Modellierung mechatronischer Systeme mit Modelica

M. Otter, **C. Schweiger**, DLR, Oberpfaffenhofen

Kurzfassung

Es wird eine Übersicht über die objektorientierte Modellierungssprache Modelica und die Modelica-Simulationsumgebung Dymola gegeben. Modelica eignet sich besonders zur Modellierung mechatronischer Systeme, die sich aus elektrischen, mechanischen, hydraulischen, pneumatischen, thermischen, fluidmechanischen und regelungstechnischen Teilkomponenten zusammensetzen. Durch die parametrische Darstellung komplexer Systeme sind Modelica-Modelle ein sehr guter Ausgangspunkt für die nichtlineare Parameteridentifikation und den optimierungsbasierten Systementwurf.

1. Einleitung

Mechatronische Systeme sind kurzen Entwicklungszyklen unterworfen. Folglich besteht ein Bedarf an Methoden und Werkzeugen, die dieser Situation gerecht werden können. So hat sich in der jüngeren Vergangenheit die Vorgehensweise des objektorientierten Modellierens mit der Sprache Modelica® zur Simulation mechatronischer Systeme etabliert. Modelica ist besonders geeignet zur Erstellung multiphysikalischer Systemmodelle, was der sowohl mechanischen, elektrischen als auch informationstechnischen Natur mechatronischer Systeme in wesentlich höherem Maße entgegenkommt, als dies bei vielen fachgebietsspezifischen Simulationspaketen der Fall ist.

In diesem Artikel wird, basierend auf [15] eine Übersicht über die Modellierungssprache Modelica gegeben, mit der multidisziplinäre Modelle komfortabel definiert und mit Hilfe eines geeigneten Übersetzers und einer Simulationsumgebung effizient simuliert werden können.

Aus Benutzersicht werden in Modelica Modelle durch Objektdiagramme beschrieben. In Bild 1 ist eine Collage solcher Modelle aus unterschiedlichen Anwendungsgebieten beispielhaft zusammengestellt.

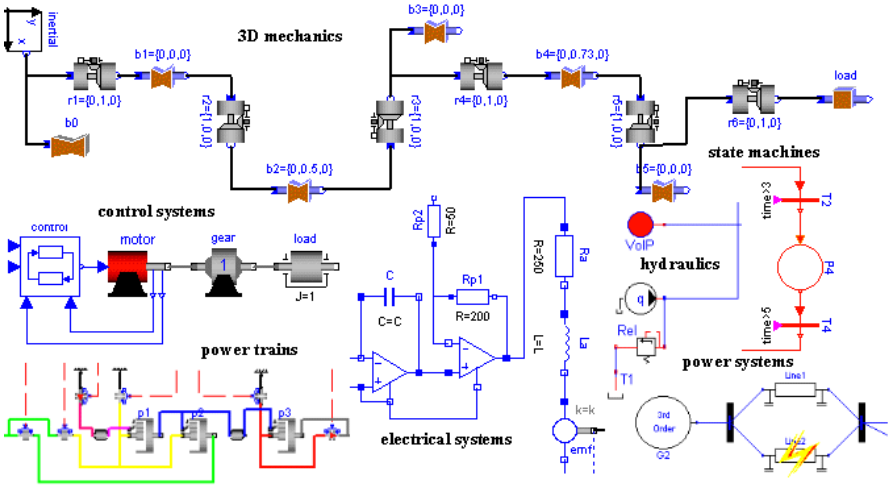


Bild 1: Modelica-Objektdiagramme aus unterschiedlichen Fachgebieten

Ein Objektdiagramm ist dabei eine Verallgemeinerung eines Blockschaltbildes, wobei die Schnittstellen von Komponenten nicht nur gerichtete Signale, sondern auch physikalische Schnittstellen enthalten können (wie mechanischer Flansch, elektrische Klemme). Eine Komponente eines Objektdiagramms ist wiederum hierarchisch aus Objektdiagrammen aufgebaut. Auf unterster Ebene werden Komponenten durch Differential-, algebraische und diskrete Gleichungen beschrieben.

Modelica ist eine frei verfügbare Sprache die zusammen mit der ebenfalls frei verfügbaren Modelica-Standard-Bibliothek seit 1996 von der gemeinnützigen Modelica Association kontinuierlich entwickelt wird. Modelica basiert wesentlich auf der von Hilding Elmqvist entwickelten objektorientierten Modellierungsmethodik [6]. Die ersten Anwendungen mit Modelica basierend auf der kommerziellen Modelica-Simulationsumgebung Dymola [4] gibt es seit Ende 1999. Seitdem steigt die Anzahl der industriellen Anwendungen stark an. Detaillierte Informationen zu Modelica sind im Internet erhältlich (siehe Tabelle 1).

Tabelle 1: Informationen über Modelica im Internet

Homepage	http://www.modelica.org/
Tutorial	http://www.modelica.org/documents/ModelicaTutorial14.pdf
Sprachdefinition	http://www.modelica.org/documents/ModelicaSpec21.pdf
Modellbibliotheken	http://www.modelica.org/libraries.shtml
Veröffentlichungen	http://www.modelica.org/publications.shtml
Werkzeuge	http://www.modelica.org/tools.shtml

2. Grundlagen von Modelica

Wie im letzten Abschnitt skizziert, ist ein Modelica-Modell hierarchisch aus Objektdiagrammen aufgebaut, wobei auf unterster Ebene ein Modell durch mathematische Gleichungen beschrieben ist. Zur Verdeutlichung ist das Objektdiagramm eines einfachen Antriebsstrangmodells in Bild 2 zu sehen.

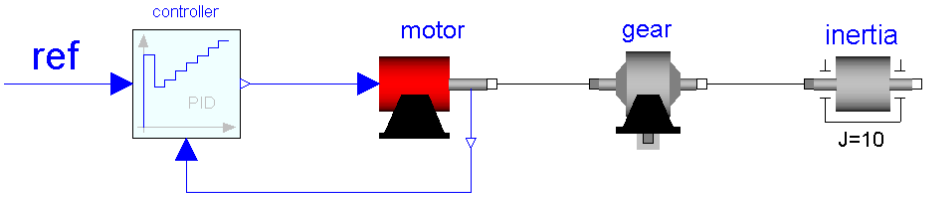


Bild 2: Objektdiagramm eines einfachen Antriebsstrangs

Dieses Modell besteht aus einem elektrischen Motor, einem Getriebe, einer Last und dem Regelungssystem und wird wie in Tabelle 2 gezeigt in Modelica definiert (Die Grafik eines Objektdiagramms wird innerhalb des Sprachelements **`annotation`** festgelegt. Aus Gründen der Übersichtlichkeit wird diese Definition jedoch in diesem Artikel nicht gezeigt. Alle Bildschirmabzüge von Objektdiagrammen wurden mit Dymola erstellt.)

Tabelle 2: Modelica-Text für das Modell eines einfachen Antriebsstrangs

```
model MotorDrive
  PID        controller;
  Motor      motor;
  Gearbox    gear(n=100);
  Inertia    inertia(J=10);
equation
  connect(controller.outPort, motor.inPort);
  connect(controller.inPort , motor.outPort);
  connect(gear.flange_a     , motor.flange_b);
  connect(gear.flange_b     , inertia.flange_a);
end MotorDrive;
```

Am Beginn des Modells werden vier Komponenten deklariert. Die Anweisung `Gearbox gear(n=100)` definiert eine neue Komponente `gear` von der vorhandenen Modellklasse `Gearbox` und setzt die Übersetzung `n`, einen Parameter, auf den Wert 100. Im unteren Teil des Modells wird definiert, wie die Schnittstellen der Komponenten miteinander verschaltet werden. Das Objektdiagramm des Motors ist in Bild 3 zu sehen. Das Dreieck im linken Teil des Bildes charakterisiert einen Signaleingang, während das Viereck im rechten Teil des Bildes die Schnittstelle eines mechanischen Flansches beschreibt.

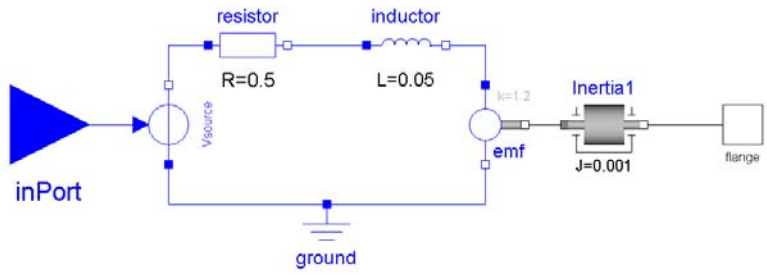


Bild 3: Objektdiagramm eines einfachen elektrischen Motors

Schnittstellen werden mit der Connector-Klasse definiert. Tabelle 3 zeigt elementare Connectoren, die in Modelica-Bibliotheken zur Verfügung gestellt werden. Diese können zum Aufbau von hierarchischen Connectoren, wie einem elektrischen „Plug“ oder einem „Signal-Bus“, verwendet werden.

Tabelle 3: Elementare Modelica-Connector-Definitionen

	Potentialvariable	**Flußvariable**	**Aussehen**
Elektrik	Elektrisches Potential	Strom	PositivePin NegativePin
Mechanik, translatorisch	Position (skalar)	Schnittkraft (skalar)	Flange_a Flange_b
Mechanik, rotatorisch	Winkel	Schnittmoment (skalar)	Flange_a Flange_b
Mechanik, 3-dimensional	Positionsvektor, Transformationsmatrix	Schnittkraft (vektoriell), Schnittmoment (vektoriell)	Frame_a Frame_b
Wärmeübertragung	Temperatur	Wärmestrom	HeatPort_a HeatPort_b
Hydraulik (inkompressibel)	Druck	Volumenstrom	Port_A Port_B
Pneumatik (kompressibel)	Druck	Massenstrom	Port_1 Port_2
Rohrströmung (mehrere Phasen und Substanzen)	Druck, spezifische Enthalpie, Massenanteil	Gesamt-/Einzelmassen-ströme, Enthalpiestrom,	FluidPort_a FluidPort_b
Signalflüsse	Real-/Integer-/Boolean-Signalvektor	—	InPort OutPort

In einer Connector-Klasse werden alle Variablen der Schnittstelle beschrieben. Zum Beispiel werden in der Connector-Klasse Flange der absolute Drehwinkel `phi` und das Schnittmoment im Flansch `tau` deklariert. Variablen können entweder vom Typ `Real`, `Boolean`, `Integer`, `String` oder eines davon abgeleiteten Typs sein, siehe Tabelle 4.

Tabelle 4: Modelica-Text zur Typendefinition

```
type Angle  = Real(quantity="Angle" , unit="rad", displayUnit="deg");
type Torque = Real(quantity="Torque", unit="N.m");
```

Eine Verbindungsgleichung der Form **connect**(Pin1, Pin2), wobei Pin1 und Pin2 Instanzen der Connector-Klasse Pin sind, definiert eine physikalische Verbindung der beiden Klemmen und erzeugt implizit die Gleichungen Pin1.v = Pin2.v und Pin1.i + Pin2.i = 0. Eine „Null-Summengleichung“ wird verwendet, wenn die entsprechenden Variablen in der Connector-Klasse mit dem Attribut **flow** als Flußvariablen gekennzeichnet sind. Wie in Tabelle 5 exemplarisch für eine Drehträgheit gezeigt, werden auf unterster Ebene Modelle mit Gleichungen beschrieben.

Tabelle 5: Modelica-Text für Drehträgheit

```
model Inertia "Drehtraegheit"
  parameter Real J=1 "Wert der Traegheit";
  Flange a, b;
  Real   w "Drehzahl";
equation
         a.phi = b.phi;   // gleiche Winkel (Kommentar)
    der(a.phi) = w;
  J*der(w)     = a.tau + b.tau;
end Inertia;
```

Eine Zeichenkette nach dem Modellnamen bzw. nach einer Deklaration enthält eine Kurzbeschreibung, die von Simulationsumgebungen speziell verwendet werden können, z. B. als Legende in einem Plot. Das Sprachelement **parameter** kennzeichnet eine Variable die während einer Simulation konstant ist. Der Operator **der**(...) definiert die Zeitableitung.
Neben den obigen Basis-Sprachelementen unterstützt Modelica ein- und mehrdimensionale Felder mit einer Matlab-ähnlichen Syntax. Jede Instanz einer Klasse kann als Feldelement verwendet werden. Auf diese Weise können einfach diskretisierbare partielle Differentialgleichungen komfortabel definiert werden. Modelica hat eine ausgefeilte Unterstützung von unstetigen und strukturvariablen Komponenten, wie elektrische Schalter, Lagerreibung, Kupplungen, Abtastsysteme etc. Hierbei werden die notwendigen Zeit- und Zustandsereignisse automatisch erzeugt. Schließlich hat Modelica ein leistungsfähiges Bibliothekskonzept mit Versionsverwaltung, um viele und große Komponentenbibliotheken verwalten zu können. Zum Beispiel kann eine Modell-Klasse automatisch im Dateisystem gefunden werden, wenn der hierarchische Modelica-Name dieser Klasse bekannt ist.

3. Modelica-Komponentenbibliotheken

Neben der Modelica-Sprache entwickelt die Modelica Association auch frei verfügbare Modelica-Bibliotheken. Alle ausgereiften Bibliotheken werden in der Bibliothek *Modelica* zusammengefasst. Darüber hinaus gibt es auch von anderen Personen und Organisationen freie und kommerzielle Bibliotheken. Eine Zusammenfassung der wichtigsten freien Bibliotheken ist in Tabelle 6 zu finden. Detailliertere Informationen zu allen diesen Bibliotheken gibt es unter http://www.modelica.org/libraries.shtml. Weitere Bibliotheken sind zur Zeit in der Entwicklung.

Tabelle 6: Auswahl von freien Modelica-Bibliotheken

Modelica	Modelica-Standard-Bibliothek
Blocks	ca. 65 kontinuierliche Ein-/Ausgangsblöcke wie PT1, Signalgeneratoren, ...
Constants	ca. 20 wichtige Konstanten wie π, g_n, σ
Electrical	ca. 70 analoge und schaltende elektrische Komponenten [3]
Icons	Vordefinierte Icons
Math	ca. 15 mathematische Modelica-Funktionen wie sin, cos
Mechanics	
Rotational	ca. 30 mechanisch rotatorisch Komponenten (1-dim.) inkl. viele Reibelemente
Translational	ca. 15 mechanisch translatorische Komponenten (1-dim.)
SIunits	ca. 450 Typ-Definitionen nach ISO, sowie Konversion zwischen Einheiten
Thermal	
HeatTransfer	ca. 25 Komponenten für 1-dim. Wärmeleitung
Modelica (neu)	*Neue*, fertige Unterbibliotheken, die in den nächsten Monaten in die Modelica-Bibliothek aufgenommen werden (können zum Teil schon von der Modelica-Web-Seite geladen werden)
Sampled	ca. 45 Blöcke für Abtastsysteme, IIR- und FIR-Filter, Multi-Raten, Rauschen, Konvertierung von kontinuierlich nach diskret (5 Verfahren)
Fluid	Rohrströmungen für ein- und mehr-phasige Fluide mit mehreren Substanzen
Media	1250 Mediummodelle, inkl. sehr detailliertem IF97-Modell für Wasser
Matrices	ca. 20 Modelica-Funktionen für Eigenwerte, Frequenzgang, LU, QR, SVD, e^A, ...
MultiBody	ca. 60 Komponenten und 75 Funktionen für 3-dim. Mechanik; 3D-Animation jeder Komponente, automatische und analytische Schleifenbehandlung [16]
ModelicaAdditions	Zusätzliche Basis-Bibliotheken vom DLR
Blocks	ca. 40 kontinuierliche/diskrete/logische Ein-/Ausgangsblöcke
HeatFlow1D	obsolet (1-dim. Wärmeleitung)
MultiBody	obsolet (3-dim. Mechanik)
PetriNets	Einfache Petri-Netze (nur für kleine Netze geeignet)
Tables	1- und 2-dim. Tabellen
VehicleDynamics	> 100 Komponenten für Fahrzeugdynamik inkl. komplette Fahrzeugmodelle (basiert auf Modelica.MultiBody) [1]
HyLibLight	ca. 30 Hydraulikkomponenten (Untermenge der kommerziellen HyLib)
PneuLibLight	ca. 45 Komponenten für technische Pneumatik (Untermenge von PneuLib)
ThermoFluid	sehr große Bibliothek für Rohrströmungen [5, 23] (wird durch Modelica.Media und Modelica.Fluid ersetzt)
ThermoPower	Rohrströmungen
Objectstab	elektrische Energiesysteme (für Stabilitätsuntersuchungen)
SPICElib	Untermenge von SPICE als Modelica-Komponenten [9]
WasteWater	biologische Abwasserreinigung [20]

Für mechatronische Systeme ist es besonders wichtig, dass Reibelemente sehr benutzerfreundlich, sowie effizient und robust unterstützt werden [18]. Ein typisches Beispiel ist in Bild 4 zu sehen. Im linken Teil des Bildes ist ein detailliertes Modell eines Sechsgang-Automatikgetriebes zu sehen, wobei in den Planetengetrieben und den fünf Kupplungen Coulomb-Reibung inklusive der Haftphasen modelliert werden. Die in den Planetengetrieben verwendeten Reibkennlinien (Reibmoment $\Delta\tau$ = f(Lastmoment τ_A, Drehzahl ω_A)) sind im rechten Teil von Bild 4 zu sehen.

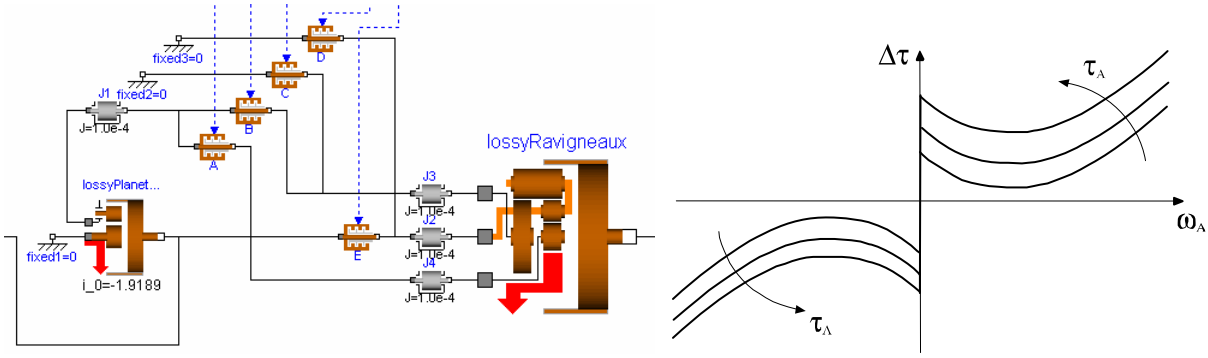
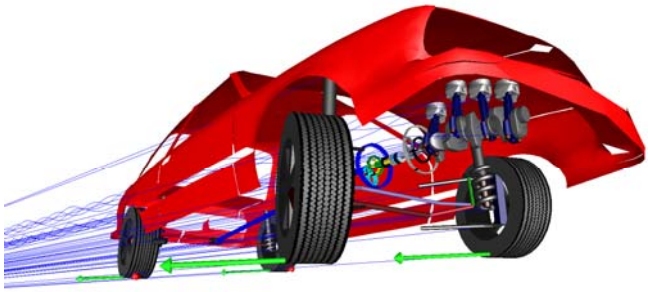


Bild 4: Detailliertes Modell eines Automatikgetriebes (links) und Reibkennlinien (rechts)

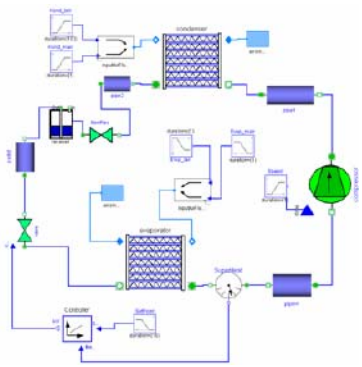


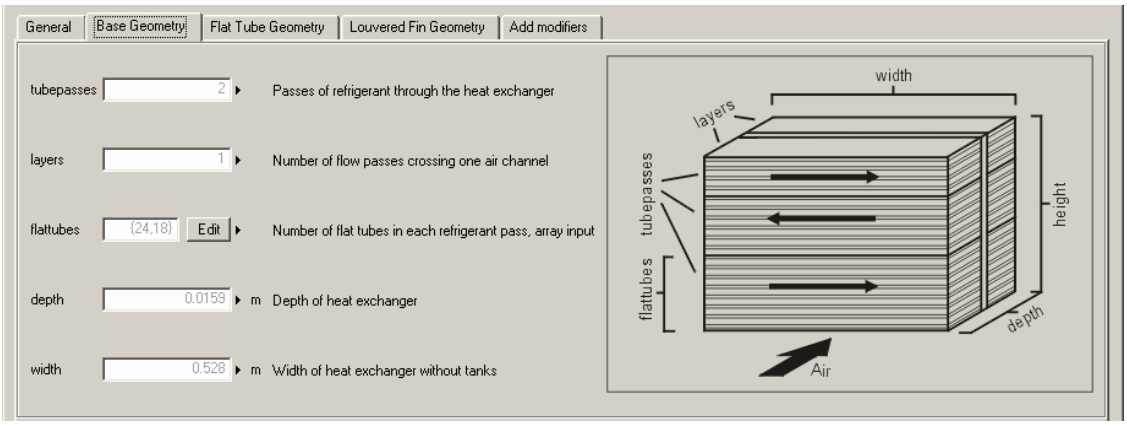
Im nebenstehenden Bild ist eine komplexere Anwendung zu sehen: Mit der VehicleDynamics-Bibliothek wird die Fahrdynamik modelliert, mit der Modelica.MultiBody-Bibliothek wird der Motor und die Animation beschrieben und mit der PowerTrain-Bibliothek wird der Antriebsstrang definiert, insbesondere ein detailliertes Modell des Automatikgetriebes.
Kommerziell verfügbare Modelica-Bibliotheken von unterschiedlichen Herstellern sind in Tabelle 7 aufgeführt (genauere Infos auch über die Modelica-Web-Seite)

Tabelle 7: Kommerzielle Modelica-Bibliotheken

HyLib	ca. 90 hydraulische Komponenten, wie Pumpen, Motoren, Ventile
PneuLib	Grosse Bibliothek zur Modellierung pneumatischer Schaltkreise
PowerTrain	ca. 45 Komponenten für Antriebsstränge, wie Planetengetriebe, Reibelemente, Kupplungen, detaillierte Automatikgetriebe-Modelle
AirConditioning	Parametrische Komponenten und Systeme für KFZ-Klimaanlagen (Beta)
Simelica	Transformation von Simulink-Modellen nach Modelica. Enthält Modelica-Bibliothek fast aller Simulink-Blöcke [10] (jedoch kein Stateflow). *Hinweis*: Transformation von jedem Modelica Modell nach Simulink als Cmex-S-Function (also volle Integration, *keine* Co-Simulation) wird als Option von Dymola angeboten.

Ganz neu ist die auf [5, 19, 23] basierende AirConditioning-Bibliothek, die vor allem von der deutschen KFZ-Industrie zur Simulation von KFZ-Klimaanlagen eingesetzt werden wird. Hervorzuheben ist insbesondere die leistungsfähige Modellierung von Zweiphasen-Strömungen wie sie in Klimaanlagen auftreten. Im nebenstehenden Bild ist ein typisches Systemmodell zu sehen. In dieser Bibliothek werden auch konsequent die in Modelica 2.1 eingeführten Sprachelemente zur einfachen Definition komplexer Parameter-Menüs verwendet (annotations: group, tab, enable), die in Dymola mit der Version 5.2 voll unterstützt werden. Im untenstehenden Bild ist als typisches Beispiel das Eingabe-Menü zur Definition eines Wärmetauschers zu sehen.





4. Simulation von Modelica-Modellen

Ein Modelica-Übersetzer erstellt aus einem Modelica-Modell eine Menge von Differential-, algebraischen und diskreten Gleichungen, indem die Gleichungen von allen Komponenten und die Gleichungen aufgrund von Verbindungen zu einem Gesamtgleichungssystem zusammengefaßt werden. Eine direkte numerische Lösung dieses Ausgangsgleichungssystems ist jedoch in den meisten Fällen höchst *uneffizient* und *unzuverlässig*. Die Modelica-Sprache wurde deswegen so entworfen, dass symbolische Transformationsalgorithmen angewandt werden können, um das Ausgangsgleichungssystem in eine Form zu transformieren, die mit Standard-Integrationsverfahren besser gelöst werden kann.

Die wesentlichen Schritte bei einer solchen Transformation sind Umsortieren und Aufteilen in explizit lösbare und implizit zu lösende Teilsysteme. Bei letzteren wird mittels BLT-Transformation [6] versucht, sie in möglichst kleine, voneinander unabhängige algebraische Schleifen

aufzuteilen. Eine weitere Reduzierung ermöglicht „intelligente Variablensubstitution“ mit dem Tearing-Verfahren [14].

Beim Zusammenschalten von Modelica-Komponenten kann es vorkommen, dass in unterschiedlichen Komponenten definierte Systemzustände miteinander verkoppelt und nicht mehr voneinander unabhängig sind. Eine direkte numerische Lösung daraus abgeleiteter Systeme ist problematisch. Mit geeigneter Variablentransformation bzw. der Anwendung der „Dummy-Derivative“-Methode [11, 12, 17] ist im allgemeinen ein zuverlässiges, robustes Lösen linearer bzw. nichtlinearer Differentialgleichungen und algebraischer Gleichungen möglich.

Aus Effizienzgründen und Praxiserwägungen werden häufig idealisierte Modelle eingesetzt (z. B. idealen Diodenkennlinie). Sie können in Modelica relativ problemlos formuliert werden, führen jedoch auf strukturvariable Gleichungssysteme, bei denen neben reellen Größen auch Boolsche bzw. Integer-Variable als Unbekannte auftreten. In Modelica lösen alle Wertänderungen von logischen Ausdrücken standardmäßig ein Ereignis aus, da hierbei potentiell eine Unstetigkeit eingeführt wird. Nach Detektion des Ereignisses wird die Integration angehalten, die Änderung des logischen Ausdrucks durchgeführt und die Integration neu gestartet. Bei dieser Vorgehensweise sind Boolsche Variable während der Integration bekannt (= Wert vom letzten Ereignis), nicht jedoch an einem Ereignispunkt. Dort liegt ein gekoppeltes System von reellen und Boolschen Unbekannten vor, das nicht mehr mit Standardalgorithmen gelöst werden kann. In [13] sind benötigte Erweiterungen dieser Algorithmen skizziert.

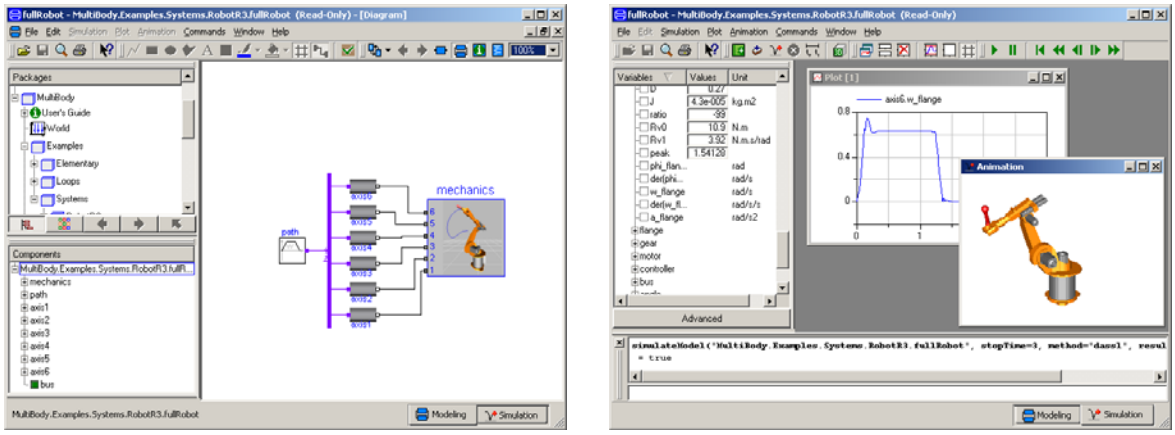


Bild 5: Bildschirmabzug von Dymola mit Modelldefinition (links) und Simulation/Plot (rechts)

Die beschriebenen Transformationsalgorithmen werden von der kommerziellen Modellierungs- und Simulationsumgebung Dymola [4] bereitgestellt, welche nahezu die gesamte Modelica-Sprache unterstützt. Zusätzlich enthält Dymola eine ganze Reihe weiterer Algorith-

men, um die Effizienz und Robustheit zu steigern. Zum Beispiel, werden die Gleichungen so umsortiert, dass alle *konstanten Gleichungen* vor Beginn der Simulation nur einmal und alle *Ausgangsgleichungen* nur an Kommunikationszeitpunkten ausgewertet werden. Dymola kann sehr grosse Systeme behandeln (siehe z. B. [2, 21]) und kann die transformierten Gleichungen auch in Form von SIMULINK-S-Functions als C-Code ausgeben, der einfach in SIMULINK [10] als Ein-/Ausgangsblock des Modells verwendet werden kann.

Mit MathModelica (http://www.mathcore.com/) steht eine zweite Modelica-Simulationsumgebung zur Verfügung, die intern die Symbolmaschine und den Simulator von Dymola benutzt und sehr gute Schnittstellen zu Mathematica besitzt. Schließlich wird zur Zeit mit OpenModelica (http://www.ida.liu.se/~pelab/modelica/OpenModelica.html) eine kostenlose Modelica-Simulationsumgebung entwickelt. Es ist jedoch noch nicht abzusehen, wann diese Umgebung für praktische Zwecke einsetzbar ist.

5. Zusammenfassung und Ausblick

Es wurde ein Überblick über die objektorientierte Modellierungssprache Modelica, verfügbare Modelica-Bibliotheken und Modelica-Simulationsumgebungen gegeben.

In [2, 21] wird das wohl größte bisher erstellte und mit Dymola simulierte multidisziplinäre Modell eines Gesamtfahrzeugs beschrieben. Dieses besteht aus einem detaillierten Fahrzeugdynamik-Modell (60 Gelenke, 70 Körper), dem Motor inklusive einem Verbrennungsmodell, dem Antriebsstrang mit detailliert modelliertem Automatikgetriebe, sowie einem Teil der hydraulischen Ansteuerung des Automatikgetriebes. Die symbolische Transformation dieses Modelica-Modells mit Dymola führt auf ein System von rund 25000 nicht-trivialen algebraischen Gleichungen und 320 Differentialgleichungen.

Modelica und die Modelica-Simulationsumgebung Dymola werden insbesondere auch für Echtzeitanwendungen eingesetzt: Zum Beispiel bei verschiedenen Automobilherstellern und -zulieferern zur Hardware-in-the-Loop-Simulation von Automatikgetrieben, sowie zur Generierung von inversen Dynamikmodellen für „embedded control“ bei Robotersteuerungen und bei neuartigen, im Flugversuch schon erfolgreich getesteten Autopilot-Reglern zum automatischen Landen von Verkehrsflugzeugen [8].

Zur Einführung in Modelica bieten sich insbesondere die Bücher von Michael Tiller [22] und das 1000-seitige Buch von Peter Fritzson [7] an.

Literatur

[1] Andreasson J.: VehicleDynamics library, Modelica ’03, S. 11-18
http://www.modelica.org/Conference2003/papers/h28_vehicle_Andreasson.pdf

[2] Bowles P., Tiller M., Elmqvist H., Brück D., Mattsson S. E., Möller A., Olsson H., Otter M.: Feasibility of Detailed Vehicle Modeling. SAE World Congress 2001, Nr. 2001-01-0334.

[3] Clauß C., Schneider A., Leitner T., Schwarz P.: Modelling of Electrical Circuits with Modelica, Modelica ’00, S. 3-12, http://www.modelica.org/workshop2000/proceedings/Schneider.pdf

[4] Dymola. http://www.dynasim.se/

[5] Eborn J.: On Model Libraries for Thermo-hydraulic Applications. PhD thesis ISRN LUTFD2/TFRT--1061--SE, 2001, Lund, Schweden, http://www.control.lth.se/articles/article.pike?action=fulltext&artkey=ebo01phd

[6] Elmqvist H.: A Structured Model Language for Large Continuous Systems. Dissertation. Report CODEN:LUTFD2/(TFRT--1015), Department of Automatic Control, Lund Institute of Technology, Lund, Schweden (1978)

[7] Fritzson P.: Principles of Object-Oriented Modeling and Simulation with Modelica. Wiley-IEEE Press (2003), http://www.mathcore.com/drmodelica/

[8] Joos H.-D., Looye G., Willemsen D.: Application of Optimization-Based Design Process for Robust Autoland Control Laws. AIAA-2001-4206. http://www.robotic.dlr.de/control/publications/2001/looye_Aiaa01_4206.pdf

[9] Martin C., Urquia A., Dormido S.: SPICELib - Modeling & Analysis of Electric Circuits with Modelica, Modelica ’03, S. 161-170, http://www.modelica.org/Conference2003/papers/h02_Martin.pdf

[10] The MathWorks. http://www.mathworks.com/

[11] Mattsson S. E., Söderlind G.: Index reduction in differential-algebraic equations using dummy derivatives. SIAM Journal of Scientific and Statistical Computing 14 (1993) S. 677-692

[12] Mattsson S. E., Olsson H., Elmqvist H.: Dynamic Selection of States in Dymola. Modelica ’00, S. 61-67, http://www.modelica.org/workshop2000/proceedings/Mattsson.pdf

[13] Otter M., Elmqvist H., Mattsson S. E.: Hybrid Modeling in Modelica based on the Synchronous Data Flow Principle. CACSD '99, http://www.modelica.org/papers/hybrid99.pdf

[14] Otter M.: Objektorientierte Modellierung Physikalischer Systeme, Teil 4, Transformationsalgorithmen. at Automatisierungstechnik 47 (1999) 4 S. A13-A16

[15] Otter M., Elmqvist H.: Modelica – Language, Libraries, Tools, Workshop and EU Project RealSim. Simulation News Europe, (2000), S. 3-8, http://www.modelica.org/documents/ModelicaOverview14.pdf

[16] Otter M., Elmqvist H., Mattsson S.E.: The New Modelica MultiBody Library, Modelica ’03, S. 311-330, http://www.modelica.org/Conference2003/papers/h37_Otter_multibody.pdf

[17] Pantelides C.: The Consistent Initialization of Differential-Algebraic Systems. SIAM Journal of Scientific and Statistical Computing 9 (1988) S. 213-231

[18] Pelchen C., Schweiger C., Otter M.: Modeling and Simulating the Efficiency of Gearboxes and of Planetary Gearboxes. Modelica ’02, S. 257-266. http://www.modelica.org/Conference2002/papers/p33_Pelchen.pdf

[19] Pfafferott T., Schmitz G.: Modelling and transient simulation of CO2-refrigeration systems with Modelica. International Journal of Refrigeration 27 (2004) 1 S. 42-52

[20] Reichl G.: WasteWater - a Library for Modeling and Simulation of Wastewater Treatment Plants in Modelica. Modelica ’03, S. 171-176, http://www.modelica.org/Conference2003/papers/h05_Reichl.pdf

[21] Tiller M., Bowles P., Elmqvist H., Brück D., Mattsson S. E., Möller A., Olsson H., Otter M.: Detailed Vehicle Powertrain Modeling in Modelica. Modelica ’00, S. 169-178 http://www.modelica.org/workshop2000/proceedings/Tiller.pdf

[22] Tiller M.: Introduction to Physical Modeling with Modelica. Kluwer Academic Publisher 2001, http://www.Dynasim.se/Tiller/BookInfoPage/

[23] Tummescheit H.: Design and Implementation of Object-Oriented Model Libraries Using Modelica, PhD thesis ISRN LUTFD2/TFRT--1063--SE, 2002, Lund, Schweden http://www.control.lth.se/articles/article.pike?action=fulltext&artkey=tum02dis