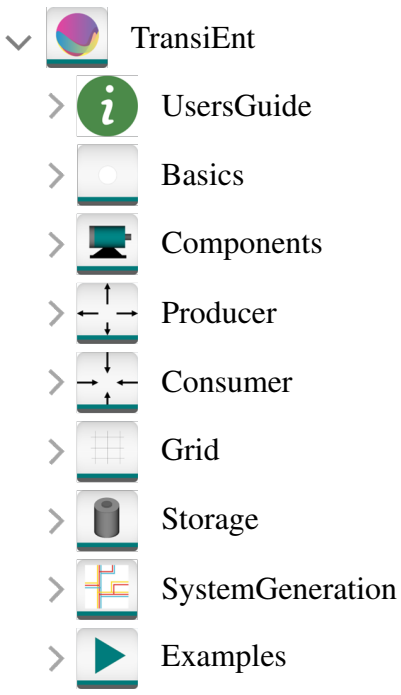


Figure 1. Icons of the relevant top level packages inside the TransiEnt Library

3 Exemplary Developments and application examples

The focus of this chapter is to present the most important developments in the TransiEnt library. First, the automated model creation through the `Superstructure` model is presented. Second, the developments in modeling medium and low voltage networks are explained. Finally, two modeling approaches for large heating networks are introduced.

3.1 Cellular Energy Systems with the Superstructure

Modeling of large energy systems can include large structural data sets and may require a substantial amount of manual work and is error prone. Script based automated model generation can be a solution. However, generating Modelica code for detailed energy systems from scratch can become very complex, as completeness of the equation system in the generated code needs to be ensured. The superstructure model of the TransiEnt Library enables the

user to create checkable models of energy systems at the scale of coupled regions or more generally, coupled energy cells by choosing available technologies and setting (structural) parameters.

As shown on the left of Figure 2, the superstructure contains for each cell (region) subsystems for (1) the heating Grid including thermal storages, (2) electrical storages of different types, (3) gas storages, (4) power to gas, (5) CO_2 capture and storage, (6) available power plants of different types, (7) local renewable power generation and (8) local demand. The technologies available in each of these subsystems can be specified by the user with a replaceable package called technology portfolio. Here all technologies available in the TransiEnt Library as well as user specific models may be added by extending the existing base package.

All subsystems and their sub components are connected to the sectors they involve (electricity, heat, gas), e.g. power to gas is connected to electricity and gas. For the electric and gas sector, each cell provides a connector to the outside as well as three bus connectors for input, control and output signals. At present there is no outside connector for the district heating network (DHN). It is assumed that each DHN is located entirely inside its cell. The technical reason behind this is the lack of an automated DHN model generation, entirely from structural parameters, similarly to the electric grid connector in Figure 3. This may change in the near future, since the new modeling approach of section 3.3 offers more flexibility for model initialization and simulation performance.

In order to resolve larger spatial scales, an array of cells (regions) can be used consisting of `nRegions` cells. This is done in the so called portfolio mask model (Figure 2, center). For each of the `nRegions` cells (regions), its local energy system can be specified through structural parameter records: instance records can be used to disable technologies through conditional components in order to to adapt the energy cell model to a specific configuration, for example, to set up the available generation technologies. Enabled technologies can be further specified by parameter records, by setting e.g. the total power of each individual renewable generation technology or storage capacities of a given energy cell. For each technology an individual number of plants in a cell with their specific properties may be specified.

By construction the portfolio mask model provides an array of `nRegions` gas and electric grid connectors as well as a collection of all cell-buses in a master bus.

In order to model a scenario (Figure 2, right), the different cells of the portfolio mask model need to be connected by an electric and gas grid. Furthermore a control algorithm for the system has to be provided.

For constructing a simple electric grid model just from parameters, the `GridConnector` model can be used. There, as shown in Figure 3, an incident matrix specifies how many lines exist between two given regions.

Each line can be individually parametrized with respect to its voltage level, cable type, length and cross sectional area. For the gas grid a lumped model is used by default, however more detailed models may be provided by the user.

By using the masterBus, the control of the overall system is separated from the physics based cell models: all status information of the different cell components ("measurements") are aggregated towards the controller and the controller signals are propagated down to the individual cell component.

3.2 Integrated Electric Distribution Grids

The ongoing electrification of consumers in the electrical grid is one of the central consequences of the energy transition. This particularly affects consumers and producers in the distribution grid at medium- and low-voltage level, i.e. electric heat pumps (EHP), battery electric vehicles (BEV) and battery energy storage systems (BESS) as consumers and photovoltaic generators (PVG) as producers. These electrified components represent not only large additional loads or generators, but can also provide flexibility for the operation of the electrical grid, if according control and communication strategies are implemented. As these components couple the sectors, slower dynamics from the heat and mobility sectors are integrated into the electricity grid. To represent those dynamic effects of interest in grid simulations at system level, the library was expanded by several components. In order to be able to carry out simulations at system level, a scenario generator was developed that maps consistent development scenarios by parameterizing different low- and medium-voltage grid models on the basis of the SimBench data sets topologies (Meinecke et al. 2020; Wiegel et al. 2023). With the help of the scenario generator, a set of 17 preconfigured exemplaric grid models, 15 low voltage and 2 medium voltage, were implemented and added to the library. For the electric grid the RMS simulation approach was chosen. For all underlying novel component models, numerical efficiency was considered so that simulations of large non-aggregated grid models are feasible. The most important developments are briefly described below.

Building Heating System Building heating systems are increasingly being electrified by replacing gas or oil heating systems with heat pumps. The electrical energy demand of heat pumps is primarily dependent on the heat flow required to maintain a defined room temperature and for the provision of hot water. For this purpose, a building model was created in accordance with the enveloping surface method of DIN EN 12831, which calculates the heat flows occurring through walls, windows, floor and ceiling. The heat transfer coefficients required for this were taken from the current design standards for buildings. In addition to conductive heat transfer, the mass-bonded enthalpy exchange through room ventilation is also taken into account.

In order to perform system simulations using the developed model, numerical efficiency competes with model

DOI 10.3384/ecp218989

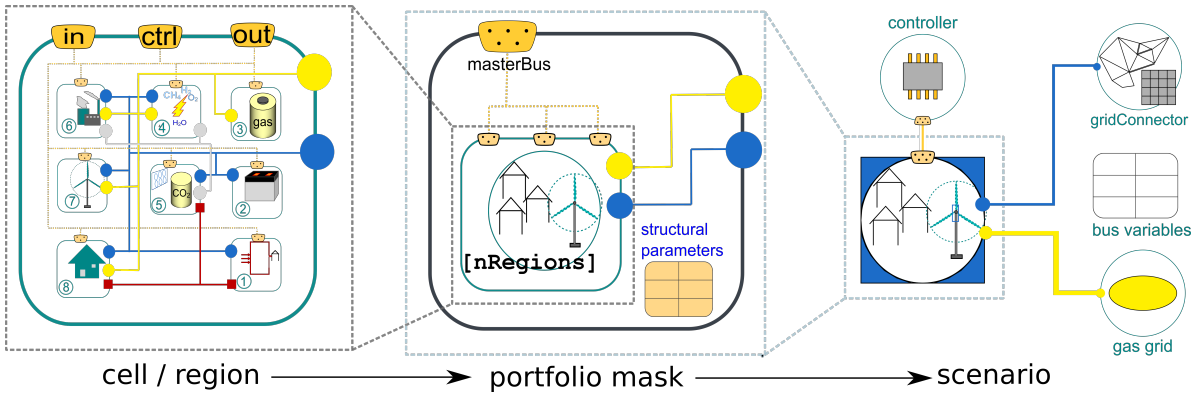


Figure 2. Workflow with superStructure model. Left: individual cell, center: array of cells, right scenario model with spatial resolution given by cell array and grid models. Color code: electric sector (blue connectors), heat sector (red connectors), gas sector (yellow connectors). Additionally the CO_2 cycle (gray connectors).

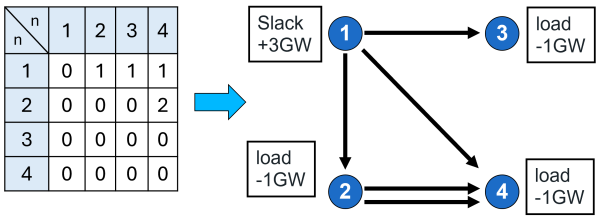


Figure 3. Using the `GridConnector` to model a simple electric grid, linking electric connectors of `nRegions` = 4 different cells/regions: the lines of the grid are specified by an incidence matrix and line parameters.

accuracy. Fluid dynamic simulations are therefore not useful at the system level for modeling of larger systems of decentralized heating units. Therefore, the behavior of the EHP in the system is mapped by a gray-box controller model. The basic structure of the model consists of a PID controller, a power splitter, and various components that represent the operating state under real-world conditions. The controller parameters were measured and validated on a real test facility as part of the European ERIGrid 2.0 project at the SYSLAB of the Technical University of Denmark (DTU). Part of the measurements are steady state operation and dynamic state changes. In addition to system simulation under regular operation, the behavior under external influence, for example by the grid operator when the load is reduced or limited, can be investigated using these models. Another use case is aggregators that use the flexibility of the components for energy trading on the electricity exchange.

Battery Electric Vehicles BEVs are characterized by two main aspects from the perspective of the electrical grid: availability according to the charging behavior and the electrical properties of the on-board charger. An empirical profile generator was therefore developed as part of the scenario generator, which maps the presence and driving behavior of individual owners. The electrical behavior of the on-board charger was investigated as part of the ERIGrid 2.0 project. This mainly affects the efficiency of the charger and the reactive power behavior. Moreover, the grid voltage dependency of the parameters and the dynamics of operating point shifts were analyzed and derived into the newly developed models (Steffen et al. 2024).

Battery Electric Storage System BESSs are usually controlled in such a way that self-consumption is maximized. This is achieved by measuring the load at the grid connection point and setting the operating point of the BESS according to the available power. Measurement takes place via a smart meter, which also takes measurement uncertainties into account.

Grid-level simulations The scenario generator developed allows system configurations to be created on the basis of parameters derived from the TYNDP scenarios to estimate future developments (Wiegel et al. 2023). These are automatically converted into an executeable Modelica model using a Python script. The modular structure allows the parameters to be adapted to the users individual needs. The electric grid simulation is based on a complex root mean square approach, since the fast electromagnetic transient effects are not in the scope of this modelling approach. The dynamic effects included originate from the models described above.

Use Case With the previously introduced component models, large distribution grid scenarios can be modelled and simulated. An example, a medium voltage ring based on the urban distribution grid model, named MV-urban-6, from the benchmark topology *1–MV-urban–6-no-sw* published by SimBench (Meinecke et al. 2020) is simulated. The medium voltage ring consists of in total 18 low voltage grid models, whereby 12 are different semiurban and 6 are urban low voltage grid topologies, in a future scenario. In total, over seven thousand individual component models contribute to this simulation.

Table 1 shows the share of included components. For the simulation the C-variable-coefficient ordinary differential equation solver (CVODE) with fixed integration step size of 60 seconds and tolerance of 10e-4 was used. 7 days of winter weather data for the location Berlin in

Table 1. Overview of grid components and their number in different topologies

Component	LV-semiurb-4	LV-semiurb-5	LV-urban-6	MV-urban-2
Transformers	1	1	1	19
Lines	42	109	57	1583
Nodes	43	110	58	1601
Households	41	104	111	1851
Battery Electric Vehicle (BEV)	16	48	48	832
Building Heating System (BHS)	20	47	49	831
Battery Energy Storage System (BESS)	1	4	4	69
Photovoltaic Generator (PVG)	5	15	18	278

January 2024 where chosen as input data sources. The data was acquired by the German Weather Service (DWD) (Reinert et al. 2025). The simulation time is close to 17 hours on an Intel I9 13900k CPU. The results are given in figure 4 and show the maximum loaded cables, node deviations and the power consumption of the low voltage grids connected to the medium voltage ring as well as the maximum cable load from the medium voltage grid. One can see, that the medium voltage grid in terms of line load is usually less loaded than the low voltage grid. The daily rising loads and falling voltages can be justified by the increased probability of BEV arriving in the late afternoon of a day. Additionally the influence of the electric heating within the BHS is noticeable with a slower dynamic due to the falling temperatures in the middle of the simulated week.

3.3 Simulation of large-scale district heating networks

In the following section, the developments of the heat components in the TransiEnt Library are described. There are two main use cases for the heat models of the TransiEnt library:

- the simulation of dwelling heating systems and
- the simulation of district heating networks (DHN).

Modelica is a suitable modeling language for DHN simulations because of its acausal character allowing physical modeling and easy model adaption. Moreover, it enables dynamic simulation, which allows the investigation of thermal inertia.

Until now, large-scale simulations using the TransiEnt Library have only been performed for electricity and gas grids. In contrast, the simulation of heating grids has been limited to a few components. This absence of large-scale examples has also been mentioned in literature (e.g. Schweiger et al. 2017). For this reason, the emphasis of recent developments has been on the simulation of large-scale DHN. Therefore, new modeling concepts for the simulation of DHN have been developed. Within the TransiEnt library two different approaches regarding the consideration of thermal dynamics in DHN are available - Euler and Lagrange. The Euler approach focuses on fixed locations inside a fluid field while the Lagrange approach is about tracking an individual fluid particle inside a fluid field. Both approaches are valid and have their flaws and assets.

The first one is described by (Westphal, Brunnemann, and Speerforck 2025) and is based on the discretization of the balance equations of momentum, mass, and energy. The presented modeling concept can be categorized as shown in Figure 5. DHN simulations are often categorized with regard to simulation type, calculation method, loop method, software environment, and control strategy. The modeling concepts consider thermal dynamics but neglect the hydraulic dynamics inside the network.

To achieve an efficient simulation performance and scalability for larger models, three key points were discovered and implemented in the component models using the Euler approach:

1. Avoidance of implicit non-linear systems of equations through a suitable consumer model and the strategic use of mass flow states, which is partly based on Zimmer (2020).
2. Discretization of only the energy balance to enable the use of a sparse solver.
3. Avoidance of unnecessary equations and function calls by assuming constant or fitted curves for the fluid properties.

The new models have been presented by (Westphal, Brunnemann, and Speerforck 2025) and used by (Vieth, Westphal, and Speerforck 2025). Further explanations regarding the modeling concept and how to achieve an efficient simulation can be found in (Westphal, Brunnemann, and Speerforck 2025).

The second concept is based on a plug flow Lagrangian approach and is used in the models `SinglePipe_LX` and `DoublePipePair_LX`. The same approach is used in the IBPSA library, which is, for instance, integrated into the Buildings library (Heijde et al. 2017). For the TransiEnt library, the model was adapted within the project IntegraNet (Benthin, Hagemeier, et al. 2020) based on the work of (Hägg 2016) and (Heijde et al. 2017) to enable usage within the TransiEnt framework. The model `SinglePipe_LX` comprises submodules for heat loss,

DOI 10.3384/ecp218989

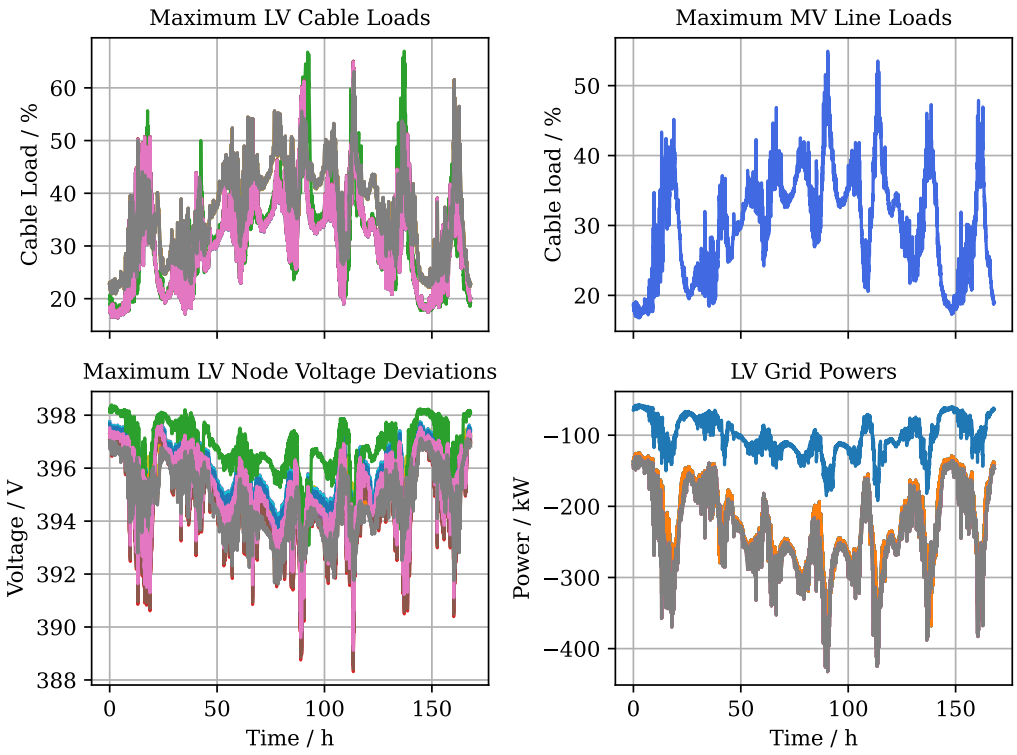


Figure 4. Illustration of the most loaded lines and nodes as well as the power consumption of the low voltage grids within the medium voltage ring simulation.

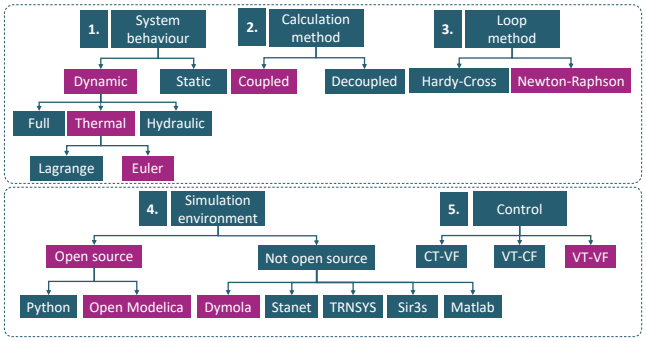


Figure 5. Categorization of modeling concepts for district heating networks based on the literature review of Kuntuarova et al. (2024)

residence time, pressure drop, and volume. It contains two heat loss components, one for the nominal flow and one for the reverse flow. Each of them implements the plug flow approach in a different flow direction. In the plug flow approach, thermophysical properties are evaluated solely at the pipe inlet and outlet, significantly reducing numerical complexity.

The heat loss rate $\dot{Q}$ is calculated as a function of the mass flow $\dot{m}$, the heat capacity of the medium c_p and difference of the inlet temperature T_i and the outlet temperature T_e:

$$\dot{Q} = \dot{m} \cdot c_p \cdot (T_\mathrm{e} - T_\mathrm{i}). \tag{1}$$

The outlet temperature T_e is calculated as shown in Equation 2 as a function of the residence time τ, the inlet temperature T_i, the ambient temperature T_a, the thermal resistance R, and the heat capacity of the heat carrier per meter C:

$$T_\mathrm{e} = T_\mathrm{a} + (T_\mathrm{i} - T_\mathrm{a}) \cdot e^{\frac{-\tau}{RC}}. \tag{2}$$

A key feature of the model is the physically consistent representation of the residence time, which is essential for capturing transient thermal effects in pipe systems. The residence time required for the heat loss calculation is determined in the residence time component using the `spatialDistribution()` function as described in (Heijde et al. 2017). The pressure drop across the pipe is computed using the Darcy-Weisbach equation, linearized for small mass flows.

The plug flow approach facilitates initialization by allowing initial values for physical parameters such as temperatures to be less precise. The approach itself does not account for delays in temperature changes due to the carrier pipe. Therefore, an optional volume component is added (Benonysson 1991; Hägg 2016; Heijde et al. 2017). Another module, `Pipe-Parameter`, provides a centralized interface for setting initial conditions and defining the geometric parameters of the pipe.

3.3.1 Use case: Using the piping network as a short-term storage

To prove the applicability of the component models, a dynamic simulation of a large-scale DHN with 2167 substations and a network length of 140 km is performed. The scope of the simulation is to use the piping network as a short-term storage to reduce peak heat production. This can be done by adapting the supply temperature of the network shortly before peak loads appear.

In Westphal, Brunnemann, and Speerforck (2025) the DHN topology used for this study has been introduced. Heat is generated by one large-scale heat producer, and the network is configured in a meshed topology. The main topology is derived from Frederiksen and Werner (2013) who classified different DHN based on the position of the heat producers. Distribution topologies have been based upon the work of Benthin, Hagemeier, et al. (2020) and designed with assumed heat fluxes. By combining the suggested main topology with the designed distribution topologies, a realistic heating network is created (see Figure 6). The mass flows are calculated by the consumer models with the use of a P-Controller, whose output depends on the room temperature.

To prove a decrease of peak heat production the heat flows required by the consumers are increased manually over a time span of four hours. In Figure 7, the generated heat flows can be seen for two scenarios. The blue graph describes the reference scenario where the supply temperature is held constant during the whole simulation. Therefore, the generated heat increases simultaneously with the heat load. The orange graph describes a scenario where the supply temperature was increased shortly before the peak load. It can be seen that the produced heat is increased when the supply temperature is raised. However, after a while, the produced heat starts to decrease again. The reason for this is that the increased temperature starts to reach the consumers with a time delay. Because of the higher supply temperature, the consumers need less mass flow and the generated heat flow starts to decrease. As a result, heat generation is decoupled from the heat load.

It can be concluded that the maximum peak generation is decreased due to the supply temperature adaptation. Therefore, peak generation can be avoided by using an advanced control strategy for the supply temperature and thus using the piping network as a short-term storage. This could, for example, be used for DHN operators to avoid costly peak generations like gas boilers and use more base load generation like combined heat and power plants (CHP).

3.3.2 Use case: Analysing future developments of a district heating network

The pipe models using the plug flow approach were used within the project IQDortmund (Gas- und Wärme-Institut Essen e.V et al. 2023) to simulate and analyze the district heating network of the city of Dortmund. This includes the assessment of the current state, as well as expansion and transformation scenarios for the period up to 2045. Using detailed simulations in combination with GIS-based modelling, several neighborhoods were evaluated in terms of future heating infrastructure development. A detailed model of the heating network with approximately 56 km of pipeline was created for simulating dynamic network behavior and future scenarios. To ensure robust initialization and improve numerical convergence in such a large-scale simulation, a node model is inserted between pipe segments. This model dampens rapid transients and decouples adjacent sections by introducing a time constant in the mass flow and energy balance equations.

The overall system model enables the investigation of changes in the system infrastructure like different heat sources or an expansion of the district heating network and their effects on the loading of the pipe infrastructure and resulting pressure losses. In the current state of the district heating network evaluated, a CHP plant serves as the primary heat source. In a future scenario, waste heat from a source located in the northern area becomes the main supply. Natural gas boiler plants are replaced by hydrogen boiler plants and a biomass heating plant. For larger potential waste heat sources, two heat pumps are installed. The shift in the primary heat generator results in a changed pressure distribution within the network (as shown in Figure 8), leading to increased loads of specific pipeline sections. Due to the intensified use of waste heat, a significant portion of the thermal energy has to be transported from the northern area to the rest of the network. As illustrated in Figure 8, this causes increased loading on two pipelines that are critical for connecting the northern section to the remaining network. For more details and further analyses on the district heating in Dortmund please refer to the final report (Gas- und Wärme-Institut Essen e.V et al. 2023).

4 Summary and Outlook

The transformation of energy networks and their requirements for economic efficiency, sustainability, reliability and resilience require dynamic simulations of energy systems with a large number of components at different levels. With the `GridConstructor` and the `Superstructure` two models are available in the TransiEnt library, which can be used to **easily** create aggregated energy systems at city or regional level. The focus of the GridConstructor is on simulating coupled energy grids at the district level. On the other hand, the Superstructure is used for simple modeling of coupled energy systems at the regional and supra-regional levels. While in recent years the focus of electrical modeling has been on extra-high voltage networks, new models are now available in the TransiEnt library, such as electrical prosumers, which are an integral part of medium and low voltage distribution grid. These models are capable of integrating the dynamic effects associated with the residential heating and mobility sector into electric grid simulations. In the specific use case, the utilization of a medium voltage network

DOI 10.3384/ecp218989

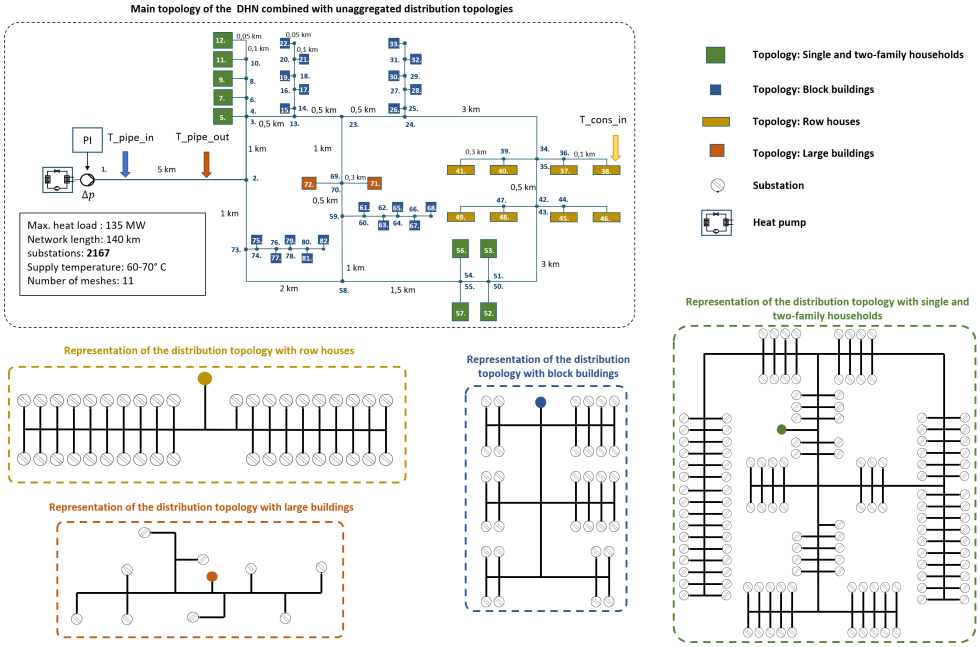


Figure 6. Topology of the investigated DHN from (Westphal, Brunnemann, and Speerforck 2025)

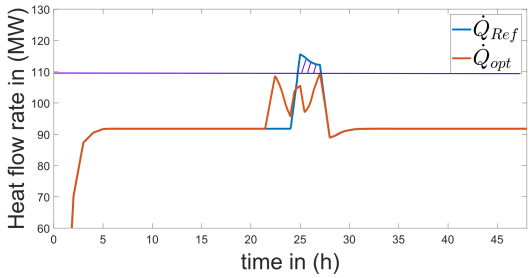


Figure 7. Heat flows of the producer for the reference scenario (blue) and the scenario with an adapted supply temperature (orange)

with over 7000 components is investigated.

Until now, the TransiEnt library has shown weaknesses in the simulation of large heating networks. With the two presented models, based on conservation equations on the one hand and the plug-flow approach on the other hand, large heating networks with e.g. over 2000 transfer stations and a network length of over 140 km can be simulated.

In addition to the standard extension of models, the future development of TransiEnt will also focus on enabling the use of the library with the OpenModelica platform. Furthermore, the documentation of the models is to be improved, access to the library simplified through tutorials and the exchange with the user community strengthened. In addition, hybrid models from Modelica and AI are to be given greater consideration.

The current version and Release 2.1.0 can be downloaded from the GitHub repository at `https://github.com/TransiEnt-official/transient-lib`. It is freely available under the terms of 3-clause BSD license. Further information about the library can also be found on the website `https://www.tuhh.de/transient-ee`.

Acknowledgements

The authors would like to thank the German Federal Ministry for Economic Affairs and Climate Action (BMWK) for funding the projects “Integranet“ (0324027B), “Integranet II“ (03EI1008B) and “EffizientEE“ (03EI1050A) in addition to the previous projects. The authors also greatly acknowledge the Hamburg University of Technology for funding the I^3 Lab project “CyentEE“ (`https://www.tuhh.de/cyentee/start`).

References

Andresen, Lisa, Carsten Bode, and Gerhard Schmitz (2018). “Dynamic simulation of different transport options of renewable hydrogen to a refinery in a coupled energy system approach”. In: *international journal of hydrogen energy* 43.42, pp. 19600–19614.

Andresen, Lisa, Pascal Dubucq, et al. (2015). “Status of the TransiEnt Library: Transient simulation of coupled energy networks with high share of renewable energy”. In: *11th International Modelica Conference 2015*. Vol. 118. Univ., pp. 695–705.

Andresen, Lisa and Gerhard Schmitz (2016). “Bewertung von power-to-gasanlagen mittels dynamischer Systemsimulation”. In: *Gwf-Gas þ Energ* 682, e9.

Benonysson, A. (1991). “Dynamic Modelling and Operational Optimization of District Heating Systems”. PhD thesis. Laboratory of Heating and Air Conditioning, Technical University of Denmark.

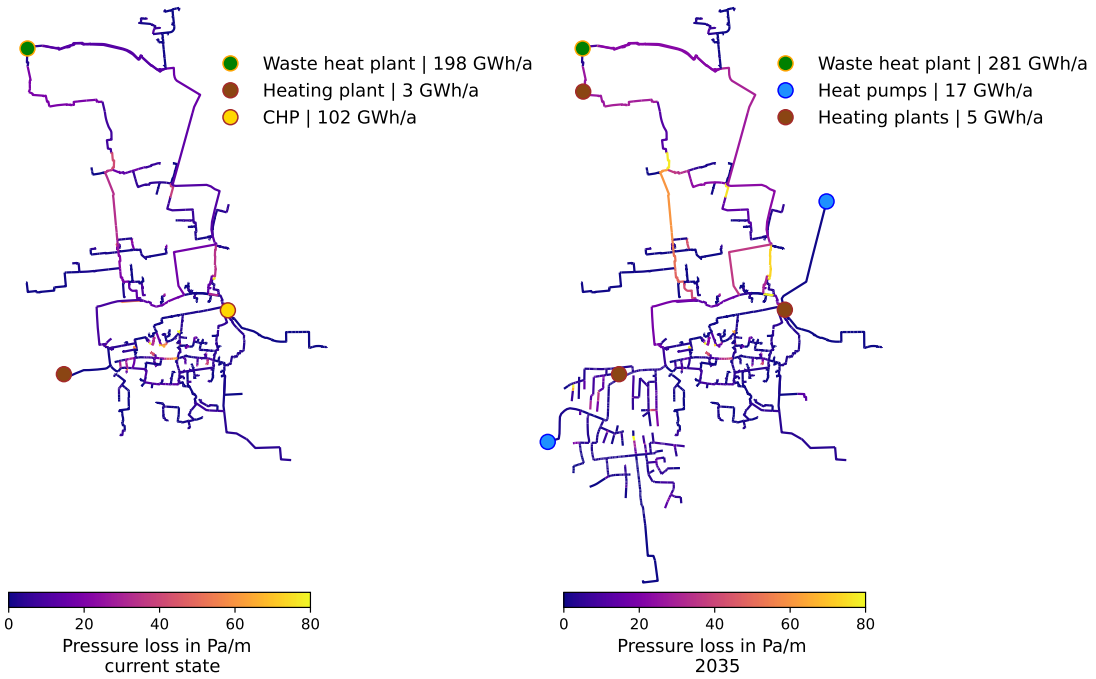


Figure 8. Average pressure loss of the network area in the current state and in the 2035 scenario based on (Gas- und Wärme-Institut Essen e.V et al. 2023).

Benthin, Jörn, Anne Hagemeier, et al. (2020-03). *IntegraNet - Integrierte Betrachtung von Strom-, Gas- und Wärmesystemen zur modellbasierten Optimierung des Energieausgleichs- und Transportbedarfs innerhalb der deutschen Energienetze*. Final Report. Oberhausen/Essen. URL: https://integranet.energy/wp-content/uploads/2020/04/IntegraNet-Abschlussbericht_V1.1.pdf (visited on 2020-08-20).

Benthin, Jörn, Annika Heyer, et al. (2019). "Demand oriented Modelling of coupled Energy Grids". In: *13th International Modelica Conference 2019*, pp. 59–66.

Bode, Carsten and Gerhard Schmitz (2019). *Influence of excess power utilization in power-to-heat units on an integrated energy system with 100% renewables*.

Bode, Carsten, Anne Senkel, and Gerhard Schmitz (2020). "Evaluating the resilience of integrated energy systems using dynamic simulation". In: *International Gas Union Research Conference IGRC (IGU 2020)*.

Dubucq, Pascal and Günter Ackermann (2015). "Frequency control in coupled energy systems with high penetration of renewable energies". In: *2015 International Conference on Clean Electrical Power (ICCEP)*. IEEE, pp. 326–332.

Dubucq, Pascal and Günter Ackermann (2017). "Optimal use of energy storage potentials in a renewable energy system with district heating". In: *Energy procedia* 135, pp. 158–171.

Frederiksen, Svend and Sven Werner (2013-01). *District Heating and Cooling*. 1st ed. Lund: Studentlitteratur AB. ISBN: 978-91-44-08530-2.

Gas- und Wärme-Institut Essen e.V et al. (2023). *EnEff:Wärme: IQDortmund: Konzeptionierung eines integrierten Wärmenetzes zur sektorenübergreifenden Quartiersversorgung in Dortmund*. Final Report. Essen/Dortmund. URL: https://www.gwi-essen.de/medien/publikationen/abschlussberichte/2024/iq_dortmund.pdf (visited on 2025-04-07).

Glombik, Sebastian Dominik and Felix Fromme (2024). *Real Options Analysis applied on Residential Energy Systems using least Squares Monte Carlo Simulation*. DOI: 10.52202/077185-0032.

Greve, Marco et al. (2021). *Gutachten zur Weiterentwicklung der Strom-Verteilnetze in Nordrhein-Westfalen auf Grund einer fortschreitenden Sektorenkopplung und neuer Verbraucher*. Ministry of Economic Affairs, Industry, Climate Action and Energy of the State of North Rhine-Westphalia. URL: https://www.wirtschaft.nrw/sites/default/files/documents/210609_nrw_verteilnetzstudie_final.pdf.

Hägg, Rickard (2016). *Dynamic Simulation of District Heating Networks in Dymola*. Tech. rep. Lund: Department of Energy Sciences Faculty of Engineering, Lund University.

Heckel, Jan-Peter and Christian Becker (2019). "Advanced Modeling of Electric Components in Integrated Energy Systems with the TransiEnt Library." In: *13th International Modelica Conference 2019*, pp. 157–079.

Heckel, Jan-Peter and Christian Becker (2020). "Investigation of the voltage stability in the integrated energy system of northern Germany". In: *Conference on Sustainable Energy Supply and Energy Storage Systems (Neis) 2019*. VDE, pp. 1–6.

Heckel, Jan-Peter, Stefan Möws, and Christian Becker (2022). "A Concept for Providing Control Reserve in Integrated Energy Systems". In: *2022 International Conference on Renewable Energies and Smart Technologies (REST)*. Vol. 1. IEEE, pp. 1–5.

Heckel, Jan-Peter, Anne Senkel, et al. (2022). "Investigation of dynamic interactions in integrated energy systems". In.

Heckel, Jan-Peter, Tom Steffen, and Christian Becker (2022). "Voltage Stability Risks Caused by Dynamic Interactions in

Integrated Energy Systems". In: *CIGRE 2022 Paris Session*. CIGRE.

Heijde, B. van der et al. (2017-11). "Dynamic equation-based thermo-hydraulic pipe model for district heating and cooling systems". In: *Energy Conversion and Management* 151, pp. 158–169. ISSN: 0196-8904. DOI: 10.1016/j.enconman.2017.08.072. URL: https://www.sciencedirect.com/science/article/pii/S0196890417307975 (visited on 2025-04-07).

Kuntuarova, Saltanat et al. (2024-01). "Design and simulation of district heating networks: A review of modeling approaches and tools". In: *Energy* 305, p. 132189. ISSN: 0360-5442. DOI: 10.1016/j.energy.2024.132189. URL: https://www.sciencedirect.com/science/article/pii/S0360544224019637.

Leitner, Benedikt et al. (2019). "A method for technical assessment of power-to-heat use cases to couple local district heating and electrical distribution grids". In: *Energy* 182, pp. 729–738.

Maier, Laura et al. (2024). "AixLib: an open-source Modelica library for compound building energy systems from component to district level with automated quality management". In: *Journal of Building Performance Simulation* 17.2, pp. 196–219.

Meinecke, Steffen et al. (2020-06). "SimBench—A Benchmark Dataset of Electric Power Systems to Compare Innovative Solutions Based on Power Flow Analysis". en. In: *Energies* 13.12, p. 3290. ISSN: 1996-1073. DOI: 10.3390/en13123290. (Visited on 2024-01-25).

Modelica Association (2025). *Modelica Libraries - A collection of free and commercial Modelica libraries*. https://modelica.org/libraries/. [Accessed 09-07-2025].

Peniche, Ricardo and Alfons Kather (2015). "Modellierung städtischer Energiesysteme. Heizkraftwerke, Fernwärmenetze und die Integration fluktuierender Erneuerbarer Energien". In: *Beckmann M., Hurtadi A (Hg.): Kraftwerkstechnik; Strategien, Anlagentechnik und Betrieb, Freiberg, Saxonia Standortentwicklungs-und-verwaltungsgesellschaft mbH*.

Pfafferott, Torge and Gerhard Schmitz (2003). "Implementation of a modelica library for simulation of refrigeration systems". In: *3rd International Modelica Conference 2003*, pp. 197–206.

Reinert, D et al. (2025). *DWD Database Reference for the Global and Regional ICON and ICON-EPS Forecasting System*. (Visited on 2025-04-03).

Schweiger, Gerald et al. (2017-01). "District heating and cooling systems – Framework for Modelica-based simulation and dynamic optimization". In: *Energy* 137, pp. 566–578. ISSN: 0360-5442. DOI: 10.1016/j.energy.2017.05.115. URL: https://www.sciencedirect.com/science/article/pii/S0360544217308691.

Senkel, Anne, Carsten Bode, Jan-Peter Heckel, et al. (2021). "Status of the transient library: Transient simulation of complex integrated energy systems". In: *14th International Modelica Conference 2021*, pp. 187–196.

Senkel, Anne, Carsten Bode, and Gerhard Schmitz (2021). "Quantification of the resilience of integrated energy systems using dynamic simulation". In: *Reliability engineering & system safety* 209, p. 107447.

Steffen, Tom et al. (2024-09). *Impacts of Distribution Grid Congestion Management on Charging Efficiency of Private Electric Vehicles*. URL: https://hdl.handle.net/11420/55290.

Vieth, Jonathan, Jan Westphal, and Arne Speerforck (2025). "A GIS-based Co-Planning Approach for District Heating Networks". In: *Energy Proceedings*. Vol. 50. Type: Conference paper. DOI: 10.46855/energy-proceedings-11423. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85209360724&doi=10.46855%2fenergy-proceedings-11423&partnerID=40&md5=8bba137897d56b841319254f5f8ccb0a.

Vojacek, Ales et al. (2023). "Status of the ClaRa Library: Detailed transient simulation of complex energy systems". In: *Modelica conferences*, pp. 617–626.

Westphal, Jan, Johannes Brunnemann, and Arne Speerforck (2025). "Enabling the dynamic simulation of an unaggregated, meshed district heating network with several thousand substations". In: *Energy* 322. Type: Article. DOI: 10.1016/j.energy.2025.135434. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-86000589421&doi=10.1016%2fj.energy.2025.135434&partnerID=40&md5=00b31ac6c0901f75494f2db7961fc49c.

Wetter, Michael et al. (2014). "Modelica Buildings library". In: *Journal of Building Performance Simulation* 7.4, pp. 253–270. DOI: 10.1080/19401493.2013.765506. URL: https://doi.org/10.1080/19401493.2013.765506.

Wiegel, Béla et al. (2023). "Towards a more comprehensive open-source model for interdisciplinary smart integrated energy systems". In: *2023 11th Workshop on Modelling and Simulation of Cyber-Physical Energy Systems (MSCPES), Texas, USA*. DOI: 10.1109/MSCPES58582.2023.10123432.

Wischhusen, Stefan, Bruno Lüdemann, and Gerhard Schmitz (2003). "Economical analysis of complex heating and cooling systems with the simulation tool hksim". In: *3rd International Modelica Conference 2003*.

Zhdanova, Maria et al. (2022). "Local Power Grids at Risk – An Experimental and Simulation-based Analysis of Attacks on Vehicle-To-Grid Communication". In: *Proceedings of the 38th Annual Computer Security Applications Conference*. ACSAC '22. Austin, TX, USA: Association for Computing Machinery, pp. 42–55. DOI: 10.1145/3564625.3568136.

Zimmer, Dirk (2020-01). "Robust object-oriented formulation of directed thermofluid stream networks". In: *Mathematical and Computer Modelling of Dynamical Systems* 26.3. Publisher: Taylor & Francis, pp. 204–233. DOI: 10.1080/13873954.2020.1757726.

DOI
10.3384/ecp218989

Dynamic modeling of a liquid piston compressor system including conjugate heat transfer

Mathieu Specklin[1] Elie Solai[2] Clémence Rouge[3] Michael Deligant[2]

[1]Cnam, Arts et Metiers Institute of Technology, LIFSE, Paris, F-75013, France
[2]Arts et Metiers Institute of Technology, Cnam, LIFSE, Paris, F-75013, France
[3]Université Bourgogne Europe, DRIVE UR 1859, 58000 Nevers, France

Abstract

Efficient and cost-effective hydrogen storage necessitates quick compression across significant pressure ranges (usually up to 700 bar), while keeping heat generation to a minimum during the process. This can be achieved by improved understanding of gas-to-wall heat transfer enhancement in hydrogen compression systems, careful design and operational optimisation. In this context, the present paper introduces a 0D/1D lumped numerical model of a liquid piston compressor for hydrogen applications. Heat transfer is considered as (i) convective at the liquid-gas interface, (ii) conductive within the gas volume based on a 1D approach accounting for thermal stratification, and (iii) as conjugate at the gas-to-wall interface. To achieve a pressure ratio of 30 (from 15 to 450) bar, at a power density of approximately $65 kW/m^3$, the compression energy cost reaches $1.85 kWh/kg$. Further, various standard pressure vessels materials with different thermal and mechanical properties are considered, highlighting the potential compromise between lightweight and thermal efficiency.

Keywords:liquid-piston compressor, hydrogen, conjugate heat transfer

1 Introduction

Hydrogen holds great promise as an energy carrier that can progressively transition energy systems from relying on limited fossil fuels to utilizing sustainable renewable sources. It seeks to replace natural gas for domestic space heating and hot water, and enable long-range clean transportation. A major hurdle is achieving high-density hydrogen storage, particularly for hydrogen-powered vehicles that need to compete with the quick refueling and extensive range of traditional liquid fuel engines. A significant technical and economic obstacle for the transportation sector is the need to develop affordable compression, storage, and dispensing (CSD) stations outfitted with high-pressure compressors. Ideally, the hydrogen compression process should be isothermal, both to reduce the input work required and maintain the hydrogen delivery temperature very low (close to -40^oC). With limited heat transfer from the gas, this however comes at the expense of power density. Piston compressors are typically used in CSD stations as they are able to achieve high pressure ratios. In order to optimise simultaneously the piston compressors efficiency and power density, a compromise must be found.

Various methods have been developed to achieve near-isothermal compression in piston systems. Saadat et al. have concentrated on optimizing compression and expansion processes and maximizing heat transfer (Saadat, P. Li, and Simon 2012). Spraying liquid droplets can also enhance heat transfer by increasing the surface area between the gas and a secondary cooling fluid (Sapin et al. 2018). However, the energy required for injecting liquid droplets rises significantly with pressure. Another approach involves using phase change materials (PCMs) in the reservoir walls, where the heat from gas compression is stored isothermally as latent heat (Mazzucco et al. 2016). Recent focus has been on liquid-piston compressors, which replace the traditional reciprocating solid piston with a column of liquid, typically water, driven by a hydraulic pump. These systems offer excellent sealing and compatibility with heat transfer enhancers (Van de Ven and P. Y. Li 2009). An example of such enhancers is porous metal foams with high thermal inertia, which increase the heat transfer surface area when placed in the tank (Wieberdink et al. 2018). Finally, Specklin et al. investigated the potential of heat transfer augmentation with forced internal convection (Specklin et al. 2022).

In this paper, a comprehensive dynamic model is derived and used to investigate the role of the conjugate gas-to-wall heat transfer in the final energy cost of high pressure hydrogen compression. The model is based on the combination of 0D zonal approaches and a 1D discretization technique of the gas volume using elements from the Modelica Standard Library (MSL) (*MSL* 2023), in order to account for the different energy balances. The effect of different materials' thermal properties onto the heat transfer efficiency is put in parallel with their mechanical properties, especially the density, in order to give insight on potential optimal configurations. The study is introduced by presenting the composition and operation of a dual system of liquid piston compressors. Then, the main sub-models with their inherent governing equations are detailled. Finally, the thermodynamical behaviour of the system is hilighted and the final energy costs are assessed for different

wall materials.

2 Modeling methodology

The current numerical model is constructed following the same principle as the full scale setup described in (Solai, Specklin, and Deligant 2025), originally working with air. A simple zonal 0D approach was proposed and used to assess the system compression efficiency through uncertainty analysis. This approach is extended here, the main improvements being:

- mimicking the dynamic behavior of the full test bench which includes a dual system of liquid piston,
- adapted for hydrogen gas,
- proposing a 1D discretization of the time-varying hydrogen gas volume accounting for temperature gradients,
- based on an innovative extension of the thermal components of MSL.

This modeling strategy is implemented and tested with OpenModelica (*OpenModelica* 2023).

2.1 Liquid piston compressor system

The present liquid piston compressor is designed with a dual system of pistons, which facilitates the compression-expansion operation. The full system is represented in a schematic diagram in Figure 1. The liquid piston compressor is operated via the control of four solenoid valves. The opening of the valves is cyclic and the switch is determined based on the water level. The solenoid valves operate in parallel by pairs, as water from one tank is pumped into the other one. The check valves at the top of the reservoirs allow to bound the internal pressure between the two extrema given by the boundary conditions. Hence, as the water level increases in one of the cylindrical tank, the gas is being compressed. Once the final (or maximum) pressure is reached during compression, the corresponding check valve opens and the compressed gas is discharged. During the expansion process, the water level decreases and consequently the pressure too. Similarly, if the initial (or minimum) pressure is reached, the opening of the check valve occurs and the tank is filled with gas at this minimum pressure.

The liquid-piston system model shown in Figure 1 is constructed from components from MSL and from the in-house developed library based on MSL components (*MSL* 2023). Check valves, solenoid valves and the volumetric pump are considered as ideal. The thermodynamic behaviour of both left and right liquid piston cylinders are described in the next section.

2.2 Components modeling and governing equations

The liquid piston gas-spring arrangement, described in Figure 2, is initially filled with hydrogen at $P_0 = 15$ bar

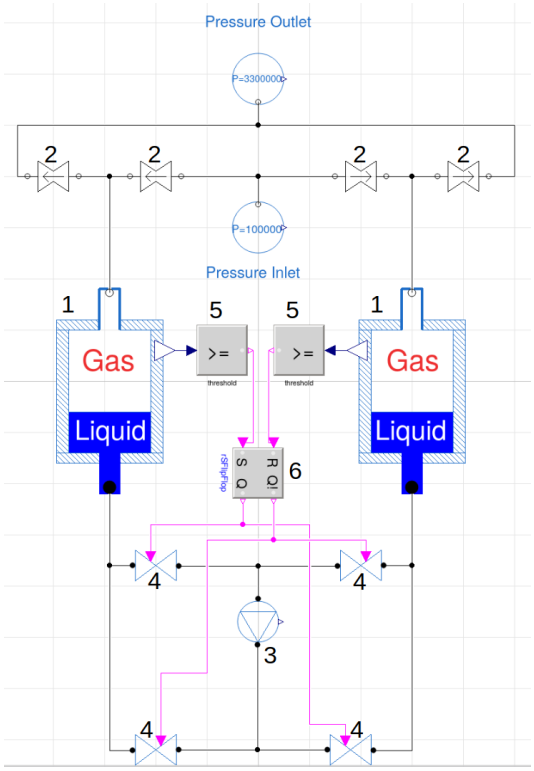


Figure 1. Schematic view of the experimental setup. 1) liquid piston tanks. 2) check valves. 3) volumetric pump. 4) solenoid control valves. 5) level threshold. 6) logical Reset-Set flipflop.

and at $T_0 = 20$ °C. Water is injected to compress the gas contained within the cylinder up to a pressure of 450 bar, which corresponds to a pressure ratio of 30. During compression, the hydrogen is heated up by the boundary work applied by the liquid piston motion, while being cooled down by the gas-to-solid and gas-to-liquid heat transfers. Further, heat transfer by conduction within the hydrogen gas is considered in the vertical direction.

The liquid piston cylinder model is composed of two sub-models, one for both time-varying gas and liquid volumes, as illustrated in Figure 3. Conjugate heat transfer is considered with two half thermal resistor and a heat capacitor representing the thermal inertia of the wall.

The outer half resistance includes both conductive thermal resistance through half thickness of the solid wall and the convective thermal resistance with the ambient. The inner half resistance includes the same conductive resistance and convective thermal resistances with the gas volume and the liquid volumes. The conductive resistances consist of resistances due to the lateral cylinder surface (Patil, Acharya, and Ro 2019). The convective thermal resistance with the ambient is governed by a heat transfer coefficient fixed to a standard value of $5W.K^{-1}.m^{-2}$ for natural convection. Finally, the inner convective resistances associated to both fluids are time-dependent and their inherent heat transfer coefficients are computed based on Reynolds-Prandtl correlations:

DOI 10.3384/ecp2181001

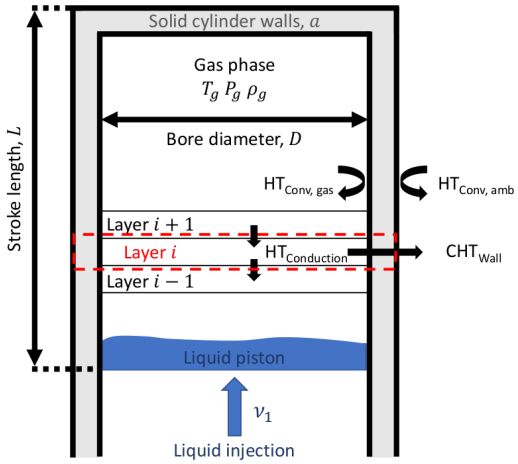


Figure 2. Schematic view of the liquid-piston cylinder involving 1D discretization of the gas volume.

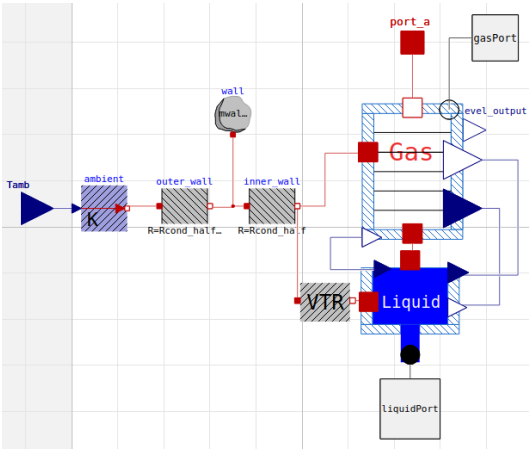


Figure 3. Schematic diagram of the liquid-piston cylinder model and its sub-models. The following variables are transfered between the sub-models of the gas and liquid volumes: the gas pressure P_g, the liquid volume V_l and the gas-to-liquid heat transfer coefficient h_{gl}.

$$Nu = aRe^b Pr^c \tag{1}$$

where Nu is the Nusselt number. More specifically, for the liquid-to-wall heat transfer, a, b and c coefficients are fixed considering standard correlation for forced convection in laminar regime over a flat plate. As for the gas-to-wall heat transfer the coefficient values are adjusted based on the uncertainty quantification approach described in (Solai, Specklin, and Deligant 2025) for the same piston liquid system operating at similar conditions, yet with air. Both Reynolds Re and Prandtl Pr numbers are computed based on the instantaneous values of the thermophysical gas properties, which are all obtained with the hydrogen ideal gas media from MSL.

The volume of the gas is further decomposed by an array of layers that encompass the internal convective heat transfer described above. This 1D approach allows to account for the thermal gradient in the vertical dimension.

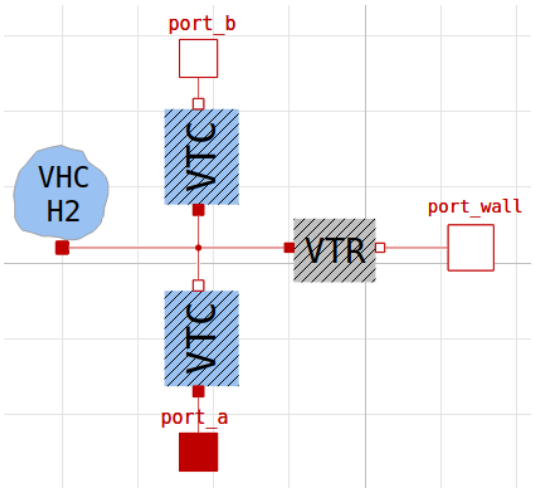


Figure 4. Schematic diagram of the gas layer element which includes convective thermal resistance with the adjacent wall. VHC stands for Variable Heat Capacitor, VTC stands for Variable Thermal Conductor and VTR stands for Variable Thermal Resistor.

A gas layer is thus made of the following element: a heat capacitor for the mass of hydrogen, two thermal conductors for conduction with the bottom and top gas layers, and the thermal resistor accounting for gas-to-wall heat transfer. The layer composition is shown in Figure 4. This is a modeling approach typically used for wall elements (Şeren and İpek 2023). In order to obtain a 1D discretization of the gas volume, these elements are defined as an array and connected successively. Another possibility lies in the direct resolution of the 1D partial differential equation such as in (Maccarini et al. 2023). In the present paper, the gas layer elements account for the time variation of the gas volume and thermodynamical properties during compression, expansion, discharge and admission stages. Hence, the different thermal components are extended versions of the MSL equivalent components with time-varying inputs governing their thermodynamical properties.

The variable heat capacitors defining the gas layers are adapted to satisfy the first law of thermodynamics:

$$\frac{dU_g^i}{dt} = \dot{m}_g^i \mathbf{h_g^i} - P_g^i \frac{dV_g^i}{dt} \tag{2}$$

where U_g and $\mathbf{h_g}$ are respectively the internal energy and the specific enthalpy of the hydrogen gas at pressure P_g and for a volume V_g. During the compression or the expansion stage strictly, the first term of right hand side in Eq. 2 depending on the mass flow rate $\dot{m}_g$ is null as the valves are closed. The superscript i stands for the i-th gas layer. Heat transfer by conduction with the adjacent layer is taken directly into account with the two thermal conductors, while conjugate heat transfer through the solid wall is taken into account through the thermal resistor. Thus equation 1 is layer dependent.

Heat transfer between the highest gas layer and the top wall surface is neglected due to the higher thickness of the

tank there. At the interface between the lowest gas layer and the liquid volume, heat transfer $\dot{Q}_{gl}$ is defined as:

$$\dot{Q}_{gl} = S_p h_{gl}(T_l - T_g) \tag{3}$$

S_p is the piston area and h_{gl} is coefficient of gas-to-liquid heat transfer, also estimated with a Reynolds-Prandtl correlation.

The number of gas layers n is fixed to 20, which has shown to be sufficient to predict the temperature gradient and to achieve independence of the results on n. The layers are defined with identical hydrogen mass at all time, and their volumes can thus be different depending on the thermodynamical properties of the gas within the layer. To ensure that the pressure is identical in all layers, a stable dynamics procedure is implemented such as in (Srivatsa and P. Y. Li 2018), which depends on a small positive number λ and reads:

$$\frac{d(P_g^{i+1} - P_g^i)}{dt} = -\lambda(P_g^{i+1} - P_g^i) \tag{4}$$

Here, $\lambda = 1e^{-4}$ is sufficient to force the pressure to be equal in all gas layers. As for the liquid volume finally, a 0D zonal approach is used, for which the volume follows the first law of thermodynamics:

$$\frac{dU_l}{dt} = \dot{m}_l \mathbf{h_l} + \dot{Q}_{lg} \tag{5}$$

where $\dot{Q}_{lg} = -\dot{Q}_{gl}$ and $\mathbf{h_l}$ is the specific enthalpy of water. Due to the high thermal inertia of liquid water, its temperature increase remains much below 1^oC in all present simulations, however considering a fixed inlet temperature within the volumetric pump.

2.3 Material selection

Four materials were selected, representing both standard pressure vessels materials and materials with both high thermal effusivity and thermal diffusivity. Commercially available high-pressure vessels are all based on the use of aluminium, steels and carbon fiber reinforced plastics. The study focuses on structures based onto those three main types of materials. In the current study, both the thermal and mechanical properties of materials are considered, a priori, with the same level of importance regarding their impact on the compression energy consumption. Hence, best performing grades of steel and aluminium were chosen in terms of thermal effusivity and diffusivity and in terms of yield strength to density ratio (21.2.0 n.d.): AISI 316L, an austenitic stainless steels, and Al 7068, a 7000 aluminium series. To complete the selection, as metal hybrids and composites are also good material candidates for application as high-pressure vessels (Solazzi and Vaccari 2022) (Bhattacharyya, El-Emam, and Khalid 2022) (Liu et al. 2012), the following two materials are also investigated: a metal matrice composite (MMC), based on an aluminium matrix filled with carbon fibers, and a carbon fiber reinforced polymer (CFRP),

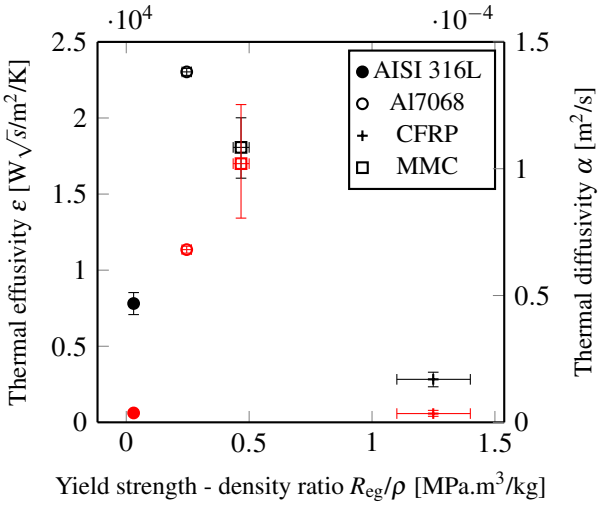


Figure 5. Variations of the selected materials thermal and mechanical properties. Thermal effusivities are represented in black while thermal diffusivities are represented in red.

based on an epoxy matrix reinforced by high strength unidirectional carbon fibers.

The mean thermal and mechanical properties of the four materials are presented **Table 1** and **Table 2**, respectively. **Figure 5** presents the variations of the thermal diffusivity and effusivity of those materials as a function of the ratio between the material yield strength and its density, thus relating the thermal and the mechanical properties of interest for this study.

For each material selected, the minimum wall thickness is determined by both the size of the reservoir and the targeted maximum gas pressure $P_{g,max}$. Thickness and diameter of the pressure vessels are the two main design criteria to ensure strength (Vullo et al. 2014). In the present study, the diameter of the vessel is fixed by the volume of gas that it must contain. Therefore, following recent ISO standard 9809-1:2019 for high pressure gas cylinder (*ISO 9809-1:2019: Gas cylinders - Design, construction and testing of refillable seamless steel gas cylinders and tubes - Part 1: Quenched and tempered steel cylinders and tubes with tensile strength less than 1 100 MPa* 2019), the cylinder wall thickness is obtained from the Lamé von Mises formula. It varies linearly with the vessel diameter D following:

$$a = \frac{D}{2}\left(1 - \sqrt{\frac{10FR_{eg} - \sqrt{3}P_{g,max}}{10FR_{eg}}}\right), \tag{6}$$

where R_{eg} is the minimum guaranteed value of yield stress and $F = min(\frac{0.65}{R_{eg}/R_{mg}}, 0.85)$ is a safety factor which depends on both R_{eg} and R_{mg}, the minimum guaranteed value of tensile stress. Assuming a perfect cylindrical shape for the piston, which is a general simplification of the real design of pressure vessel, the volume of material required is directly deduced, and so is the total mass required. Figure 6 shows this mass normalized by the max-

DOI
10.3384/ecp2181001

Table 1. Mean thermal properties of the selected materials.

Thermal property	Al 7068	CFRP	MMC	AISI 316L
Thermal conductivity $[W/m/K]$	190	5.25	182	15
Specific heat capacity $[J/kg/K]$	975	971	795	510
Density $[kg/m^3]$	2850	1565	2250	7970
Thermal diffusivity $[m^2/s^{-1}]$	6.81E-05	3.45E-06	1.02E-04	3.69E-06
Thermal effusivity $[J/m^2/s^{1/2}/K]$	23030	2820	18070	7810

Table 2. Mean mechanical properties of the selected materials.

Mechanical property	Al 7068	CFRP	MMC	AISI 316L
Yield strength [MPa]	702	1955	1050	240
Tensile strength [MPa]	740	1955	1050	522

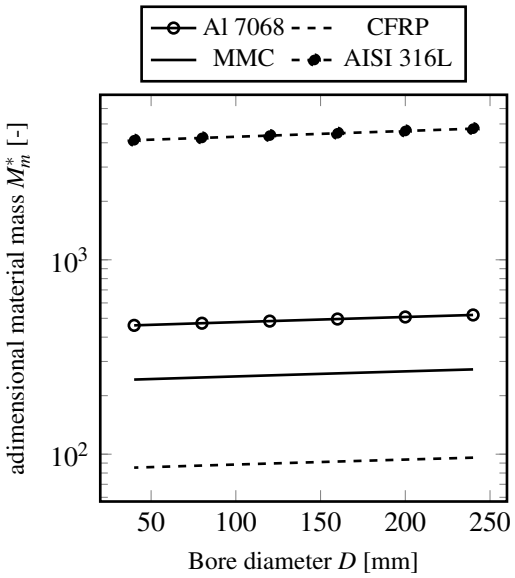


Figure 6. Variations of the selected materials mass required to sustained a 450 bar compression for diffrent cylinder diameter.

imum mass of gas (where the gas density ρ_g is taken at the initial baseline conditions). It highlights that the decrease of the piston thickness compensates the supplementary material needed for smaller piston. Hence, looking strictly only at the material mass required, the smaller the piston, the better it is.

3 Results

The conjugate thermal model described in the previous section has been used to predict the compression work required and heat production (through the effective temperature ratio) during the compression strokes. Salient results are presented in this section. First, time-resolved data are inspected in a baseline configuration. The influence of the stroke speed on the thermal gradient within the gas volume and compression work required are investigated. Second, the sentivity of the energy compression cost to the wall material is assessed.

The energy compression cost $\bar{C}$ is defined based on the energy requirement of the hydraulic pump. Further, it is

Table 3. Geometrical, initial, and operating conditions of the liquid piston system.

Geometrical conditions	
Cylinder bore (internal)	240 mm
Cylinder stroke	1 m
Initial conditions	
Initial hydrogen pressure	15 bar
Initial hydrogen temperature	20 ^oC
Initial water level	0.4 m (L) & 0.6 m (R)
Initial water temperature	20 ^oC
Operating conditions	
Overall pressure ratio	30
Injection flowrate	0.8 L/s
Ext. ambiant temperature	20 ^oC
Ext. heat transfer coefficient	3 $W.K^{-1}.m^{-2}$

defined per unit of mass of hydrogen discharged at the targeted pressure. The cost thus reads:

$$\bar{C} = \frac{\int Q_v |P_{g,left} - P_{g,right}|}{\int (q_{m,left} + q_{m,right})} \tag{7}$$

where q_m are the mass flow rates through the two exhaust check valves. The energy compression cost is assessed when the thermal transient behaviour of the system is over and a purely periodic state is reached. A final water level threshold of $99,5\%$ of the stroke length is considered, triggering the switch from the discharge stage to the expansion stage. This level corresponds directly to a dead volume of $0,5\%$ within both cylinder tanks. To sum up, Table 3 lists the geometrical, initial, and operating conditions of the present system in the baseline scenario.

3.1 General results

Figure 7 presents the evolution of different thermophysical variables for fixed regime of pump operation, corresponding to a water flow rate of $Q_v = 0.8L/s$. Pressure rise is shown for both cylinders, as well as the temperature evolution of the left cylinder wall, and, bottom and top gas layers. Vertical delimiters separate the four stages of the compressor system: (i) a long compression stage higlighting the polytropic evolution of pressure and temperature, (ii) a short discharge stage where hydrogen is ex-

hausted at constant pressure and fluctuating temperature, (iii) a short expansion stage where the remaining mass of hydrogen within the dead volume is expanded until the piston pressure reaches the inlet pressure, and finally (iv) a long admission stage where the hydrogen is filled again at constant inlet pressure, and which corresponds to the compression and discharge stage of the other piston. Pressure and temperature are normalized by their initial values (15bar, 20^oC). Aluminium is considered for the material wall, its maxium temperature reaching 35.6^oC at the end of the compression, i.e an extra 15^oC. As for the gas, the temperature ratio reaches 2.11 in the highest gas layers. This finally leads to a compression cost of $1.85kWh/kg$. Due to the heat conduction within the gas volume, the hydrogen temperature is not uniform as shown by the lower temperature obtained in the first layer ($R_T = 1.73$ at maximum).

Figure 8 shows how reducing the pump flow rate allows for more time to cool down the gas during compression, and reduces the vertical temperature gradient. In the second configuration, the flow rate has been divided by a factor height, and so is the power density which is proportional to the stroke duration. The temperature ratio reaches 1.88 at maximum at the end of the compression in this case. It corresponds to an energy cost of compression of $1.73kWh/kg$, representing a 8% cost reduction in comparison to the high flow rate scenario.

3.2 Sensitivity to the cylinder material

In this section, the energy cost of compression obtained with the present model is compared for the four different wall materials considered. The results are shown in Figure 9 along with the associated maximum temperature ratio at the end of the compression stage. Two scenarios are considered. For the first scenario, the internal gas-to-wall heat transfer is modeled with the Reynolds-Prandtl correlation calibrated for this specific liquid piston system in (Solai, Specklin, and Deligant 2025). Results (in light grey in Figure 9) highlight the negligible influence of the wall material in these conditions. The maximum relative difference for the compression cost is below 0.3%. Although the materials exhibit different thermal properties, the relatively short stroke time (around $56s$) and low gas-to-wall convective heat transfer coefficient (at maximum $340W/m^2/K$ at the end of compression and $100W/m^2/K$ in average) do not allow for the wall to affect the temperature rise within the cylinders.

In the second scenario, the heat transfer coefficient has been increased and fixed to $1500W/m^2/K$. This high value corresponds to what can be achieved when aiming at heat transfer enhancement through forced convection, as detailed in (Specklin et al. 2022) where the presence of a high speed fan in the cylinder is modeled. With the present model, this forced convective cooling reduces the energy cost of compression from $1.85kWh/kg$ to approximately $1.55kWh/kg$. Further, in this configuration, the influence of the wall material type becomes visible. Hence, a 2% difference is found in the energy cost of compression between the stainless steel and the carbon fiber reinforced polymer. Their thermal diffusivities and effusivities being relatively close, it shows that it is the thermal inertia which actually drives the energy cost among the material properties. The CFRP heat capacity is 16 folds smaller than the AISI 316L heat capacity, due to its high strength properties that allows for thinner wall and thus lower mass. Hence, after a thermodynamic periodic state being reached, the CFRP wall experiences temperature variations from 27^oC to 62^oC during a stroke, while the stainless steel wall only exhibits temperature periodic oscilations from 40^oC to 42^oC only.

To conclude, although it seems low, a 2% difference in the energy cost of compression can quickly add-up for expensive processes such as high pressure hydrogen compression. This result suggests then that there may be an interesting trade-off between the material cost for the compression system and the energy cost of the compression process.

4 Discussion

The results obtained with the present model can be put in perspective with those from the CFD investigations from (Specklin et al. 2022). In the latter, a lower energy cost of compression was obtained, $1.67kWh/kg$ at a power density of $540kW/m^3$, for an identical hydrogen compression configuration. There are however two differences in the modeling techniques that all lead to an overestimation of the cost in the present case or an underestimation in (Specklin et al. 2022). First, a Nusselt-Peclet correlation is used in (Specklin et al. 2022) for the unsteady gas-to-wall heat transfer, leading to higher coefficent values than the Reynolds-Prandtl correlation used in the present case. Second, Specklin et al. neglected the conjugate heat transfer and assumed an isothermal conditions for the internal wall which could slightly overestimates its cooling power. In our future studies, the Nusselt-Peclet correlation, together with a real gas media or Equation Of State, will be implemented in Modelica and will be used to bridge the modeling gap. Finally, a common limitation to both lumped model and CFD approaches is the omission of the liquid to vapor phase change during the process. The significance of this phenomena has yet not been clearly quantified in literature (Specklin et al. 2022) and certainly depends on gas nature and operating conditions. With air at low pressure ratio, the impact of evaporation seems to have a limited influence on the energy cost of compression (Solai, Specklin, and Deligant 2025). At high pressure ratio, it was shown theoretically by (Srivatsa and P. Y. Li 2018) that mass transfer between liquid water and vapor can significantly affect the maximum temperature reached during compression, and thus the energy cost. However, Wieberdink et al. (Wieberdink et al. 2018) did not mention this issue for instance during the operation of their liquid-piston system for air compression at similar ratio

DOI 10.3384/ecp2181001

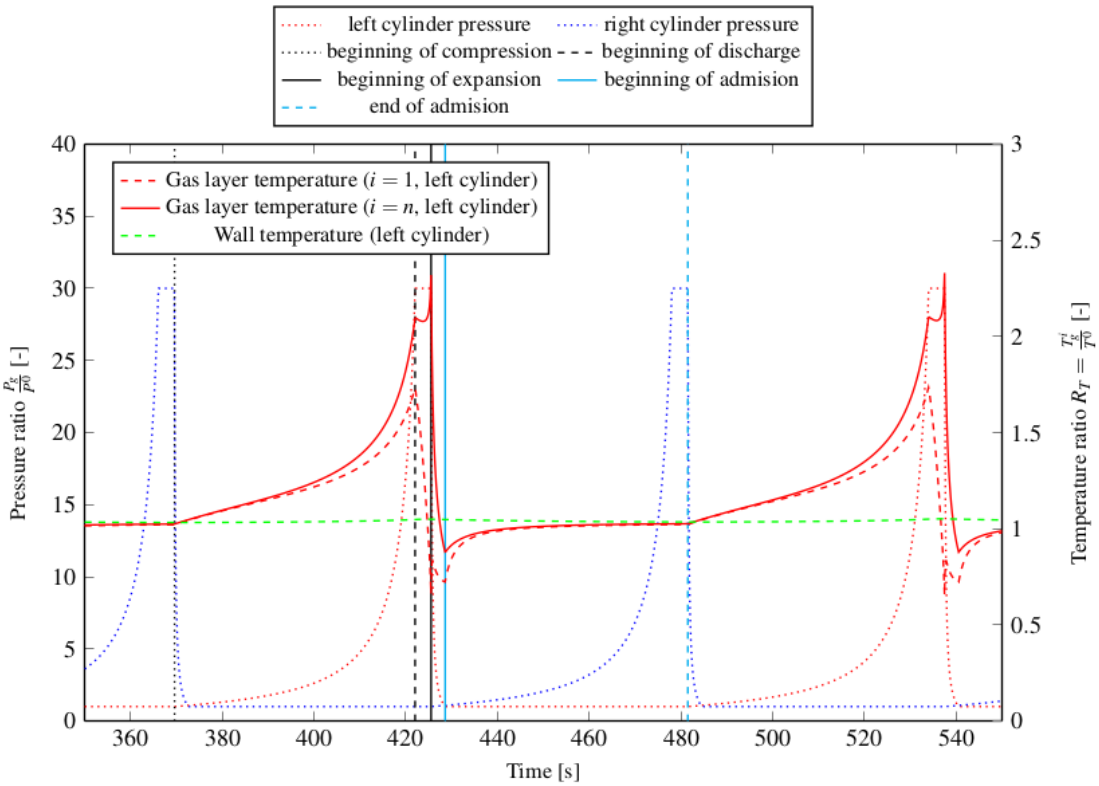


Figure 7. Evolution of pressures and temperatures obtained for $Q_v = 0.8L/s$ with aluminium wall.

and maxima of pressure. To avoid this issue, ionic liquid can be used to compress the gas, as they do not have a vapor pressure (Wang et al. 2018), or additional separator technologies to recover the purity of the materials but with additional costs.

5 Conclusion

In this paper, an object-oriented modeling methodology using the Modelica Programming Language has been presented for a high pressure liquid piston compressor system. The model is based on an existing experimental test bench using air, and has been extended for hydrogen gas. The modeling strategy involves considering the temperature gradient within the compressed gas, using a 1D discretization technique based on MSL elements. Further, conjugate heat transfer through the piston wall is considered and a specific attention is paid to the material choice. In its present state, the lumped model overestimates the energy cost of compression, in comparison to figures obtained with more accurate numerical investigations. However, the developed dynamic model is able to respond to changes in operating conditions and wall material types, and by means of this, it could be used for optimizing processes. In particular, promising insights can be expected from possible trade-off between material cost and energy cost, which could be investigated in full Life Cycle Analysis (LCA) or eLCC (environmental Life Cycle Costs) studies.

Acknowledgements

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

21.2.0, Ansys Granta EduPack software (n.d.). “ANSYS, Inc., Cambridge, UK, 2021 (www.ansys.com/materials)”. In: ().

Bhattacharyya, Rupsha, Rami S El-Emam, and Farrukh Khalid (2022). “Multi-criteria analysis for screening of reversible metal hydrides in hydrogen gas storage and high pressure delivery applications”. In: *International Journal of Hydrogen Energy*.

ISO 9809-1:2019: Gas cylinders - Design, construction and testing of refillable seamless steel gas cylinders and tubes - Part 1: Quenched and tempered steel cylinders and tubes with tensile strength less than 1 100 MPa (2019-10). Standard. Geneva, CH: International Organization for Standardization.

Liu, P F et al. (2012). “Numerical simulation and optimal design for composite high-pressure hydrogen storage vessel: A review”. In: *Renewable and Sustainable Energy Reviews* 16.4, pp. 1817–1827.

Maccarini, Alessandro et al. (2023). “Low-order aquifer thermal energy storage model for geothermal system simulation”. In: *Proceedings of the 15th International Modelica Conference 2023, Aachen, October 9-11* 204.October, pp. 389–396. DOI: 10.3384/ecp204389.

Mazzucco, Andrea et al. (2016). “Integration of phase change materials in compressed hydrogen gas systems: Modelling and parametric analysis”. In: *International Journal of Hydrogen Energy* 41.2, pp. 1060–1073. ISSN: 03603199. DOI:

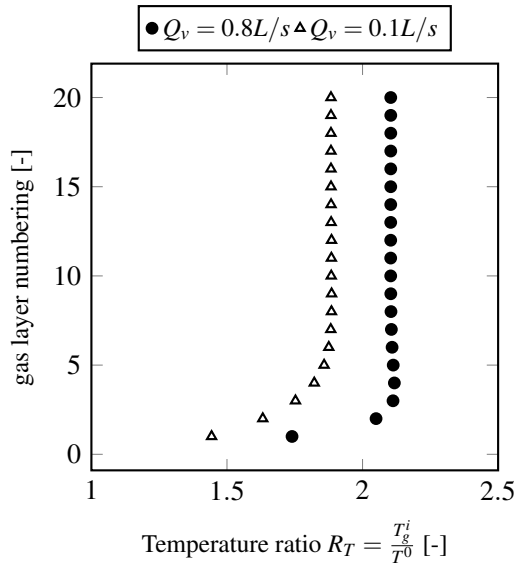


Figure 8. Non-dimensional temperature profiles along the vertical direction of the cylinder tank at the end of the compression stroke, for two different pump flow rates.

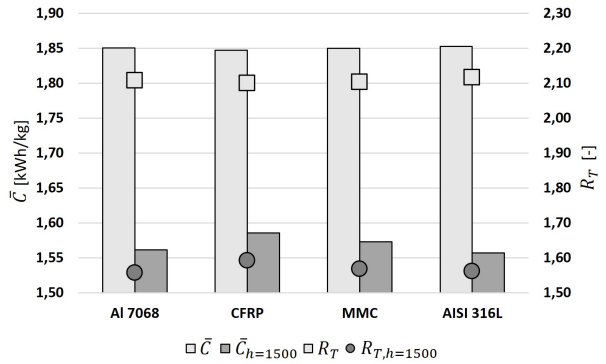


Figure 9. Specific energy cost for different wall materials with two internal convection cases

10.1016/j.ijhydene.2015.09.034. URL: http://dx.doi.org/10.1016/j.ijhydene.2015.09.034.

MSL (2023). Version 3.2.2. URL: https://github.com/modelica/ModelicaStandardLibrary.

OpenModelica (2023). Version 1.24.4. URL: https : / / openmodelica.org/ (visited on 2024-11-23).

Patil, Vikram C., Pinaki Acharya, and Paul I. Ro (2019). "Experimental investigation of heat transfer in liquid piston compressor". In: *Applied Thermal Engineering* 146.June 2018, pp. 169–179. ISSN: 13594311. DOI: 10 . 1016 / j . applthermaleng.2018.09.121. URL: https://doi.org/10.1016/j.applthermaleng.2018.09.121.

Saadat, M., P.Y. Li, and T.W. Simon (2012). "Optimal trajectories for a liquid piston compressor/expander in a Compressed Air Energy Storage system with consideration of heat transfer and friction". In: *2012 American Control Conference (ACC)*, pp. 1800–1805. DOI: 10.1109/ACC.2012.6315616.

Sapin, Paul et al. (2018). "Dynamic modelling of water-droplet spray injection in reciprocating-piston compressors". In: *ECOS 2018 - Proceedings of the 31st International Conference on Efficiency, Cost, Optimization, Simulation and Environmental Impact of Energy Systems* June.

Şeren, Erdoğan Mert and Mutlu İpek (2023). "Dynamic Modeling and Experimental Validation of Dishwasher with Heat Pump System". In: *Proceedings of the 15th International Modelica Conference 2023, Aachen, October 9-11* 204, pp. 347–356. DOI: 10.3384/ecp204247.

Solai, Elie, Mathieu Specklin, and Michaël Deligant (2025). "Assessment of liquid piston compression efficiency through uncertainty analysis based on experimental measurements and 0D system modeling". In: *Energy Conversion and Management* 326.December 2024, p. 119431. ISSN: 01968904. DOI: 10.1016/j.enconman.2024.119431. URL: https://doi.org/10.1016/j.enconman.2024.119431.

Solazzi, Luigi and Marco Vaccari (2022). "Reliability design of a pressure vessel made of composite materials". In: *Composite Structures* 279, p. 114726.

Specklin, Mathieu et al. (2022). "Numerical study of a liquid-piston compressor system for hydrogen applications". In: *Applied Thermal Engineering* 216.July, p. 118946. ISSN: 13594311. DOI: 10.1016/j.applthermaleng.2022.118946. URL: https://doi.org/10.1016/j.applthermaleng.2022.118946.

Srivatsa, Anirudh and Perry Y. Li (2018). "How moisture content affects the performance of a liquid piston air compressor/expander". In: *Journal of Energy Storage* 18.August 2017, pp. 121–132. ISSN: 2352152X. DOI: 10.1016/j.est.2018.04.017. URL: https://doi.org/10.1016/j.est.2018.04.017.

Van de Ven, James D. and Perry Y. Li (2009). "Liquid piston gas compression". In: *Applied Energy* 86.10, pp. 2183–2191. ISSN: 03062619. DOI: 10.1016/j.apenergy.2008.12.001. URL: http://dx.doi.org/10.1016/j.apenergy.2008.12.001.

Vullo, Vincenzo et al. (2014). "Circular cylinders and pressure vessels". In: *Stress Analysis and Design; Springer: Berlin/Heidelberg, Germany*.

Wang, Ying et al. (2018). "Research on the design of hydrogen supply system of 70 MPa hydrogen storage cylinder for vehicles". In: *International Journal of Hydrogen Energy* 43.41, pp. 19189–19195. ISSN: 03603199. DOI: 10.1016/j.ijhydene.2018.08.138. URL: https://doi.org/10.1016/j.ijhydene.2018.08.138.

Wieberdink, Jacob et al. (2018). "Effects of porous media insert on the efficiency and power density of a high pressure (210bar) liquid piston air compressor/expander – An experimental study". In: *Applied Energy* 212.August 2017, pp. 1025–1037. ISSN: 03062619. DOI: 10.1016/j.apenergy.2017.12.093. URL: https://doi.org/10.1016/j.apenergy.2017.12.093.

Rumoca: Towards a Translator from Modelica to Algebraic Modeling Languages

Micah K. Condie[1] Abigaile Woodbury[1] James M. Goppert[1] Joel A.E. Andersson[2]

[1]School of Aeronautics and Astronautics, Purdue University, United States,
{condiem,awoodbu,jgoppert}@purdue.edu
[2]Freelance software developer and consultant, United States & Finland, joel@jaeandersson.com

Abstract

We present Rumoca, a translator written in Rust that forms the basis of a symbolic toolchain, automatically converting Modelica models into a variety of target algebraic modeling languages. Rumoca is demonstrated on three models and translated into two different algebraic representations: CasADi and SymPy. Designed for generalizability, Rumoca has the potential to accommodate increasingly complex Modelica models and additional target languages.

Keywords: Modelica, symbolic computation, algebraic modeling, cyber-physical systems, Model Translation, CasADi, SymPy

1 Introduction

Cyber-physical systems (CPSs) are designed using a variety of modeling languages and tools, each with its own advantages (Amrani et al. 2021). However, interoperability between these tools remains limited. Porting models across platforms often requires manual conversion or unverifiable code generation techniques, such as those based on large language models (LLMs), which is both time-consuming and error-prone (Liu et al. 2020). To address this, we present Rumoca, a translator written in Rust that converts Modelica models into various algebraic modeling backends, enabling symbolic manipulation, automatic differentiation, and optimization (Jakob Andersson et al. 2019).

The long-term vision for Rumoca is to provide a flexible, extensible compiler infrastructure for hybrid systems modeling and analysis. It targets full coverage of a flattened Modelica representation—stripped of class hierarchies as defined later in this paper—and currently supports SymPy and CasADi backends. By exposing a lightweight abstract syntax tree (AST) and a modular templating engine, Rumoca makes it easy to develop additional code generation backends.

1.1 Modeling and Compiler Languages

Modelica was selected as the source language due to its concise semantics for modeling CPSs and its precise mapping to differential-algebraic equations (DAE) as defined in the language standard (Henriksson et al. 2011). These characteristics make it more verifiable than general-purpose languages like Python and C++, while availability of graphical editors and a large user community further support its adoption.

Rumoca is implemented in Rust, which offers significant benefits for compiler construction. The language provides strong static typing and memory safety without garbage collection, helping to eliminate entire classes of bugs while maintaining runtime performance. Modern features like pattern matching, algebraic data types, and expressive traits support the creation of robust and maintainable translation infrastructure (Klabnik and Nichols 2019; Matsakis and Klock 2014; Jung et al. 2018).

1.2 Target Languages

In designing Rumoca, CasADi and SymPy were selected as the primary languages for translated Modelica models. These languages were chosen because of their strengths in symbolic and numerical computation and optimization.

CasADi is a symbolic framework for algorithmic differentiation and numeric optimization, particularly well-suited for control and estimation in dynamic systems (Jakob Andersson et al. 2019). It represents expressions as directed acyclic graphs (DAGs), which allows for efficient symbolic manipulation and code generation. This structured representation offers significant advantages for optimization tasks compared to black-box models, such as those commonly encapsulated in Functional Mock-up Units (FMUs). CasADi is widely used in optimal control, parameter estimation, and nonlinear programming, especially in embedded applications where runtime performance and gradient computation are critical. By translating Modelica models into CasADi expressions, Rumoca enables users to perform gradient-based optimization, sensitivity analysis, and simulation leveraging high-performance solvers.

A key advantage of using CasADi is its close integration with numerical solvers like IPOPT (Biegler et al. 2009) and C code generation. This makes it highly suitable for deploying optimized controllers or estimators on embedded platforms. A disadvantage of CasADi is its lack of object-oriented features, which makes it difficult to scale to larger, more sophisticated models. However, since object-oriented design is a strength of Modelica, CasADi

is a strong candidate for serving as the backend of the symbolic toolchain enabled by Rumoca.

SymPy, in contrast, is a general-purpose computer algebra system written entirely in Python (Meurer et al. 2017). It supports a broad range of symbolic operations including simplification, equation solving, symbolic differentiation and integration, and matrix algebra. SymPy is especially useful for verifying model structure, performing symbolic manipulations, and producing readable mathematical expressions. These capabilities make SymPy an ideal target when the goal is to inspect or manipulate equations algebraically.

Although SymPy lacks the high-performance optimization and solver integration features of CasADi, it excels in educational, research, and verification settings. SymPy's strength lies in its ability to express mathematical models symbolically and perform analytical transformations, including the computation of Jacobians, Laplace transforms, or simplification of symbolic DAE systems. Its integration with Jupyter and Python's scientific stack makes it highly accessible and extendable. As an immediate application of the SymPy backend, we intend to leverage these computational features to construct reachability proofs.

1.3 Related Work

Numerous tools exist for compiling or translating Modelica models, each with distinct design goals. Rumoca distinguishes itself by acting as a *translator*–focused on high-level symbolic representation and model interoperability–rather than as a compiler generating low-level machine code.

The most mature open-source option is OpenModelica, which supports the Modelica Standard Library and offers a graphical user interface via OMEdit ((OSMC) 2025). While well-suited for simulation and code generation, OpenModelica does not currently support symbolic backends such as CasADi or SymPy. Since the OpenModelica compiler does support multi-code generation backends, it is possible it could be extended to these languages. Rumoca, however, offers unique advantages by being implemented in Rust, having algebraic modeling languages in mind.

Marco is a high-performance Modelica compiler written in C++ and based on LLVM (Agosta et al. 2023). It targets efficient simulation of large-scale models, prioritizing speed and robustness. However, Marco is focused on generating C code and does not offer translation into symbolic frameworks, limiting its utility for applications involving algebraic transformation or formal analysis.

In constrast to traditional Modelica compilers that focus on simulation and code generation, Rumoca is designed as a translator–targeting symbolic and algebraic frameworks. Its purpose is not to compete on execution speed or complete language coverage, but to facilitate workflows involving optimization, model verification, and hybrid system analysis. We envision a workflow that utilizes OpenModelica or Marco, by using these compilers to convert Modelica models to an intermediate representation that could be handled by Rumoca. The clear candidate would be the ongoing effort to define a BaseModelica language specification, which is a simplified subset of Modelica.

While Rumoca may not be able to handle a complex model directly, the model could first pass through a compiler such as Marco or OpenModelica, removing its object-oriented structure, and produce the model in BaseModelica. The BaseModelica version could then be processed by Rumoca. This proposed workflow is shown in Figure 1.

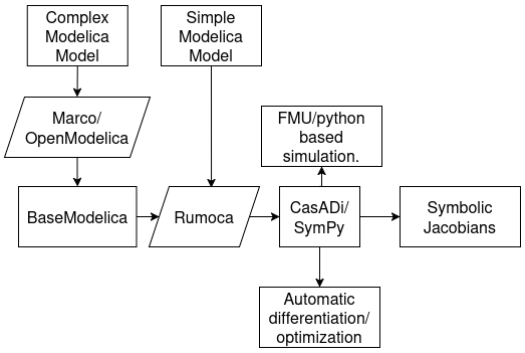


Figure 1. Proposed workflow utilizing compilers for generation of BaseModelica. Rumoca is a translator that takes simple Modelica models to target algebraic modeling languages. Complex models could first be passed through a more extensive compiler, removing the object oriented structure, before being processed by Rumoca.

Another notable effort is Pymoca, an open-source translator written in Python and based on ANTLR. It shares similar goals with Rumoca and provides a Pythonic interface for model transformation (Contributors 2025). Pymoca benefits from accessibility and ease of use, especially for Python developers. However, it suffers from significant performance limitations due to Python's speed and the use of un-typed ASTs. The lack of strong typing and the slow parsing pipeline–where the source is first converted into a parse tree before AST generation–makes Pymoca less suitable for handling large-scale models efficiently.

1.4 Overview

The remainder of the paper is structured as follows. Section 2 explains the details of Rumoca, including modifications to the Parol parser, structure of the AST, and use of Jinja2 templates for the generator code. Section 3 describes some changes that were made to the CasADi library to more easily accommodate the Modelica structure. Section 4 demonstrates Rumoca on three example systems, converting them from Modelica to Sympy and CasADi. Section 5 provides a summary and direction for future work on the Rumoca translator.

DOI 10.3384/ecp2181009

2 Rumoca Details

The compiler pipeline can be understood in terms of four components. First, the automatically generated parse tree is overwritten to a customized IR format. Second, this AST is flattened, removing class hierarchies. Third, the flattened IR is lowered into a DAE representation, consistent with the semantics defined by the Modelica specification. Finally, the DAE system is rendered into the target language using a template engine.

2.1 Parser

For parsing, Rumoca is using the Parol parser (Singer 2025). Parol is a Rust based parser generator, similar to ANTLR. Given an EBNF grammar file, Parol can automatically generate a parser. While we evaluated other parser generators such as ANTLR and LALRPOP, we found that Parol was best suited for the project. ANTLR handles EBNF grammar well, but does not have a native Rust implementation, and lacks the capability to significantly modify the automatically generated parser code. ANTLR does however support (LR*) grammar with arbitrary look-ahead which simplifies grammar handling.

The LALRPOP parser generator, which handles LR(1) grammar, we found to be limiting, and the compilation times for the compiler slowed our development cycle. LALRPOP did provide an easy mechanism for modifying the generated parser tree at runtime so that a second pass over the parser tree was not generated to obtain an abstract syntax tree.

Finally, Parol was selected as it had the advantage of supporting EBNF grammar and allowing for customization of the parse tree at parse time. When customization of the parse tree is necessary to obtain a desirable abstract syntax tree to be used as an internal representation, it is useful to selectively re-implement the parse tree generation for specific language elements. The automatically generated parsing functions may then be leveraged when customization is not necessary. Parol has a fast compile time, similar to ANTLR, and is a LL(k) parser. LL(k) means that the EBNF grammar is analyze for the largest look-ahead necessary. We were able to implement the Modelica language specification with a look-ahead of 3 tokens (k = 3). In order to simplify the implementation of the Modelica language parser, we requested several features to be added to Parol, and they were quickly implemented.

2.2 Language Coverage

While the Parol parser automatically generates a parse tree for the entire Modelica language grammar, the steps converting it to the intermediate representations require significant customization. For this reason, Rumoca only supports a portion of the Modelica language. Independently, there is an ongoing effort to define a BaseModelica language specification, which is a simplified subset of Modelica. It is our goal to have Rumoca work for the entirety of the BaseModelica features. Table 1 shows a preliminary draft of the features that may be included in BaseModelica, and which are currently supported by Rumoca. The features listed were taken from the BaseModelica meeting notes (Modelica Association 2023). We envision that compilers such as OMC or Marco will compile to BaseModelica, so that by chaining the tools any Modelica model could be handled by Rumoca.

Table 1. Rumoca support for selected Modelica features

Modelica Feature	*Supported*
Scalar types (e.g., `Real`, `Integer`)	Yes
Record and enum types	No
Arrays (decl. & indexing)	Partial
Component decl.	Yes
Equations and algorithms	Yes
Source location metadata	No
Documentation strings	Yes
Vendor-specific annotations	No
All variabilities (no param eval)	Yes
Params treated as const	Yes
Preserve values of inlined constants	Yes
Relaxed record field access	No
Relaxed array subscripting	Partial
Naming of intermediate vars	No
Alias expressions (ref & sign)	No
Used function declarations	Yes

In addition to the features listed, we have added support for simple class structures. This is because we intend to use Rumoca for hierarchical reachability proofs which require nested components.

2.3 Generator Code

Jinja2, a widely used text templating engine for Python, is employed to construct the generator code. With millions of downloads per month and integration into major frameworks such as Flask and Ansible, Jinja2 benefits from a large and active user base (Index 2024; Projects 2024; Red Hat 2024). In Rumoca, each Jinja template defines how the symbolic expressions described in the AST are rendered into source code. Due to the common DAE structure of the AST, templates can be created and modified for specific target languages with relative ease. This gives Rumoca the advantage of being extendable to a variety of backends with minimal effort and expertise. This allows for end-users to create custom backends. Examples and documentation for writing custom templates are available online.[1]

An example of how a template defines the handling of model parameters is shown below. Listing 1 presents a portion of the template for rendering the CasADi repre-

[1] `https://github.com/CogniPilot/rumoca/blob/main/tests/templates`

sentation, while Listing 2 displays the corresponding rendered code.

Listing 1. Jinja2 template for parameter handling

```
{%- elif var == 'p' %}
    # Declare {{ var }}
    {% for c in dae[var] -%}
    {{ c.name }} = dae.add('{{ c.name }}',
        'parameter', 'tunable', dict(start
        = {{ render_expression(c.start) }})
        )
    {% endfor -%}
```

Listing 2. Rendered code for parameter declarations

```
# Declare p
e = dae.add('e', 'parameter', 'tunable',
    dict(start = 0.8))
h0 = dae.add('h0', 'parameter', 'tunable',
    dict(start = 1.0))
```

3 CasADi Additions

In this work we leverage and extend CasADi's DaeBuilder class (Joel Andersson 2024), which can be used for both import of Modelica in both symbolic form and in the form of exported FMUs adhering to FMI 2.0 and 3.0. In the recent CasADi 3.7 release, the DaeBuilder class was refactored to use a representation of model variables that closely resembles the representation of model variables in the FMI standard. The same release also saw a new refactored high-level interface for symbolically defining model equations, now including model equations with event dynamics, compatible with the recently added support for analytic derivative calculations for dynamic systems with events (Joel Andersson and Goppert 2025). Another important update is an extension of CasADi's support for FMI export, which can now be used for systems with events. We refer to the updated CasADi user guide for details on this support.

4 Examples

This section demonstrates the use of Rumoca on three representative examples: a simple bouncing ball, a mass-spring-damper system, and a more complex quadrotor model. For each case, Rumoca takes the corresponding Modelica model and translates it into either a SymPy or CasADi representation, depending on the selected backend template. These examples illustrate the class of systems that Rumoca currently supports, ranging from basic hybrid dynamics to nonlinear multibody systems.

In addition to translation, further analyses are performed to highlight some of the benefits of the target symbolic languages. Specifically, an optimal disturbance attack is computed for the mass-spring-damper model, leveraging CasADi's directed acyclic graph (DAG) structure and automatic differentiation. For the quadrotor, the Jacobian matrix is extracted using SymPy's symbolic differentiation capabilities, which enable exact analytical expressions for sensitivity analysis and control design.

Rumoca is accessed through a command-line interface. Listing 3 shows the help menu output. As shown, Rumoca requires two inputs: a Modelica model and a code generation template specifying the target language. Each example uses a distinct Modelica file and a corresponding template—either for SymPy or CasADi—to produce the desired output.

Listing 3. Rumoca Help Menu

```
Rumoca Modelica Translator

Usage: rumoca [OPTIONS] --template-file <
    TEMPLATE_FILE> --model-file <MODEL_FILE
    >

Options:
  -t, --template-file <TEMPLATE_FILE>  The
      template
  -m, --model-file <MODEL_FILE>        The
      model file to compile
  -v, --verbose
      Verbose output
  -h, --help
      Print help
  -V, --version
      Print version
```

4.1 Bouncing Ball

As a first example, consider the classic bouncing ball model. Listing 4 shows this in Modelica.

Listing 4. Bouncing Ball model in Modelica

```
model BouncingBall "The 'classic' bouncing
    ball model"
  parameter Real e=0.8 "Coefficient of
      restitution";
  parameter Real h0=1.0 "Initial height";
  Real h = 5.0 "Height";
  Real v "Velocity";
  Real z;
equation
  z = 2*h + v;
  der(h) = v;
  der(v) = -9.81;
  when h<0 then
    reinit(v, -e*pre(v));
  end when;
end BouncingBall;
```

The bouncing ball model above is processed by Rumoca using the CasADi template, which generates a corresponding Python script. Listing 4.1 shows the command used to run this transformation via the command-line interface.

```
rumoca BouncingBall.mo -t casadi.jinja >
    bouncing_ball_ca.py
```

This generated script, here called "bouncing_ball_ca.py", contains a class with several methods. A portion of this code is shown in Listing 5.

Listing 5. A portion of the casadi code generated by rumoca (bouncing_ball_ca.py)

DOI 10.3384/ecp2181009

```
def c0():
    return (h < 0.0)

# ============================================
# Define reset functions: fr
def c0_fr():
    return dae.reinit("v", -((e * pre_v)))
dae.when(c0(), [c0_fr()])
# ============================================
# Declare x_dot
der_h = dae.der(h)
der_v = dae.der(v)
# ============================================
```

The example in Listing 6 demonstrates how to simulate the bouncing ball using the generated CasADi class, the result of which, is Figure 2 .

Listing 6. Simulation of the bouncing ball model using CasADi backend

```
import bouncing_ball_ca
import matplotlib.pyplot as plt

model = bouncing_ball_ca.Model("
    bouncing_ball")
tgrid, res = model.simulate(t0=0, tf=8, dt
    =0.01)

plt.plot(tgrid, res['xf'].T, label=model.
    dae.x())
plt.grid()
plt.legend()
plt.show()
```

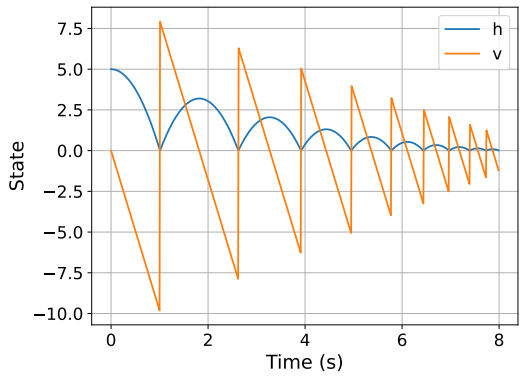


Figure 2. Height (m) and velocity (m/s) of a bouncing ball as a function of time. The simulation was done using the automatically generated CasADi script.

Consider now the example of using Rumoca on the bouncing ball model with the SymPy template. Here, the generated code is a script "bouncing_ball_sympy.py." As with the CasADi generated script, this code contains a class with several methods. A portion of this code is shown in Listing 7.

Listing 7. A portion of the SymPy code generated by rumoca ("bouncing_ball_sympy.py")

```
# Define Reset Functions: fr
def __fr_c0(x):
    pre_h, pre_v = self.x
    h, v = self.x
    v = -((e * pre_v))
    return [
        h,
        v]
self.fr_c0 = sympy.lambdify([self.x, self.p
    ], __fr_c0(self.x))

#
    ============================================

# Define Condition Update Function: fc
self.fc = sympy.Tuple(*[
    (h < 0.0)])
self.f_c = sympy.lambdify(
    args=[self.time, self.x],
    expr=self.fc,
    modules=['numpy'])
```

Listing 8 demonstrates running the bouncing ball simulation with the SymPy script, adjusting the initial height and velocity as well as the coefficient of restitution parameter. The corresponding plot can be seen in Figure 3.

Listing 8. Simulation of the bouncing ball model using SymPy backend

```
import bouncing_ball_sympy
import matplotlib.pyplot as plt

model = bouncing_ball_sympy.Model()
model.solve()
model.x0['h'] = 10
model.x0['v'] = 5
model.p0['e'] = 0.7
res = model.simulate(t0=0, tf=8, dt=0.01)

plt.plot(res['t'], res['x'].T, label=model.
    x0.keys())
plt.legend()
plt.grid()
```

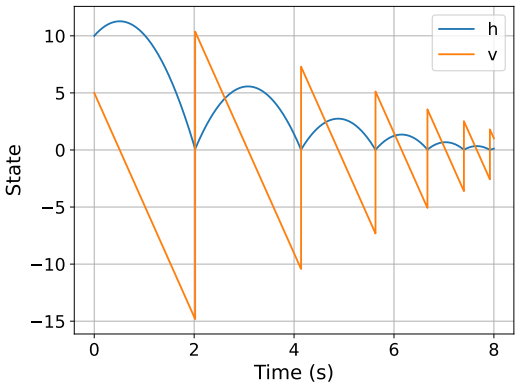


Figure 3. Height (m) and velocity (m/s) of a bouncing ball as a function of time. The simulation was done using the automatically generated SymPy script.

4.2 Mass-Spring-Damper

As an example of leveraging CasADi optimization, consider the simple mass spring damper system in Modelica

with a disturbance force shown in Listing 9.

Listing 9. Mass-spring-damper system

```
model msd "Mass spring damper"
  parameter Real k=1.0 "Coefficient of
      spring";
  parameter Real c=1.0 "Coefficient of
      damper";
  parameter Real m=1.0 "mass";
  Real x = 0.0 "Position";
  Real v "Velocity";
  input Real u "Disturbance";
equation
  der(x) = v;
  der(v) = -(k/m)*x - (c/m)*v - (1/m)*u;
end msd;
```

Once the model is processed by Rumoca using the CasADi template, the system can be optimized using familiar CasADi features. Consider the problem of optimizing a bounded disturbance for the MSD system such that it causes the mass to deviate as far as possible from equilibrium. This problem is solved using CasADi optimization tools, and the result is shown in Figure 4. This was done with the system only ever written in Modelica.

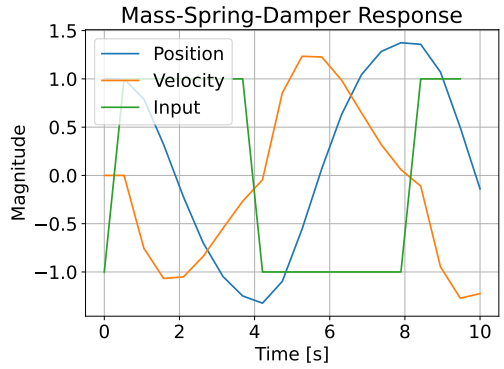


Figure 4. The optimized bounded control input to maximize disturbance from equilibrium. This was found using CasADi tools, having only written the system in Modelica.

4.3 Quadrotor

Rumoca is also capable of handling more complex models, including some with a hierarchical structure. The quadrotor model shown in Listing 10, extends a rigid body model and makes use of a separate motor model. Automatic generation of the corresponding CasADi and SymPy code are shown and demonstrate handling control inputs and using Sympy to linearize the model.

Listing 10. Quadrotor model definition in Modelica

```
model Quadrotor
    extends RigidBody6DOF;
    parameter Real l = 1.0;
    parameter Real mix_a = 1;
    parameter Real mix_e = 1;
    parameter Real mix_r = 10;
    parameter Real mix_t = 32.0;
    Motor m_1, m_2, m_3, m_4;
    input Real a "aileron";
    input Real e "elevator";
    input Real r "rudder";
    input Real t "throttle";
equation
    // body forces
    F_x = -(m*g)*sin(theta);
    F_y = (m*g)*sin(phi)*cos(theta);
    F_z = (m*g)*cos(phi)*cos(theta) -
          (m_1.thrust + m_2.thrust + m_3.
              thrust + m_4.thrust);

    // body moments
    M_x = l*(-m_1.thrust + m_2.thrust - m_3
        .thrust + m_4.thrust);
    M_y = l*(-m_1.thrust + m_2.thrust + m_3
        .thrust - m_4.thrust);
    M_z = m_1.moment + m_2.moment - m_3.
        moment - m_4.moment;

    // motor equations
    m_1.omega_ref = t*mix_t - a*mix_a + e*
        mix_e + r*mix_r;
    m_2.omega_ref = t*mix_t + a*mix_a - e*
        mix_e + r*mix_r;
    m_3.omega_ref = t*mix_t - a*mix_a - e*
        mix_e - r*mix_r;
    m_4.omega_ref = t*mix_t + a*mix_a + e*
        mix_e - r*mix_r;
end Quadrotor;
```

4.3.1 CasADi Quadrotor Simulation

Upon running Rumoca for the quadrotor model using the CasADi template, the "quadrotor_casadi.py" python script is generated. Listing 11 demonstrates how the control input can be adjusted and the quadrotor simulated. In this case, the throttle is set to a constant value. The resulting position of the quadrotor upon running the CasADi simulation is shown in Figure 5.

Listing 11. CasADi simulation setup for the quadrotor model

```
model = quadrotor_casadi.Model("quad")
states = model.dae.x()
params = model.dae.p()

x0 = model.dae.start(states)
param_vals = model.dae.start(params)

# Set control input: aileron, elevator,
    rudder, throttle
inputs = np.array([0, 0, 0, 0.5])

# Set initial height of quadrotor to 10 m.
x0[states.index('h')] = 10
tgrid, res = model.simulate(t0=0, tf=10, dt
    =0.01, f_u=inputs, x0=x0)
```

4.3.2 SymPy Quadrotor Linearization

Rumoca's SymPy output enables the generation of symbolic expressions for the Modelica model. Eq, displays a portion of the 16×16 linearized state matrix for the quadrotor automatically generated by Sympy from the

DOI
10.3384/ecp2181009

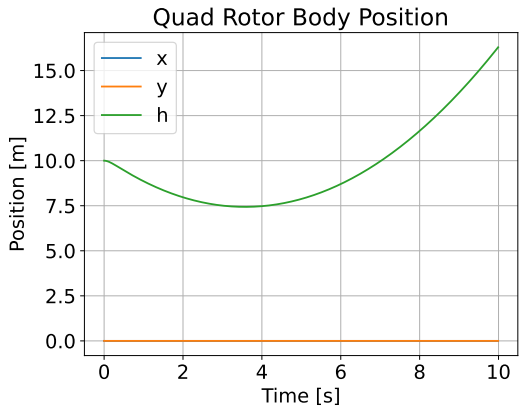


Figure 5. Quadrotor position from CasADi simulation.

Modelica model, illustrating the power of using SymPy as a back-end.

$$\begin{array}{ccc} \frac{J_x J_{xz} Q - J_{xz} J_y Q + J_{xz} J_z Q}{\Lambda} & \frac{J_x J_{xz} P - J_{xz}^2 R - J_{xz} J_y P + J_{xz} J_z P + J_y J_z R - J_z^2 R}{\Lambda} & \frac{-J_{xz}^2 Q + J_y J_z Q - J_z^2 Q}{\Lambda} \\ \frac{-J_x R - 2J_{xz} P + J_z R}{J_y} & 0 & \frac{-J_x P + 2J_{xz} R + J_z P}{J_y} \\ \frac{J_x^2 Q - J_x J_y Q + J_{xz}^2 Q}{\Lambda} & \frac{J_x^2 P - J_x J_{xz} R - J_x J_y P + J_{xz}^2 P + J_{xz} J_y R - J_{xz} J_z R}{\Lambda} & \frac{-J_x J_{xz} Q + J_{xz} J_y Q - J_{xz} J_z Q}{\Lambda} \end{array} \tag{1}$$

Linearizing the system in SymPy, the root locus and Bode plots can be quickly computed in Python. These are shown in Figure 6 and 7 respectively for the transfer function relating the motor throttle command to the altitude of the quadrotor.

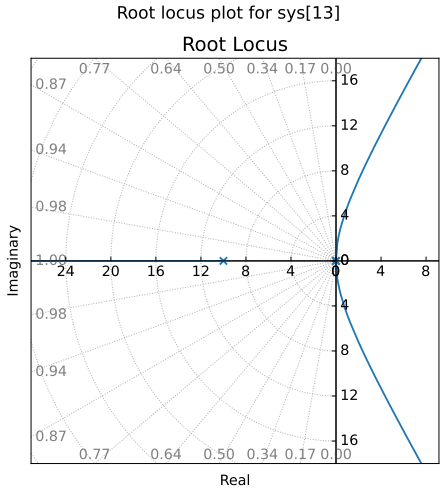


Figure 6. Root locus plot for linearized quadrotor.

5 Summary and Future Work

This paper introduces Rumoca, a Rust-based translator that automates the conversion of Modelica models into a variety of target algebraic modeling languages. Demonstrated through the translation of three models into two algebraic representations—CasADi and SymPy—Rumoca showcases its ability to bridge Modelica and common

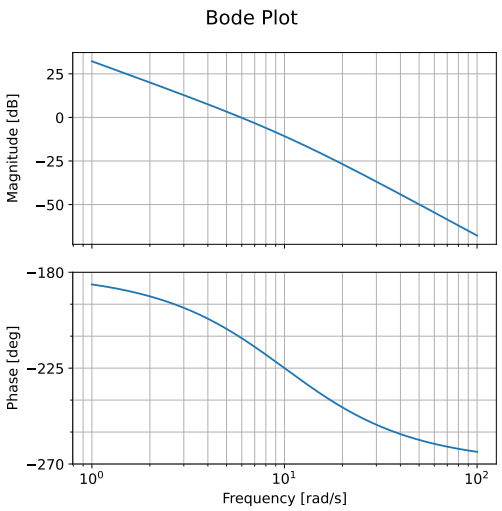


Figure 7. Bode plot for linearized quadrotor.

computational tools. While still in active development, future enhancements to Rumoca aim to expand its capabilities to handle a broader range of Modelica features. Additionally, future work may include the creation of Jinja templates to integrate with other back-ends, such as JAX for machine learning applications, Pycollimator for an intuitive graphical user interface, and Gazebo for simulations.

Acknowledgements

This material is based on research sponsored by DARPA under agreement number FA8750-24-2-0500. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

We would like to thank Jörg Singer for his assistance extending the Parol parser generator to support our work.

References

(OSMC), Open Source Modelica Consortium (2025). *OpenModelica - Open Source Modelica Environment*. https://www.openmodelica.org. Accessed: 2025-04-30.

Agosta, Giovanni et al. (2023). "MARCO: An Experimental High-Performance Compiler for Large-Scale Modelica Models". In: *Proceedings of the 15th International Modelica Conference*. Modelica Association, pp. 1–10. DOI: 10.3384/ecp20413. URL: https://ecp.ep.liu.se/index.php/modelica/article/view/909.

Amrani, Moussa et al. (2021). "Multi-paradigm modelling for cyber–physical systems: A descriptive framework". In: *Software and Systems Modeling* 20.3, pp. 611–639. DOI: 10.1007/s10270-021-00876-z.

Andersson, Jakob et al. (2019). "CasADi: A software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation* 11.1, pp. 1–36. DOI: 10.1007/s12532-018-0141-4.

Andersson, Joel (2024-03). "Import and Export of Functional Mockup Units in CasADi". In: pp. 321–326. DOI: 10.3384/ecp204321.

Andersson, Joel and James Goppert (2025-01). "Event Support for Simulation and Sensitivity Analysis in CasADi for use with Modelica and FMI". In: pp. 99–108. DOI: 10.3384/ecp20799.

Biegler, Lorenz T. et al. (2009). *Large-Scale Nonlinear Optimization*. Vol. 83. Lecture Notes in Computational Science and Engineering. Springer. ISBN: 978-3-540-88321-3. DOI: 10.1007/978-3-540-88322-0. URL: https://doi.org/10.1007/978-3-540-88322-0.

Contributors, Pymoca (2025). *Pymoca: Modelica Compiler in Python*. https://github.com/pymoca/pymoca. Accessed: 2025-04-30.

Henriksson, Carl Johan et al. (2011). "Modelica: A unified object-oriented language for modeling and simulation of complex systems". In: *Simulation Modelling Practice and Theory* 19.3, pp. 697–718. DOI: 10.1016/j.simpat.2010.10.005.

Index, Python Package (2024). *Jinja2: PyPI Package Statistics*. https://pypistats.org/packages/jinja2.

Jung, Ralf et al. (2018). "RustBelt: Securing the Foundations of the Rust Programming Language". In: *Proceedings of the ACM on Programming Languages* 2.POPL, p. 66. DOI: 10.1145/3158154.

Klabnik, Steve and Carol Nichols (2019). *The Rust Programming Language*. No Starch Press. URL: https://doc.rust-lang.org/book/.

Liu, Bo et al. (2020). "A survey of model-driven techniques and tools for cyber-physical systems". In: *Frontiers of Information Technology & Electronic Engineering* 21.11, pp. 1567–1590. DOI: 10.1631/FITEE.2000108.

Matsakis, Nicholas D. and Felix S. Klock (2014). "The Rust Language". In: *ACM SIGAda Ada Letters* 34.3, pp. 103–104. DOI: 10.1145/2660193.2660199.

Meurer, Aaron et al. (2017). "SymPy: symbolic computing in Python". In: *PeerJ Computer Science* 3, e103.

Modelica Association (2023). *Base Modelica Requirements — MCP-0031*. Accessed: 2025-07-24. URL: https://github.com/modelica/ModelicaSpecification/blob/MCP/0031/RationaleMCP/0031/Base-Modelica-requirements.md.

Projects, Pallets (2024). *Flask Documentation*. https://flask.palletsprojects.com/.

Red Hat, Inc. (2024). *Ansible Documentation*. https://docs.ansible.com/.

Singer, Jörg (2025). *Introduction to the Parol Parser Generator*. https://jsinger67.github.io/Introduction.html. Accessed: 2025-05-02.

Safe and Efficient Control of a Brayton Cycle Heat Pump Using Reinforcement Learning

A. Phong Tran[1] Fatma Cansu Yücel[1]

[1]German Aerospace Center (DLR), Institute for Low-Carbon Industrial Processes, Germany,
{Phong.Tran,Fatma.Yuecel}@dlr.de

Abstract

Decarbonizing industrial process heating will increasingly depend on high-temperature heat pumps. In particular, Brayton cycle heat pumps, which can reach temperatures above 250 °C, are viewed as a promising technology. However, ensuring safe operation and optimal control remains challenging. This study presents an experimentally validated dynamic model of a Brayton cycle heat pump, a system with multiple control inputs for regulating its thermal output. Using this model as a training environment, several control concepts integrating Reinforcement Learning (RL) and traditional PI controllers were implemented to achieve desired heat supply at target temperatures. Domain randomization was employed to improve the controller robustness against model uncertainties in preparation for deployment on the physical system. The results demonstrate that RL controllers can not only achieve the desired set-point temperature under varying loads while maintaining required safety margins, but also discovered a novel, more energy-efficient operational strategy.

Keywords: Brayton cycle heat pump, dynamic simulation, reinforcement learning, model-based control design

1 Introduction

Heating for industrial processes is predominantly fueled by fossil fuels, presenting a significant challenge for decarbonization efforts (IEA 2024). High-temperature heat pumps are crucial technologies for electrifying industrial process heating, and improving their control is essential for maximizing their operational efficiency and decarbonization impact.

1.1 High-temperature heat pumps

Recent advancements have significantly raised achievable supply temperatures of heat pump systems. The IEA HPT Annex 58 (Heat Pump Centre 2023) compiled heat pump demonstrators achieving temperatures of 115 °C to 280 °C. For the upper end of this range, particularly for sensible heating, gas compression heat pumps utilizing the Brayton cycle are emerging as a viable technology, with studies indicating technical feasibility up to 300 °C to 400 °C and economic potential in scenarios reaching 280 °C (Zühlsdorf et al. 2019). Oehler, Tran, and Stathopoulos (2022) note that while Brayton cycle heat pumps provide significant operational flexibility, their operation is constrained by compressor surge, thermal stresses, and natural frequencies, thus requiring a carefully designed control system.

1.2 Reinforcement learning

Reinforcement Learning (RL) is an area of machine learning focused on how agents learn to make optimal sequences of decisions through interaction with an environment. RL has proven useful in many areas, including games (Silver et al. 2018), robotic systems (Tang et al. 2025), and optimal energy dispatching (Di Cao et al. 2020). Model-free RL algorithms are particularly adaptable as they learn purely from environment interactions, but this often requires large amounts of data, highlighting the need for simulation. Modelica models have emerged as suitable simulators to train RL algorithms, as demonstrated by developments such as FMUGym (Wrede et al. 2024) and ModelicaGym (Lukianykhin and Bogodorova 2019), both of which utilize the Functional Mock-Up Interface (FMI). Although training in simulated environments offers a safe and efficient approach, these simulations inherently differ from their real-world counterparts. This discrepancy can cause policies trained in simulation to fail or underperform when deployed on the actual physical system. This challenge is widely recognized as the *reality gap* or the *Sim2Real* problem (Zhao, Queralta, and Westerlund 2020).

1.3 Other advanced control techniques

Optimica (Åkesson 2008), an extension of the Modelica language, allows users to formulate optimization problems alongside dynamic system models, which can then be processed by tools such as JModelica (Magnusson and Åkesson 2015) and the Optimica Compiler Toolkit (Modelon, Inc. 2024). These tools interface with the framework CasADi (Andersson et al. 2019), which translates the problem into a non-linear programming problem. The resulting NLP can then be solved by numerical solvers like Interior Point Optimizer (IPOPT). The described toolchain provides a mechanism for solving optimal control problems for systems modeled in Modelica. This capability is a prerequisite for Model Predictive Control (MPC).

1.4 Our contributions

This study utilizes an experimentally validated dynamic model of a Brayton cycle heat pump as an accurate training environment for RL. Three control concepts are comparatively analyzed, including a traditional approach with PI controllers, a pure RL controller and a combination of RL and PI controllers. To improve the potential for real-world transfer, the RL agents are trained using domain randomization. The results show that RL agents can learn the control tasks and discover novel and more energy-efficient operational strategies. The viability of training RL agents in a simulation environment is demonstrated by their control performance and ability to generalize across a wide range of system parameters.

2 The CoBra Heat Pump

This study investigates a closed-loop Brayton-cycle heat pump that uses air as its working fluid. The prototype under investigation, named "CoBra", was developed in Cottbus, Germany, and serves as a research platform for the design and operation of high-temperature heat pumps. Yücel et al. (2025) provide details on the system design and first commissioning tests of the plant.

2.1 System description

Figure 1 shows a flow schematic with the main components of the heat pump. The heat pump employs a two-stage compression system (A), using two radial compressors connected in series driven by an electric motor. After compression, the working fluid releases heat in the high-temperature heat exchanger (B). A three-way valve (C) controls the flow through the recuperator (G), which preheats the flow before compression using the temperature difference between the states 2 and 5. The turbine (D) expands the fluid, thereby recovering power via a generator and cooling the fluid further. A turbine bypass valve (E) operates in parallel to the turbine, allowing adjustment of the loop resistance to control the compressor operating point. After expansion, the cold fluid absorbs heat from a heat source or provides cooling in the low-temperature heat exchanger (F). Finally, the fluid passes through the recuperator (G) to be preheated before re-entering the compression system (A).

As investigated by Oehler, Gollasch, et al. (2021), the heat pump's fluid inventory (total working fluid mass) can be adjusted using a pair of valves. An inlet valve (H1) injects air from a high-pressure buffer tank to the low-pressure section to increase the fluid inventory, while a relief valve (H2) vents air from the high-pressure section to the ambient environment to decrease it. This fluid inventory control allows modulation of the heat pump's thermal output while maintaining the aerodynamic operating conditions of the turbomachinery. The demonstrated range for this control spans compressor inlet pressures (p_0) from 0.8 to 1.4 bar.

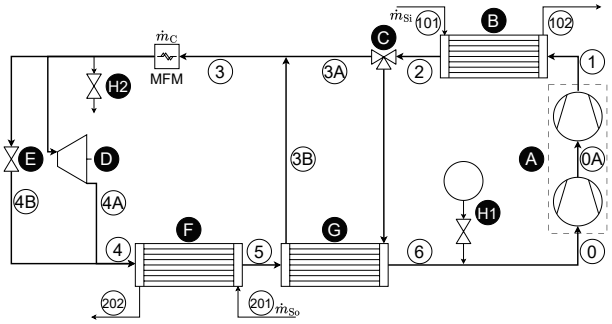


Figure 1. Flow schematic of the CoBra heat pump. (A) Two-stage compression system, (B) high-temperature heat exchanger, (C) three-way valve, (D) turbine, (E) turbine bypass valve, (F) low-temperature heat exchanger, (G) recuperator, (H1) buffer tank and air inlet valve, (H2) relief valve, MFM: Coriolis mass flow meter. Adapted from Tran and Stathopoulos (2025).

2.2 Dynamic model

A dynamic thermo-fluid model of CoBra heat pump was implemented in Modelica using the Dymola environment, as presented in previous work (Tran and Stathopoulos 2025), combining and extending models from the Modelica Standard Library (MSL), Buildings library (Wetter et al. 2014) and ThermoPower library (Casella and Leva 2006) to model the relevant thermodynamic processes and system dynamics. The model diagram is shown in Figure 2.

The libraries were selected due to their shared use of interfaces from the MSL, defined in `Modelica.Fluid` and `Modelica.Media` for fluid flow and properties, `Modelica.Thermal` for heat transfer, and `Modelica.Mechanics` for rotational mechanics. The specific contributions of each library to the overall model are as follows:

ThermoPower: The turbomachinery models (compressor and turbine) were implemented using models adapted from the ThermoPower library. These models assume steady-state behaviour without inherent mass or thermal dynamics, an assumption justified by the fast response time of the turbomachinery relative to other system components. Their behavior is defined by performance maps that relate pressure ratio, corrected mass flow, corrected shaft speed, and isentropic efficiency. The turbomachinery models also feature an `Modelica.Mechanics` flange, which enables a connection to the rotational components.

Buildings: The Buildings library was used to model the plant's piping, valves and fans. The volume and thermal inertia of the extensive piping network were represented using `MixingVolume` with dynamic energy and mass balances. Frictional losses in each pipe segment were modeled by two `PressureDrop` models in series, one with a quadratic ($\Delta p \sim \dot{m}^2$) and another with a linear ($\Delta p \sim \dot{m}$) pressure loss correlation. The three-way and turbine bypass valves are represented by the provided valve

DOI 10.3384/ecp2181017

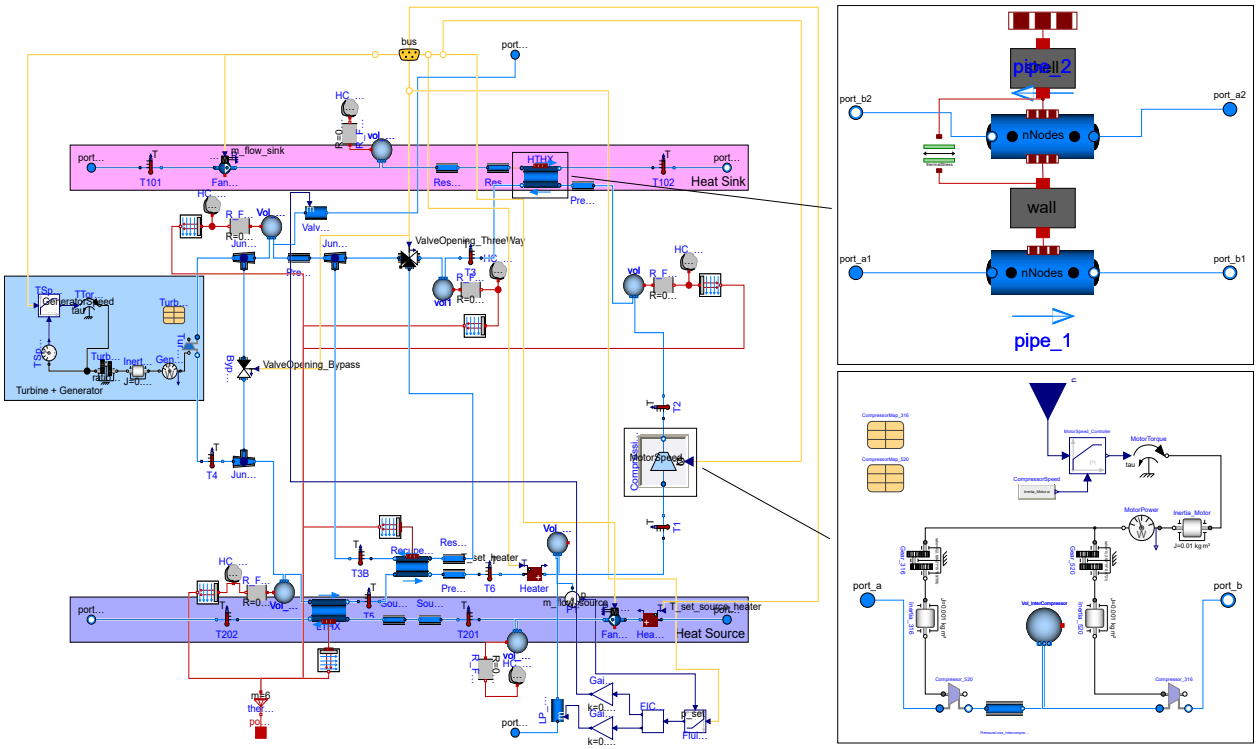


Figure 2. Diagram of the heat pump model in Dymola. Left: Overall heat pump. Top-right: Heat exchanger. Bottom-right: Compression system.

models with valve characteristics and sizings taken from manufacturer data. The fans providing air flow to the heat source and sink were represented by the library's flow-controlled mover models.

MSL: Beyond providing the interfaces, the MSL was employed to model mechanical components and heat exchangers. The rotational components, including the motor, generator, shafts and belt drives, were modeled using components from `Modelica.Mechanics`. The heat exchanger models were adapted from the `BasicHX` model and use a spatial discretization approach. In this method, the fluid flow paths and the solid metal structures (tubes and shell) are divided into multiple segments. For the purpose of this study, the baseline `BasicHX` model was extended to model several features of the real component: the heat capacity of the heat exchanger's outer shell, an additional heat port to model heat exchange with the ambient environment, and a model that calculates thermal stresses based on the temperature distribution along the tube and shell length.

2.3 Experimental validation

The methodology for calibrating the model against experimental data was presented in previous work (Tran and Stathopoulos 2025). Adjusting the compressor performance maps was a main focus, employing a transformation method aimed at minimizing prediction error while preserving the map's original shape, ensuring smoothness, and allowing for plausible extrapolation into unmeasured regions. Other model parameters, mainly heat transfer, heat loss and pressure loss coefficients, were tuned using numerical optimization to minimize deviations between simulation results and measurements.

For this study, a recalibration of the model was performed. This was necessary because new measurement data became available, recorded from a test setup that includes fully recuperated heat pump operation, a condition not covered in the previous work. An example test run from the new dataset is shown in Figure 3. In this test, following an initial ramp-up to nominal motor speed, a PI controller was applied to maintain the supply temperature (T_{102}) at 200 °C by adjusting compressor speed. Subsequently, the compressor inlet pressure (p_0) was varied stepwise (1.1 bar down to 0.8 bar in hourly intervals) using fluid inventory control. The test aimed to find the optimal operating point for the target supply temperature, indicated by the highest measured COP. The highest steady-state COP was measured at a pressure of 0.8 bar, which is the lower bound of investigated pressures. This suggests that the optimum might be at even lower pressures.

The predictions from the recalibrated model are compared to the measurement data shown in Figure 3. The resulting root mean squared errors (RMSE) and mean-normalized root mean squared errors (NRMSE) for the primary process variables are presented in Table 1. For the measured temperatures, pressures, and mass flow rate, the NRMSE values are in the range of 0.05 % to 1.95 %.

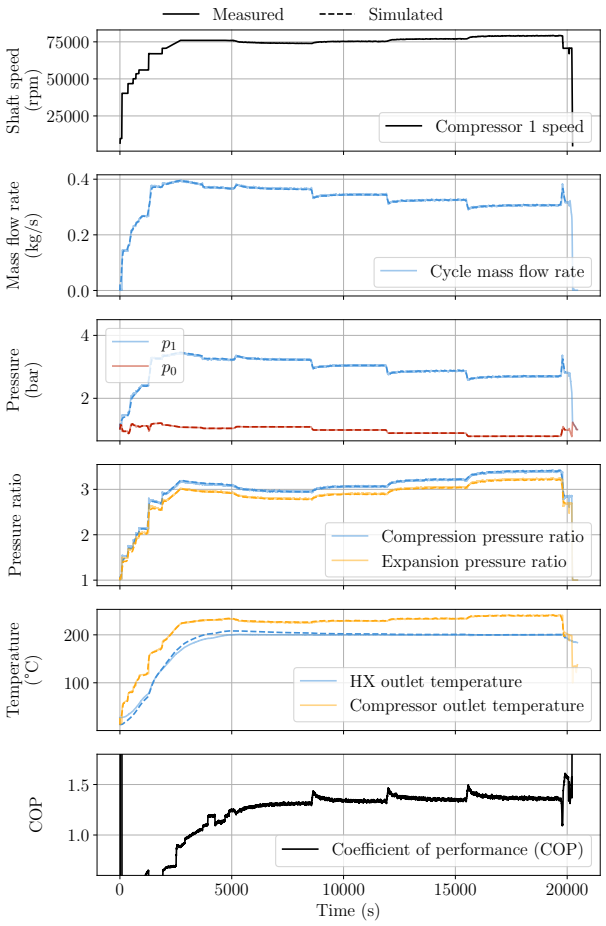


Figure 3. Comparison of measured (solid) and simulated (dashed) time series. The plot shows the alignment of experimental data and simulation results over time.

3 Heat Pump Control

The heat pump's target supply temperature (T_{102}), the return temperature from the process (T_{101}) and the heat sink mass flow rate ($\dot{m}_{\mathrm{Si}}$) are typically defined by process requirements. Consequently, a heat pump's control system is designed to track the set-point supply temperature while maintaining safe operating conditions. The CoBra heat pump features three main control variables that affect the supply temperature:

1. The *motor speed* (n_M) is the main manipulated variable for controlling the heat pump's supply temperature because it directly governs the compressor pressure ratio, mass flow rate and temperature lift.

2. The *three-way valve opening* (θ_{3W}) controls the amount of heat that is internally recovered by the recuperator. A value of $\theta_{3W} = 0$ represents full recuperation, whereas $\theta_{3W} = 1$ corresponds to fully non-recuperated operation.

3. Manipulation of the fluid inventory, indicated by the *compressor inlet pressure* (p_0), directly scales with the mass flow rate in the heat pump cycle, thereby providing control over its overall thermal output capacity.

Table 1. Root mean squared error (RMSE) and mean-normalized root mean squared error (NRMSE) of the main process variables.

Measurement signal		RMSE	NRMSE
Compressor outlet	T_1	1.64 K	0.33 %
HTHX outlet	T_2	3.89 K	1.19 %
Recuperator outlet (HP)	T_{3B}	0.72 K	0.25 %
Turbine inlet	T_3	0.56 K	0.19 %
Turbine outlet	T_{4A}	0.70 K	0.29 %
LTHX outlet	T_5	0.39 K	0.14 %
Recuperator outlet (LP)	T_6	1.73 K	0.56 %
HTHX inlet (sink)	T_{101}	0.15 K	0.05 %
HTHX outlet (sink)	T_{102}	4.82 K	1.06 %
LTHX inlet (source)	T_{201}	0.31 K	0.11 %
LTHX outlet (source)	T_{202}	1.23 K	0.49 %
Compressor inlet	p_0	1047 Pa	1.07 %
Compressor outlet	p_1	2337 Pa	0.79 %
Turbine inlet	p_3	2485 Pa	0.88 %
Mass flow rate	$\dot{m}$	0.0065 kg/s	1.95 %

During operation of the CoBra heat pump, two main safety constraints require monitoring. *Compressor surge* can occur when a compressor operates at low mass flow rates, potentially causing flow destabilization and detachment, which may lead to structural damage. However, the radial compressors currently installed in the heat pump feature broad performance maps, resulting in high inherent flow stability. Therefore, compressor surge is not considered a limiting operational constraint in this work. *Thermal stresses* arise in heat exchangers due to differences in thermal expansion of the tubes and the shell. Analyses of the HX model show that transient stresses are strongly correlated with the rate of heat accumulation, while steady-state stresses are correlated with the logarithmic mean temperature difference in the HX. Consequently, the rate of heat accumulation $\dot{Q}_{\mathrm{acc}}$ is used as a measurable proxy for these stresses. It is determined from the energy balance across the heat exchanger (HX) using measured mass flow rates and inlet/outlet temperatures, as defined in Equation 1:

$$\dot{Q}_{\mathrm{acc}} = (\dot{m}c_p)_{\mathrm{C}} \cdot (T_1 - T_2) - (\dot{m}c_p)_{\mathrm{Si}} \cdot (T_{102} - T_{101}) \quad (1)$$

A cascaded control strategy, shown in Figure 4, is applied to the supply temperature (T_{102}) while limiting the rate of heat accumulation ($\dot{Q}_{\mathrm{acc}}$). The outer loop controls T_{102} and sets the target accumulation rate ($\dot{Q}^{\mathrm{set}}_{\mathrm{acc}}$) for the inner loop, which adjusts the motor speed (n_M) accordingly. Limiting the outer loop's output combined with appropriate tuning of both controllers ensures that the thermal stress safety constraint is met.

The following three temperature control concepts are investigated in this work. Figure 5 shows schematics of the control concepts.

DOI
10.3384/ecp2181017

Concept 1: PI controllers This control approach utilizes proportional-integral (PI) controllers. Temperature control is achieved using the cascaded controller (`PI_T`) illustrated in Figure 4. In parallel, a separate PI controller (`PI_F`) adjusts p_0 to maintain a cycle-to-sink mass flow ratio of 1 ($\dot{m}_C/\dot{m}_{\mathrm{Si}} = 1$), as this ratio is typically near-optimal for minimizing heat transfer exergy losses. Throughout the operation, the three-way valve position is fixed at full recuperation ($\theta_{3W} = 0$), as heat recuperation is beneficial when the difference between T_2 and T_5 is sufficiently large to outweigh the additional pressure losses.

Concept 2: RL policy controller In this approach, a policy (usually a feed-forward neural network) trained through RL adjusts n_M, θ_{3W} and p_0 depending on the observed heat pump state and set-point value provided to the policy. Prior to deployment, the RL policy is trained to perform control actions that maximize an accumulated reward signal which incorporates the control task including the safety constraints.

Concept 3: PI and RL This approach combines traditional control and RL by using the cascaded controller for temperature control and a RL policy that learns to optimize the other control actions.

4 Reinforcement Learning

The objective of RL is for the agent to learn a policy π that maximizes the cumulative sum of rewards expected over time.

As described by Sutton and Barto (2018) and Achiam (2018), the standard RL setup involves an agent and an environment that interact over a series of time steps. At each step t, the agent receives an observation o_t from the environment. Based on this observation, which may be a subset of the full state s_t, the agent selects an action a_t according to its current policy π.

$$a_t = \pi(o_t) \tag{2}$$

In response to the action, the environment transitions to a new state s_{t+1} and provides the agent with a numerical reward r_t. This reward indicates the immediate value associated with taking action a_t in state s_t, as defined by a reward function R:

$$r_t = R(s_t, a_t) \tag{3}$$

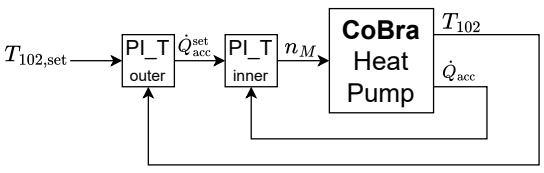
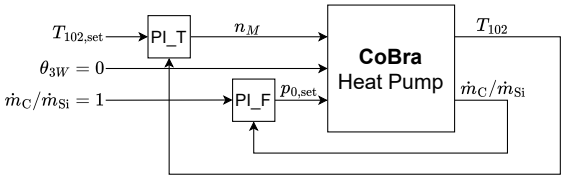
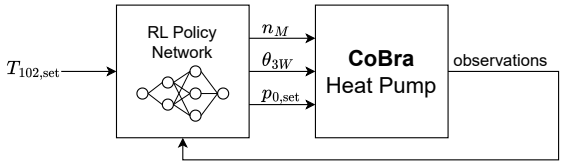


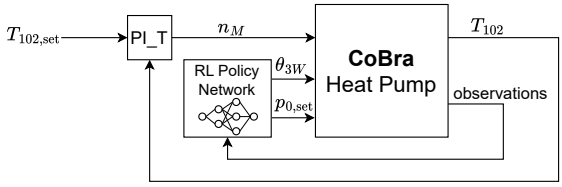
Figure 4. Cascaded controller for temperature control with limits for heat accumulation.



(a) Concept 1: PI controllers



(b) Concept 2: RL policy controller



(c) Concept 3: PI and RL

Figure 5. Control concepts for controlling the supply temperature (T_{102}) using the motor speed (n_M), three-way valve opening (θ_{3W}) and compressor inlet pressure (p_0).

4.1 Training setup

The RL training methodology in this work, depicted in Figure 6, employs established tools and standards for interfacing the heat pump model and the RL algorithm. The model is exported as a Co-Simulation Functional Mock-up Unit (FMU). A custom, Gymnasium-compliant (Towers et al. 2024) `Environment` interfaces with the FMU, providing motor speed, three-way valve opening, and compressor inlet pressure as control inputs.

The `Environment` uses the FMPy library to interface with the FMU. Following the modular design proposed by Tassa et al. (2018), problem-specific logic is delegated to a `Task` module, which calculates the reward based on the state and objective and adjusts simulation parameters and boundary conditions. This abstraction allows tasks to be interchanged - for instance, to switch between randomized and deterministic scenarios. Each episode begins from a "cold start" state, meaning all initial pressures and temperatures within the heat pump are set to ambient conditions.

Training is performed on episodes with a fixed length of 700 steps using a step size of 30 s, resulting in 21 000 s of simulated time per episode. This work utilizes the Stable-Baselines3 library (Raffin et al. 2021) which provides implementations of RL algorithms and offers parallelization and monitoring tools.

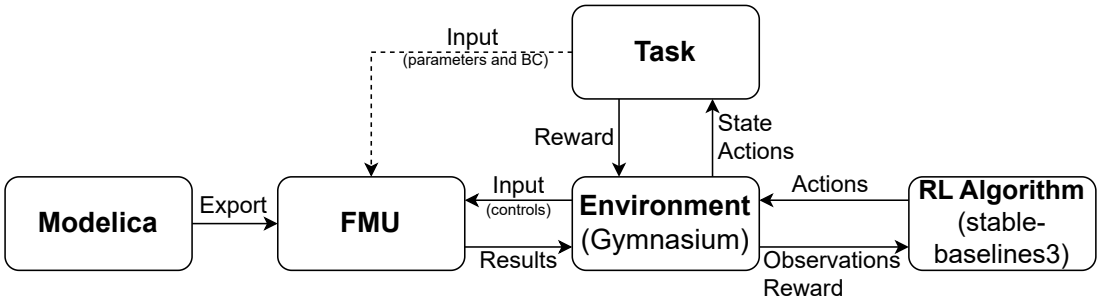


Figure 6. Task-supplied reinforcement learning using FMUs.

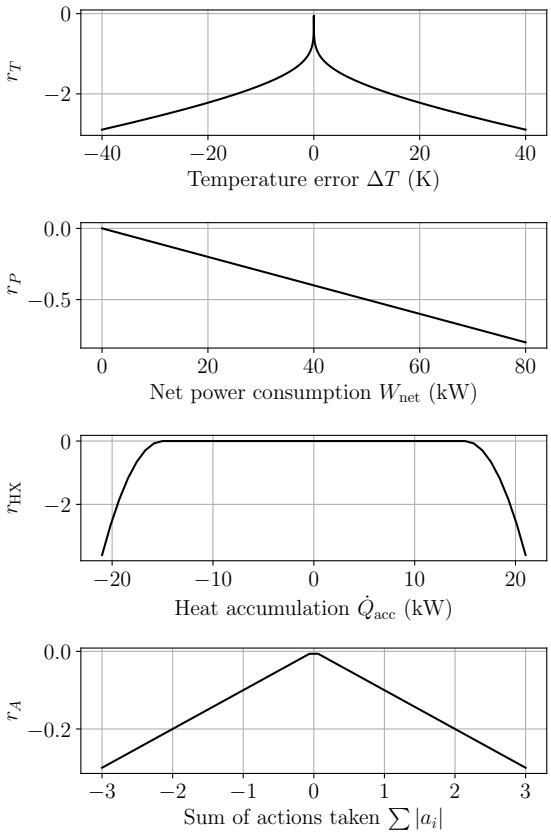


Figure 7. Components of the reward function.

4.2 Action and observation spaces

The action and observation spaces used for training are summarized in Table 2. As introduced in section 3, the action space consists of θ_{3W} and p_0 for both control concepts, with n_M additionally included for concept 2. Instead of outputting absolute values, the controller determines the desired *change* for each control input (delta action). The environment then adds this delta to the current value. This approach inherently constrains the slew rates of the actuators, while the simulation model enforces separate limits on the final values to keep them within their allowable operational range. For this study, the lower bound for compressor inlet pressure (p_0) was lowered from 0.8 bar to 0.55 bar. This adjustment was made because experimental results suggest the optimal p_0 value might be below the original 0.8 bar limit. The observation space includes the actual actuator signals, the rate of heat accumulation, pressure ratio, and various temperature and flow signals from the plant. All signals from the observation space are measured in the physical plant.

Table 2. Action and observation spaces.

Action space	
Change of motor speed	Δn_M
Change of three-way valve opening	$\Delta\theta_{3W}$
Change of compressor inlet pressure	Δp_0
Observation space	
Temperature control error	ΔT
Rate of heat accumulation	$\dot{Q}_{\text{acc}}$
Motor speed	n_M
Three-way valve opening	θ_{3W}
Compressor inlet pressure	p_0
Cycle mass flow rate	$\dot{m}_C$
Sink mass flow rate	$\dot{m}_{\text{Si}}$
Sink inlet temperature	T_{101}
Sink outlet temperature	T_{102}
Compressor outlet temperature	T_1
HTHX outlet temperature	T_2
Compressor pressure ratio	Π_C
Coefficient of performance	COP

4.3 Reward function

A major challenge in RL is the reward function design: maximizing the reward must yield the desired control behavior, yet the function should also guide the agent's learning. The reward function in Equation 4 used in this work is a sum of four reward components. Each component is defined as a penalty term (a negative reward) representing a training goal, and the agent maximizes the total

DOI
10.3384/ecp2181017

reward by learning to minimize these penalties.

$$R = r_T + r_P + r_{\text{HX}} + r_A \tag{4}$$

$$r_T = b_T \cdot \left(|\Delta T|^{0.2} + 0.02 \cdot |\Delta T|\right) \tag{5}$$

$$r_P = b_P \cdot W_{\text{net}} \tag{6}$$

$$r_{\text{HX}} = \begin{cases} b_{\text{HX}} \cdot \left(|\dot{Q}_{\text{acc}}| - \dot{Q}_{\text{acc}}^{\text{lim}}\right)^2 & \text{if } |\dot{Q}_{\text{acc}}| > \dot{Q}_{\text{acc}}^{\text{lim}} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

$$r_A = b_a \sum_i |a_i| \tag{8}$$

For the temperature tracking goal r_T, the penalty is calculated based on the control error ΔT. It includes terms proportional to $|\Delta T|^{0.2}$ and $|\Delta T|$. This structure leads to the behaviour shown in Figure 7. The linear term ($|\Delta T|$) provides a consistent gradient signal while the power term ($|\Delta T|^{0.2}$) creates a steep reward gradient near the setpoint. This discourages the agent from trading off precise temperature tracking for the conflicting goal of energy minimization.

The r_P component directly penalizes net power consumption W_{net}, thereby incentivizing energy-efficient operation.

r_{HX} penalizes the rate of heat accumulation ($\dot{Q}_{\text{acc}}$) if it exceeds a soft limit, $\dot{Q}_{\text{acc}}^{\text{lim}}$. The penalty increases quadratically with the level of exceedance. This design makes minor and brief overshoots more acceptable than large and sustained violations.

r_A penalizes the magnitude of the control actions taken, discouraging excessive actuation and promoting smoother control.

The overall behavior is shaped by the weighting coefficients (b_T, b_P, b_{HX}, b_a) within the reward components. These coefficients are carefully selected to prioritize certain objectives. For example, maintaining the target temperature (r_T) and ensuring heat exchanger safety (r_{HX}) were weighted more heavily than optimizing for energy efficiency (r_P) or minimizing control action (r_A). In this work, good learning results were found for coefficient values of $b_T = -1$, $b_P = -1 \times 10^{-5}$, $b_{\text{HX}} = -1 \times 10^{-7}$ and $b_a = -0.1$.

4.4 Domain randomization

Domain randomization is a method proposed by Tobin et al. (2017), addressing the Sim2Real challenge when deploying the agent on the real system. The idea of domain randomization is to expose the RL algorithm to variance (e.g. system dynamics, set-points, boundary conditions) during training in order to train a model that generalizes and works across all variants. In this work, randomization is implemented in two ways:

Parameter variation: At the beginning of each training episode, the model parameters (including heat transfer characteristics, turbomachinery map scalers, and actuator rise times) are sampled from uniform distributions to introduce variability. The bounds for this sampling are set symmetrically around each parameter's nominal value, defining a range that is likely to contain the true physical value. The width of this range is selected according to the estimated uncertainty for each parameter. For example, well-calibrated parameters like turbomachinery map scalers are varied over a narrow range ($\pm 2\,\%$), while more uncertain parameters such as heat transfer scalers and actuator delays are given a wider range (up to $\pm 40\,\%$).

Set-point variation: Within each episode, the temperature set-point $T_{102,\text{set}}$ is varied. It follows diverse signal types including steps, ramps, constant values, harmonics, and random walks within the heat pump's temperature operating range.

4.5 Training with Soft Actor-Critic

Training was conducted using Soft Actor-Critic (SAC) developed by Haarnoja et al. (2018). SAC is an off-policy actor-critic algorithm in which a policy network, the *actor*, is optimized using feedback from a value function network, the *critic*. It is based on the maximum entropy RL framework, where the policy is trained to maximize and balance entropy (exploration) and expected returns (exploitation) by automatically adjusting the entropy coefficient. This algorithm was selected due to its sample efficiency and demonstrated suitability for continuous control tasks.

Hyperparameters for training were determined by hyperparameter optimization using RL Baselines3 Zoo (Raffin 2020). In addition to the tuning results, a linear learning rate schedule was added, starting higher for rapid initial progress and decreasing over time to allow for finer policy adjustments. The hyperparameters used to produce the presented results are summarized in Table 3.

Table 3. Hyperparameters used for training.

Parameter	*Value*
Learning rate	5×10^{-5} to 5×10^{-6}
Parallel environments	8
Training frequency	1 per step
Gradient steps	8
Batch size	512
Replay buffer size	1×10^6
Discount rate	0.99
Entropy coefficient	Auto
Actor network	2 layers, each 128 neurons
Critic network	2 layers, each 256 neurons

To accelerate training, the training process was parallelized across 8 environments using the `SubProcEnv` implementation of Stable-Baselines3. To assess the effect of random initialization, all training was conducted independently on multiple agents, each for 4 million environment steps. A single training run required approximately 36 hours on a scientific computing workstation, and agents were saved periodically using the library's callback mechanism.

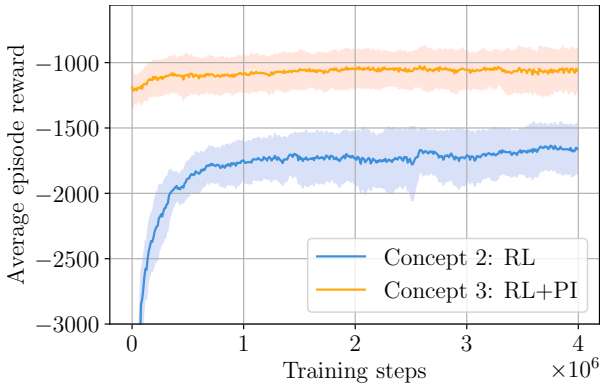


Figure 8. Average episode reward during training. Solid line: Average reward across all agents. Shaded area: Standard deviation of reward.

5 Results

This section evaluates the proposed control concepts, beginning with an analysis of the RL agents' training progress. All three concepts are then evaluated on a nominal baseline model, where the RL+PI agent demonstrates a novel and efficient control strategy developed during training. Finally, the RL agents' robustness and generalization capabilities are tested against a range of randomized model variations.

5.1 Training progress

Figure 8 shows the training performance, plotting the mean episode reward (solid line) and standard deviation (shaded area) across all runs.

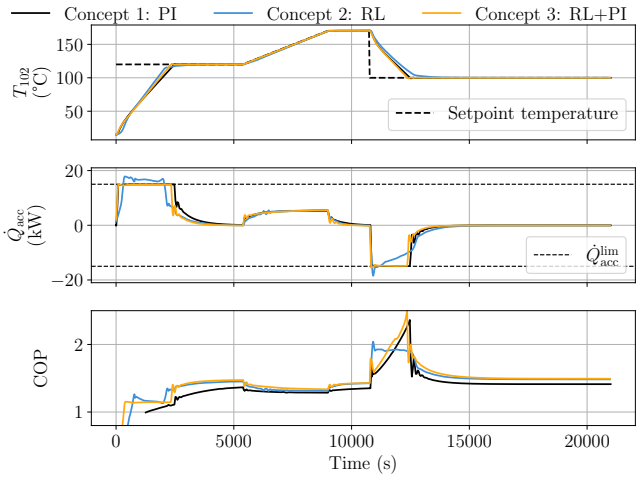
The learning curves reveal different training dynamics. Concept 2 shows a standard learning progression, with rewards rising steeply as the agent learns the control tasks from scratch. In contrast, concept 3 begins with a high initial reward that improves only marginally. This is because its embedded PI controller already handles the primary temperature control objective. The RL agent's role is thus limited to optimizing secondary, lower-weighted reward terms for efficiency and control actions.

The fluctuations observed in both curves likely stem from the inherent stochasticity of the training process, amplified by domain randomization.

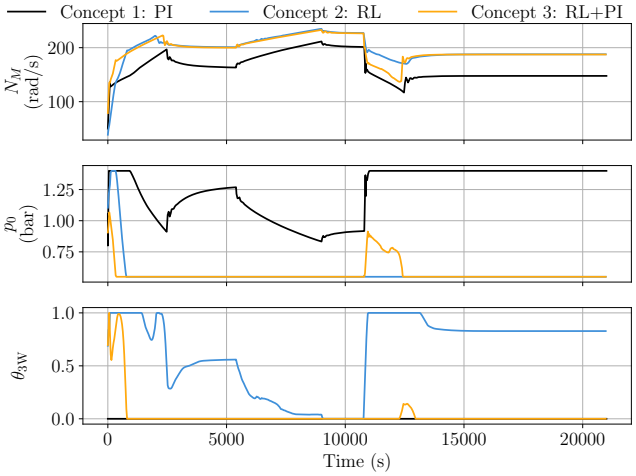
5.2 Control performance on baseline model

To directly compare the control performance of the three concepts, the best-performing agent for each was evaluated in a deterministic, non-randomized environment. The agents were tasked with tracking a temperature set-point trajectory not seen during training, as shown in Figure 9a. This trajectory includes a step increase to 120 °C, a linear ramp to 170 °C, a hold at that temperature, and a final step decrease to 100 °C.

Figure 9 shows the control and process variables for all control concepts. While concepts 1 and 3 rely on embedded PI controllers for temperature tracking and heat accumulation, concept 2's RL policy learned to perform



(a) Process variables



(b) Control variables

Figure 9. Comparison of the three control concepts for temperature tracking with a varying set-point signal on the baseline model, showing (a) process variables and (b) control variables.

these tasks, showing relatively accurate temperature control. Regarding the thermal stress constraint, concept 3 strictly maintains the soft limit for heat accumulation ($\dot{Q}_{acc}^{lim}$), whereas concept 2 exceeds this limit by a slight but acceptable margin.

Notably, concept 3 achieves the highest efficiency, maintaining a higher COP for nearly the entire duration of the maneuver. This performance is the result of a novel policy discovered by the RL agent. Similar to the baseline in concept 1, this policy keeps the three-way valve opening (θ_{3W}) fixed at 0 (fully recuperated operation). However, it manipulates the compressor inlet pressure (p_0), minimizing it during steady-state operation and increasing it during transients. This pressure manipulation strategy is noteworthy for two reasons:

1. *Steady-state operation:* The decision to lower p_0 is initially counterintuitive. A lower inlet pressure is creates imbalanced mass flow ratios in the heat exchanger, which increases entropy generation of heat

DOI
10.3384/ecp2181017

transfer. However, the observed improvement in COP indicates that the efficiency gains from reduced overall pressure losses successfully outweigh the increased exergy losses from heat transfer.

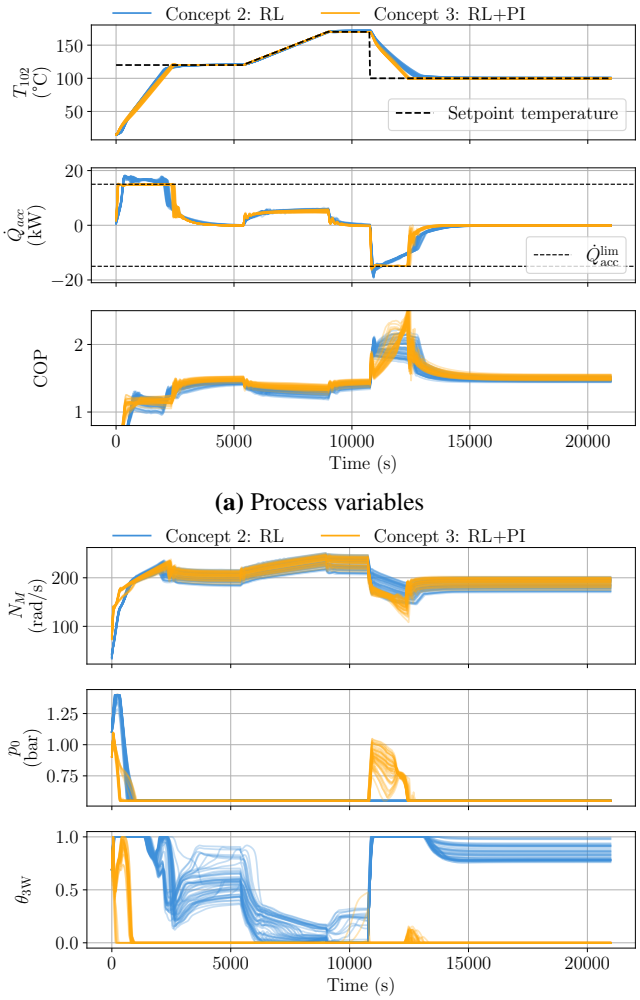
2. *Transient operation:* The strategy of increasing p_0 when ΔT is high (e.g. after a set-point change) is a particularly surprising discovery. Further analysis reveals that this action allows the system to converge to the new temperature set-point faster without exceeding the soft limit on heat accumulation.

Although further performance gains are anticipated with longer training and hyperparameter optimization, these findings confirm that RL can be successfully integrated with traditional PI controllers to discover novel and more efficient control policies.

5.3 Control performance on model variations

To assess the robustness of the learned policies against uncertainties of the system properties and dynamics, the best-performing agent from each RL control concept was evaluated on 50 model variations. These variations were sampled from the parameter range used for domain randomization.

The results from all 100 evaluation runs (50 for each RL concept) are illustrated in Figure 10. The figure shows that both RL concepts maintain temperature tracking and thermal stress limitation across the entire set of variations. Furthermore, the general control strategies employed by the agents remain consistent with those observed under nominal conditions, as described in subsection 5.2. This suggests that the RL agents are capable of generalizing to varied system properties and can maintain control performance in the presence of model uncertainty.



(a) Process variables

(b) Control variables

Figure 10. Comparison of the RL control concepts for temperature tracking with a varying set-point signal on randomized model variations, showing (a) process variables and (b) control variables.

6 Conclusion

This study investigated RL for controlling a high-temperature Brayton cycle heat pump, using an experimentally validated model. Different control concepts were compared: traditional PI controllers (Concept 1), a pure RL policy (Concept 2), and a hybrid PI+RL approach (Concept 3). Training was performed with the Soft Actor-Critic algorithm, and domain randomization was employed to improve controller robustness against parameter uncertainties.

The results indicate that while the pure RL agent learned the fundamental control tasks, the hybrid PI+RL approach (Concept 3) achieved higher operational efficiency. This improved performance was attributed to the discovery of a novel strategy for manipulating the compressor inlet pressure (p_0). The learned policy minimizes p_0 during steady-state operation to maximize efficiency, while temporarily increasing it during transients. This transient pressure increase, applied when the set-point temperature was changed, was a particularly noteworthy finding. It enables the system to converge to new set-points faster without exceeding thermal stress constraints. The controllers also demonstrated robust operation across a range of model parameters.

While the simulation results are positive, they do not serve as a guarantee of successful real-world deployment. The transfer and validation of the controller on the physical system are therefore planned for future work.

In summary, the findings suggest that RL, particularly when paired with traditional PI(D) controllers, is a viable method for controlling Brayton heat pumps and has the potential to increase their efficiency.

References

Achiam, Joshua (2018). *Spinning Up in Deep Reinforcement Learning*. URL: https://spinningup.openai.com/en/latest/index.html (visited on 2025-06-27).

Åkesson, Johan (2008). "Optimica—An Extension of Modelica Supporting Dynamic Optimization". English. In: *Proceed-*

ings of the 6th International Modelica Conference. The 6th International Modelica Conference (March 3–4, 2008).

Andersson, Joel A. E. et al. (2019). "CasADi: a software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation* 11.1. PII: 139, pp. 1–36. ISSN: 1867-2949. DOI: 10.1007/s12532-018-0139-4.

Casella, Francesco and Alberto Leva (2006). "Modelling of thermo-hydraulic power generation processes using Modelica". In: *Mathematical and Computer Modelling of Dynamical Systems* 12.1, pp. 19–33. ISSN: 1387-3954. DOI: 10.1080/13873950500071082.

Di Cao et al. (2020). "Reinforcement Learning and Its Applications in Modern Power and Energy Systems: A Review". In: *Journal of Modern Power Systems and Clean Energy* 8.6, pp. 1029–1042. ISSN: 2196-5625. DOI: 10.35833/MPCE.2020.000552.

Haarnoja, Tuomas et al. (2018). "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. 10–15 Jul. PMLR, pp. 1861–1870. URL: https://proceedings.mlr.press/v80/haarnoja18b.html.

Heat Pump Centre (2023). *IEA HPT Annex 58. High-Temperature Heat Pumps*. URL: https://heatpumpingtechnologies.org/annex58/ (visited on 2024-04-25).

IEA (2024). *World Energy Outlook 2024*. Ed. by IEA. Paris. URL: https://www.iea.org/reports/world-energy-outlook-2024 (visited on 2024-10-31).

Lukianykhin, Oleh and Tetiana Bogodorova (2019). "ModelicaGym: Applying Reinforcement Learning to Modelica Models". In: *Proceedings of the 9th International Workshop on Equation-based Object-oriented Modeling Languages and Tools*. EOOLT '19: 9th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools (Berlin Germany, November 5, 2019). Ed. by Christoph Nytsch-Geusen and Olaf Enge-Rosenblatt. New York, NY, USA: ACM, pp. 27–36. ISBN: 9781450377133. DOI: 10.1145/3365984.3365985.

Magnusson, Fredrik and Johan Åkesson (2015). "Dynamic Optimization in JModelica.org". In: *Processes* 3.2. PII: pr3020471, pp. 471–496. DOI: 10.3390/pr3020471.

Modelon, Inc. (2024). *OPTIMICA Compiler Toolkit*. URL: https://help.modelon.com/latest/reference/oct/ (visited on 2025-07-30).

Oehler, Johannes, Jens Gollasch, et al. (2021). "Part Load Capability of a High Temperature Heat Pump with Reversed Brayton Cycle". In: *13th IEA Heat Pump Conference 2021 (HPC2020) Conference Proceedings*. 13th IEA Heat Pump Conference (Jeju, Korea, April 26–29, 2021). Ed. by International Energy Agency.

Oehler, Johannes, A. Phong Tran, and Panagiotis Stathopoulos (2022). "Simulation of a Safe Start-Up Maneuver for a Brayton Heat Pump". In: *Volume 4: Cycle Innovations; Cycle Innovations: Energy Storage*. ASME Turbo Expo 2022: Turbomachinery Technical Conference and Exposition (Rotterdam, Netherlands, June 13–17, 2022). American Society of Mechanical Engineers. ISBN: 978-0-7918-8601-4. DOI: 10.1115/GT2022-79399.

Raffin, Antonin (2020). *RL Baselines3 Zoo*. URL: https://github.com/DLR-RM/rl-baselines3-zoo (visited on 2025-06-27).

Raffin, Antonin et al. (2021). "Stable-Baselines3: Reliable Reinforcement Learning Implementations". In: *Journal of Machine Learning Research* 22.268, pp. 1–8. URL: http://jmlr.org/papers/v22/20-1364.html.

Silver, David et al. (2018). "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play". eng. In: *Science (New York, N.Y.)* 362.6419. Journal Article Research Support, Non-U.S. Gov't, pp. 1140–1144. DOI: 10.1126/science.aar6404. eprint: 30523106.

Sutton, Richard S. and Andrew Barto (2018). *Reinforcement learning. An introduction*. eng. Second edition. Adaptive computation and machine learning. Cambridge, Massachusetts and London, England: The MIT Press. 526 pp. ISBN: 978-0262039246.

Tang, Chen et al. (2025). "Deep Reinforcement Learning for Robotics: A Survey of Real-World Successes". In: *Annual Review of Control, Robotics, and Autonomous Systems* 8.1, pp. 153–188. DOI: 10.1146/annurev-control-030323-022510.

Tassa, Yuval et al. (2018). *DeepMind Control Suite*. DOI: 10.48550/arXiv.1801.00690.

Tobin, Josh et al. (2017). "Domain randomization for transferring deep neural networks from simulation to the real world". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (Vancouver, BC, September 24–28, 2017). IEEE, pp. 23–30. ISBN: 978-1-5386-2682-5. DOI: 10.1109/IROS.2017.8202133.

Towers, Mark et al. (2024). *Gymnasium: A Standard Interface for Reinforcement Learning Environments*. DOI: 10.48550/arXiv.2407.17032.

Tran, A. Phong and Panagiotis Stathopoulos (2025). "Dynamic simulation and experimental validation of a high-temperature Brayton heat pump". In: *Applied Thermal Engineering* 274. PII: S1359431125011287, p. 126536. ISSN: 13594311. DOI: 10.1016/j.applthermaleng.2025.126536.

Wetter, Michael et al. (2014). "Modelica Buildings library". In: *Journal of Building Performance Simulation* 7.4, pp. 253–270. ISSN: 1940-1493. DOI: 10.1080/19401493.2013.765506.

Wrede, Konstantin et al. (2024). "FMUGym: An Interface for Reinforcement Learning-based Control of Functional Mock-up Units under Uncertainties". In: *31st International Workshop on Intelligent Computing in Engineering, EG-ICE 2024*, pp. 647–656. DOI: 10.35869/Proceedings_EGICE2024. URL: https://publica.fraunhofer.de/entities/publication/d31204fd-3480-469f-9c45-5c376d81a721/fullmeta.

Yücel, Fatma Cansu et al. (2025). "Design and Commissioning of the Brayton High-Temperature Heat Pump "Cobra"". In: DOI: 10.2139/ssrn.5290358.

Zhao, Wenshuai, Jorge Pena Queralta, and Tomi Westerlund (2020). "Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: a Survey". In: *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. 2020 IEEE Symposium Series on Computational Intelligence (SSCI) (Canberra, ACT, Australia, December 1–4, 2020). IEEE, pp. 737–744. ISBN: 978-1-7281-2547-3. DOI: 10.1109/SSCI47803.2020.9308468.

Zühlsdorf, B. et al. (2019). "Analysis of technologies and potentials for heat pump-based process heat supply above 150 °C". In: *Energy Conversion and Management: X* 2. PII: S2590174519300091, p. 100011. ISSN: 25901745. DOI: 10.1016/j.ecmx.2019.100011.