OPTIMIZATION WITH FMI AND CASADI: ANALYSIS IN INDUSTRIAL APPLICATIONS

Andreas Pfeiffer, Christoph Winter (DLR Institute of Vehicle Concepts)
Fabian Jarmolowitz, Christian Bertsch (Robert Bosch GmbH)

16th International Modelica & FMI Conference 2025 in Lucerne 10.09.2025

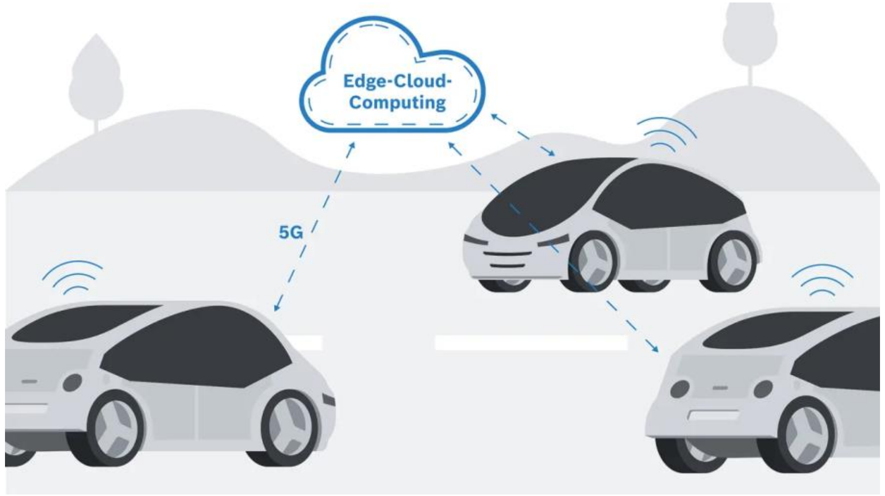
Funded by the European Union
NextGenerationEU

Supported by:
Federal Ministry for Economic Affairs and Energy
on the basis of a decision by the German Bundestag

BOSCH DLR

Motivation and Introduction

- Application at Bosch:
  - Outsource **computationally intensive functions** from car ECU to the (edge) cloud
  - Example: **Optimization** based control (e.g. nonlinear model predictive control NMPC)
- Study:
  - Test and analyze selected parts of **tool chain for optimal control**
  - **CasADi**:
    - Open source tool for nonlinear optimization and algorithmic differentiation (AD)
    - Generation of self-contained C code, interfaces to many numerical algorithms
    - Interface to **Python**, Matlab, C++ and **FMI**

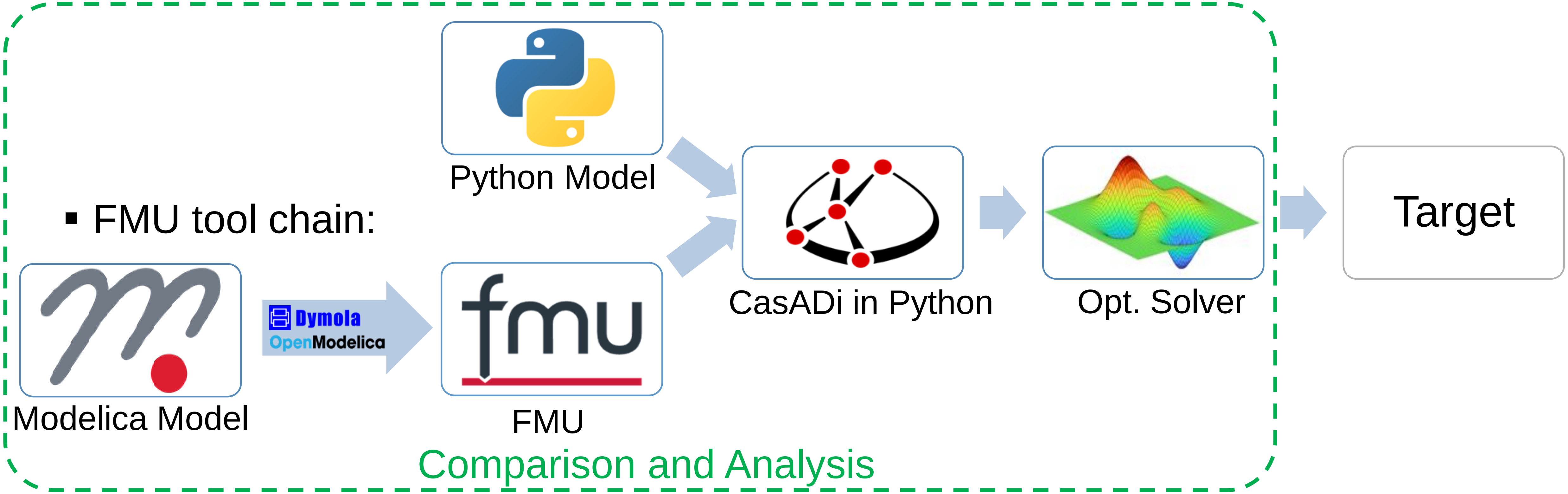


Source: https://www.bosch.com/research/news/ipcei-cis-project/

A. Pfeiffer, C. Winter, DLR FK | F. Jarmolowitz, C. Bertsch, Bosch

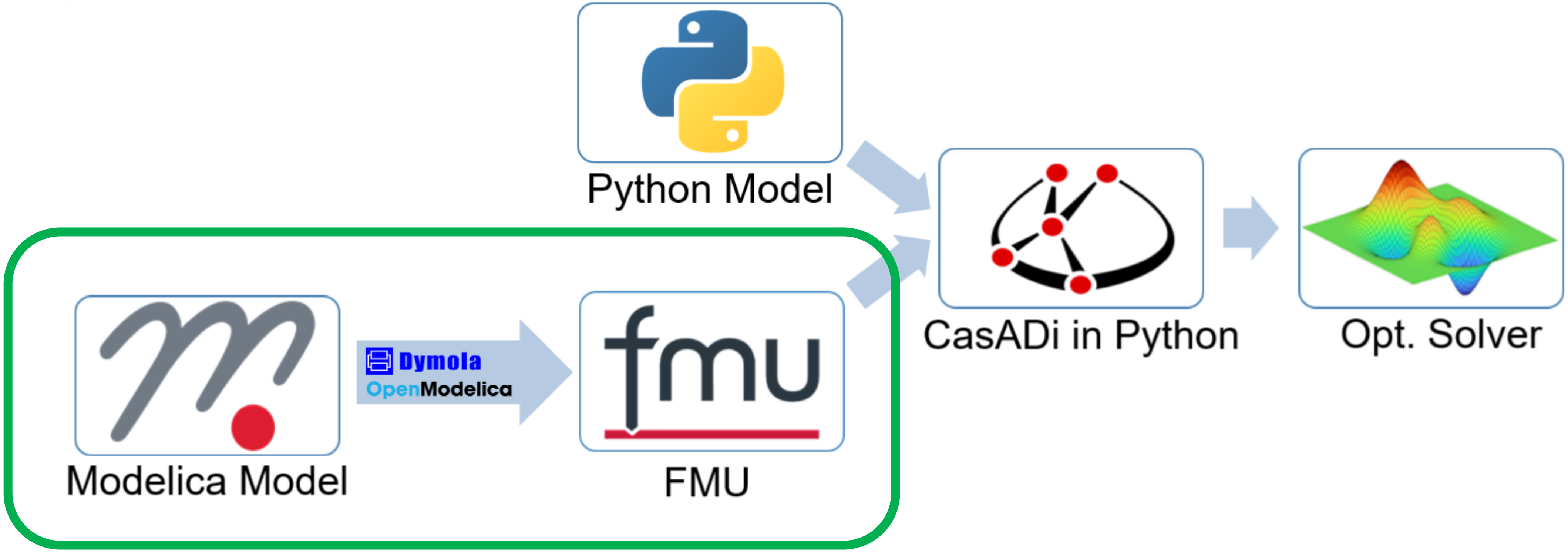
Framework of our study

- CasADi native tool chain:
- FMU tool chain:



A. Pfeiffer, C. Winter, DLR FK | F. Jarmolowitz, C. Bertsch, Bosch

Framework


Python Model
CasADi in Python
Opt. Solver
Modelica Model
Dymola
OpenModelica
fmu
FMU

FMU export from Modelica model, derivatives

FMI 2.0 for Model Exchange

- Assumption: continuously differentiable functions f, h (rough: smooth systems without events)

$$\dot{z} = f(z, u)$$
$$w = h(z, u)$$

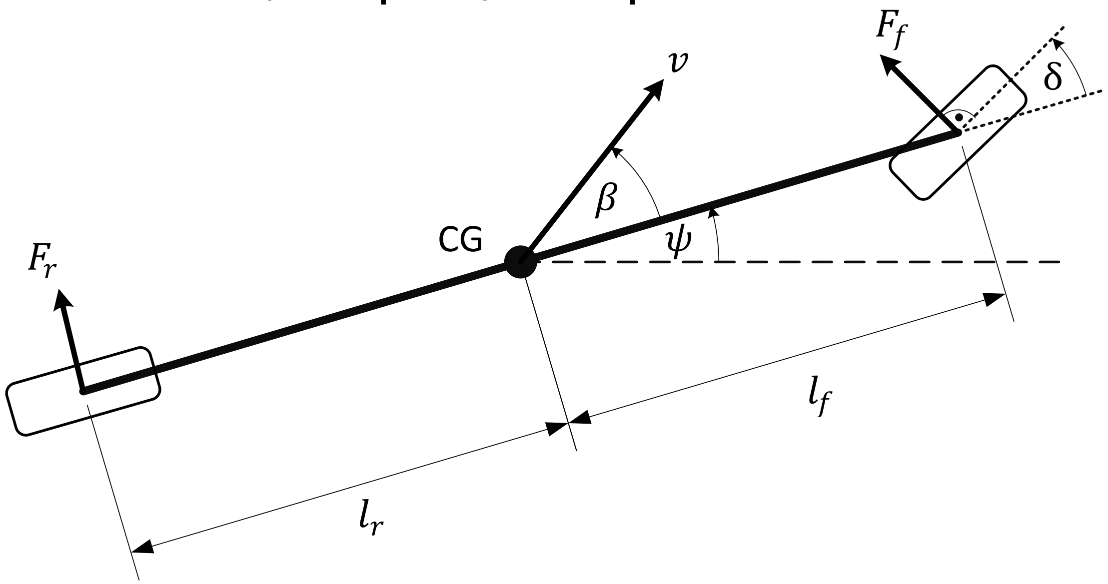
- Activate analytical derivatives $\frac{\partial f}{\partial u}, \frac{\partial f}{\partial z}, \frac{\partial h}{\partial u}, \frac{\partial h}{\partial z}$ in FMI export:
  - Dymola: set flag `Advanced.Translation.Generate.AnalyticJacobian=true`
  - OpenModelica: set compiler option `-d=-disableDirectionalDerivatives`
- Directional derivatives in FMU:
  - `fmi2GetDirectionalDerivative`
  - Structural dependency information in `modelDescription.xml`

A. Pfeiffer, C. Winter, DLR FK | F. Jarmolowitz, C. Bertsch, Bosch

Use Case – Single Track Model

Single Track Model

- Prediction model [Bünte et al. 2005]:
  - $\dot{z} = f(z, u)$
  - $a_y = h(z, u)$ with $z = (\beta, \delta, \omega, \psi, v, x, y)$, $u = (a_x, \lambda)$
  - 7 states, 2 inputs, 1 output

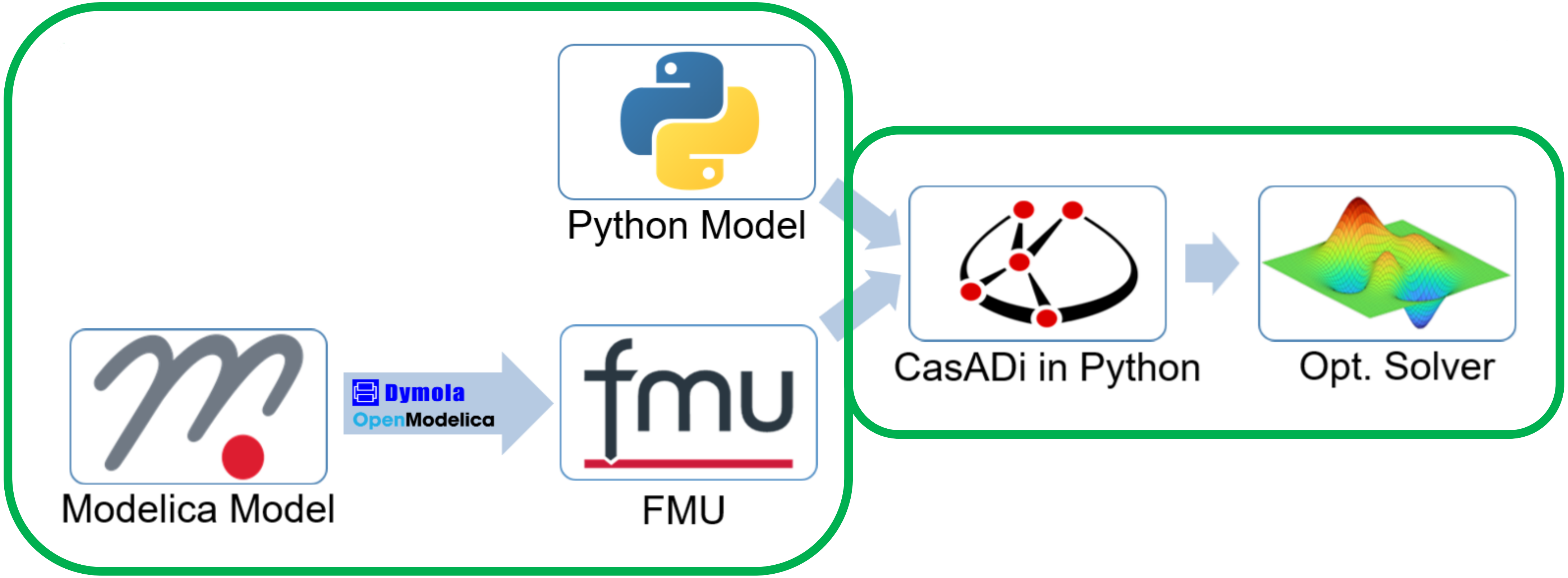


Sorted Modelica code

```
v_mod = (sign(v)*sqrt(v*v + 4*vmin*vmin) + v)/2;
F_f = mu*cf0*(delta - beta - lf/v_mod*omega);
F_r = mu*cr0*(        - beta + lr/v_mod*omega);
ay = 1/m*(F_f + F_r);
der(delta) = lambda;
der(v) = ax;
der(psi) = omega;
der(x) = v*cos(psi + beta);
der(y) = v*sin(psi + beta);
der(omega) = 1/J*(lf*F_f - lr*F_r);
der(beta) = ay/v_mod - omega;
```

Bünte, Tilman und Sahin, Akin und Bajcinca, Naim (2005) *Inversion of Vehicle Steering Dynamics with Modelica/Dymola.* In: Proceedings. 4th International Modelica Conference, 2005-03-07 - 2005-03-08, Hamburg.

Framework


Python Model
Dymola
OpenModelica
Modelica Model
fmu
FMU
CasADi in Python
Opt. Solver

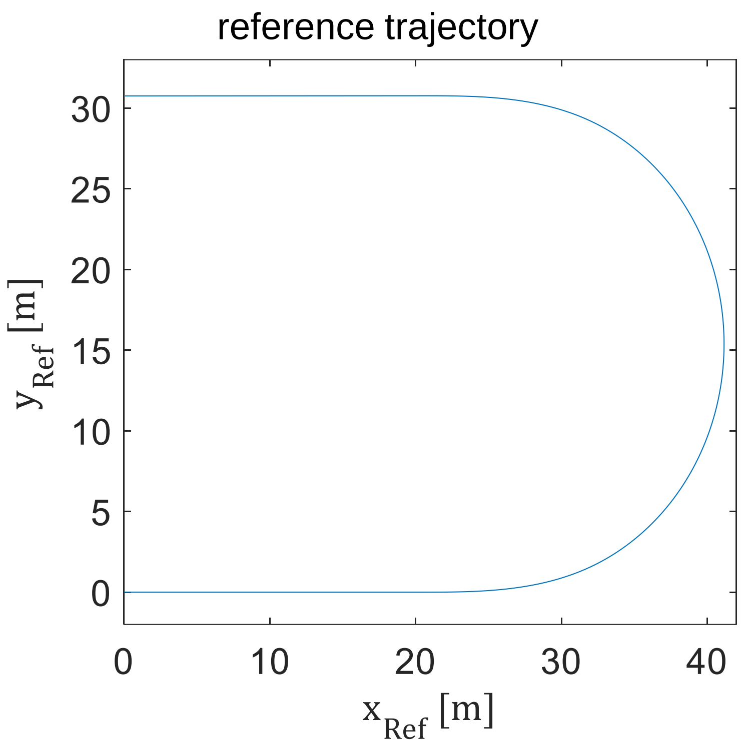
Use Case – Single Track Model

Problem

- Optimal control: Find inputs $u(t)$ to follow given trajectory
- Optimization problem:

$$\min_{u(t),z(t)} \frac{1}{N}\sum_{k=1}^{N}\left(x(t_k)-x_{\text{Ref}}(t_k)\right)^2+\left(y(t_k)-y_{\text{Ref}}(t_k)\right)^2$$

- Constraints:
  - Inputs: $\underline{u} \le u \le \overline{u},$
  - States: $\underline{\delta} \le \delta \le \overline{\delta},$ $\underline{v} \le v \le \overline{v},$
  - Algebraic: $\underline{a_y} \le a_y \le \overline{a}_y,$ $a_x{}^2 + a_y{}^2 \le \overline{K}$
  - Dynamics: $\boxed{\dot{z} = f(z,u)}$ → **FMU or Python**



- Discretization by **collocation** approach according to example code in CasADi

A. Pfeiffer, C. Winter, DLR FK | F. Jarmolowitz, C. Bertsch, Bosch

Use Case – Single Track Model

Collocation Approach in CasADi

$$\begin{aligned} \dot{z} &= f(z,u) \\ w &= h(z,u) \end{aligned}$$

- Approximate solution $z(t)$ with piecewise polynomials $p_k(t)$ on time intervals $[t_{k-1}, t_k]$ for $k = 1, \dots, N$
- **Collocation conditions** in interval $[t_{k-1}, t_k]$:

$$k \le \mathrm{N}: \quad \dot{p}_k(t_{k-1} + c_j \Delta t_k) = f\big(p_k(t_{k-1} + c_j \Delta t_k), u_k\big)$$

$$k < \mathrm{N}: \quad p_k(t_k) = p_{k+1}(t_k)$$

- Optimization variables:
  - u_k and $z_{kj} \coloneqq p_k(t_{k-1} + c_j \Delta t_k),\ j = 1, \dots, 4$
- Constraints of NLP:
  - Constraints of problem evaluated at t_k
  - Collocation conditions → (many) equality constraints

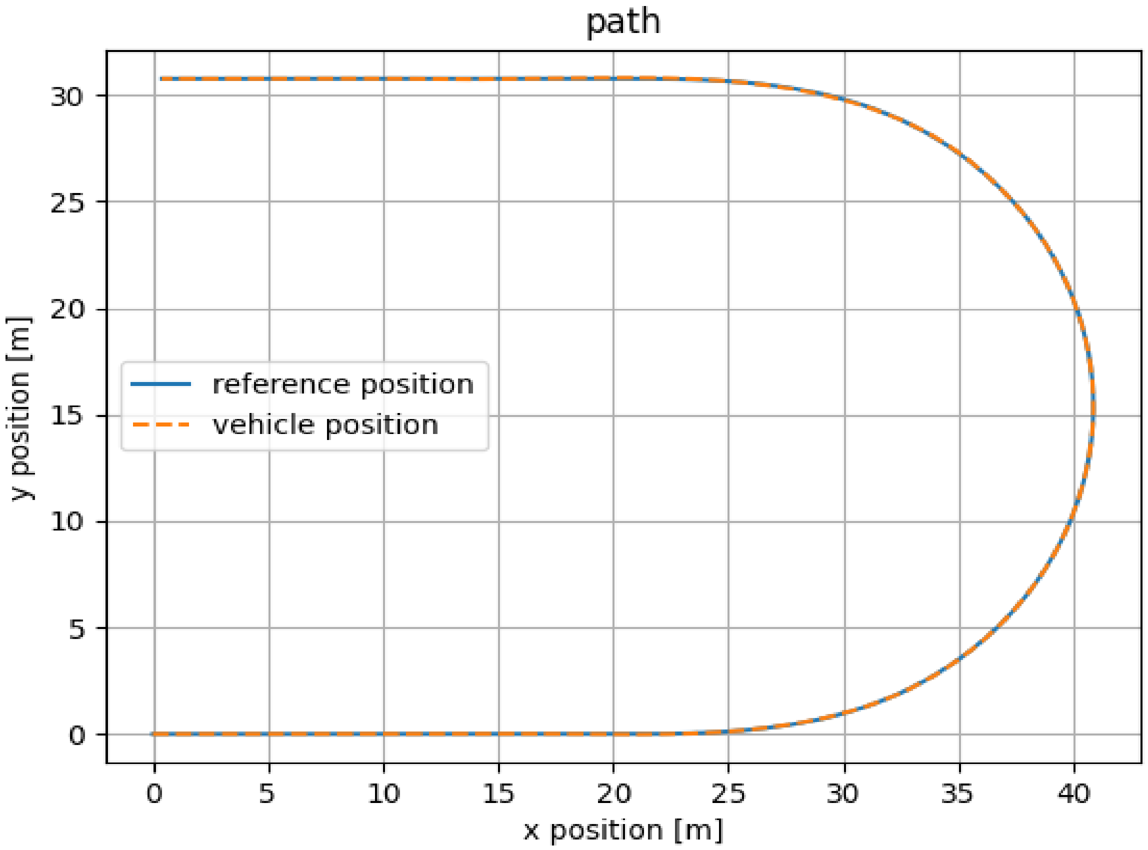
Python code (with FMI interface)

```
# Load FMU and get functions
dae = casadi.DaeBuilder('SingleTrack.fmu', …)
f = dae.create('f', ['x', 'u'], ['ode'])
h = dae.create('h', ['x', 'u'], ['ydef'])

# Define NLP problem [~75 lines of code]
zkj = casadi.MX.sym(…)
uk = casadi.MX.sym(…)
… f(zkj,uk) …
… h(zkj,uk) …
solver = casadi.nlpsol('solver', 'ipopt', …)

# Solve NLP problem by IPOPT and get solution
solution = solver(…)
```

Use Case – Single Track Model

Results

- Successfully following the reference trajectory:
  - max. position error is 5 cm
- Optimal control solution u:
  - fulfills all constraints ✓
  - stays within its limits ✓

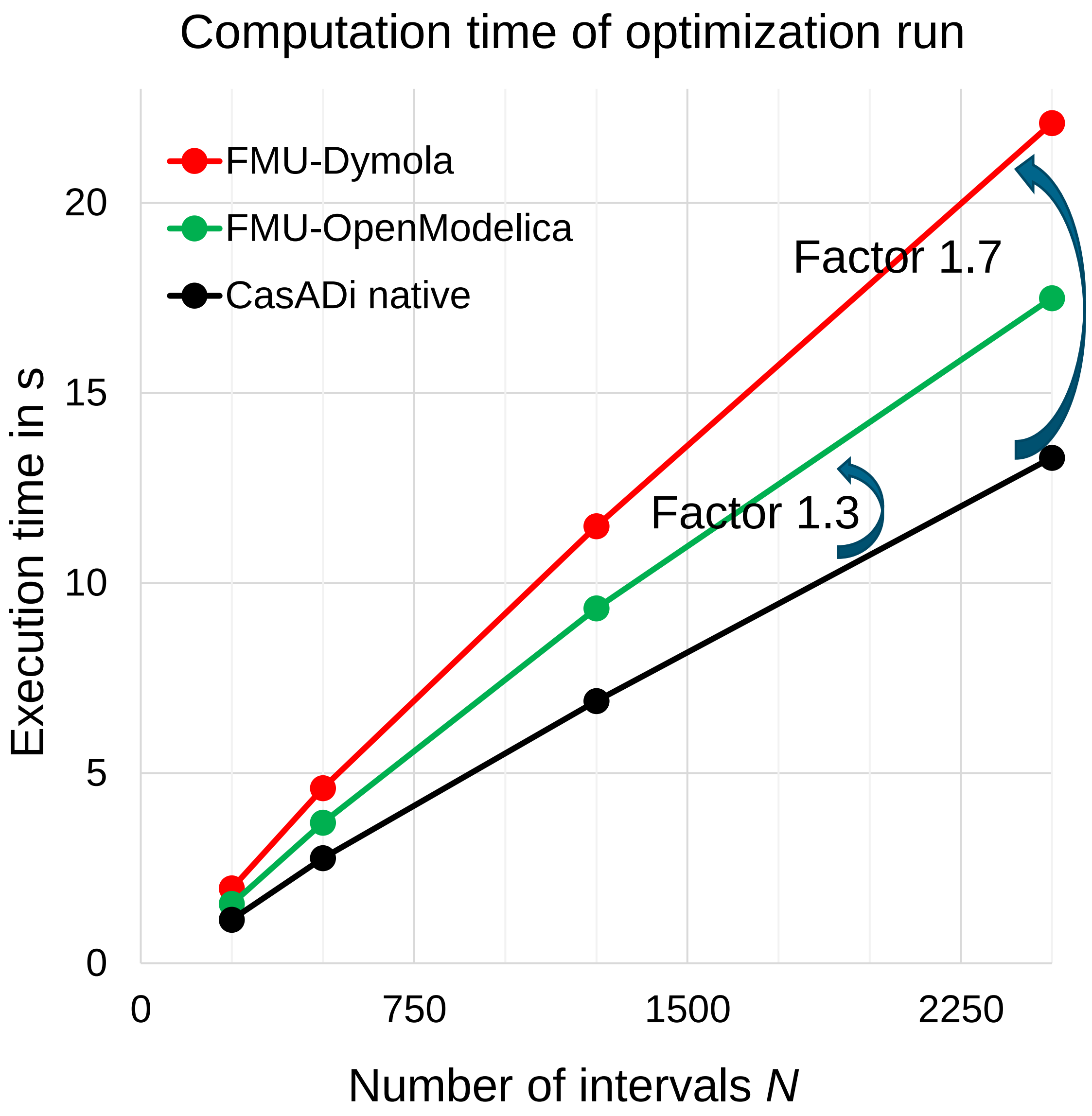


Use Case – Single Track Model

Computation Time

- Variation of number of intervals $N = 250, 500, 1250, 2500$:
  - 7500 to 75,000 optimization variables
  - 7500 to 75,000 constraints

- Solutions for fixed N and different FMUs/models very similar:
  - deviations $< 10^{-11}$ ✓



Use Case – Chiller subsystem

Problem: optimal control of multi chiller system

- Optimization problem:

$$\min_{u(t),z(t)} \sum_{k=1}^{N} P_{\mathrm{el}}(t_k)$$

- Constraints:

$$T_{\mathrm{supply}} \le T_{\mathrm{demand}}$$
$$T_{\mathrm{out},i} \le T_{\mathrm{in},i} \qquad (i = 1, \dots, 5)$$

- Prediction model:

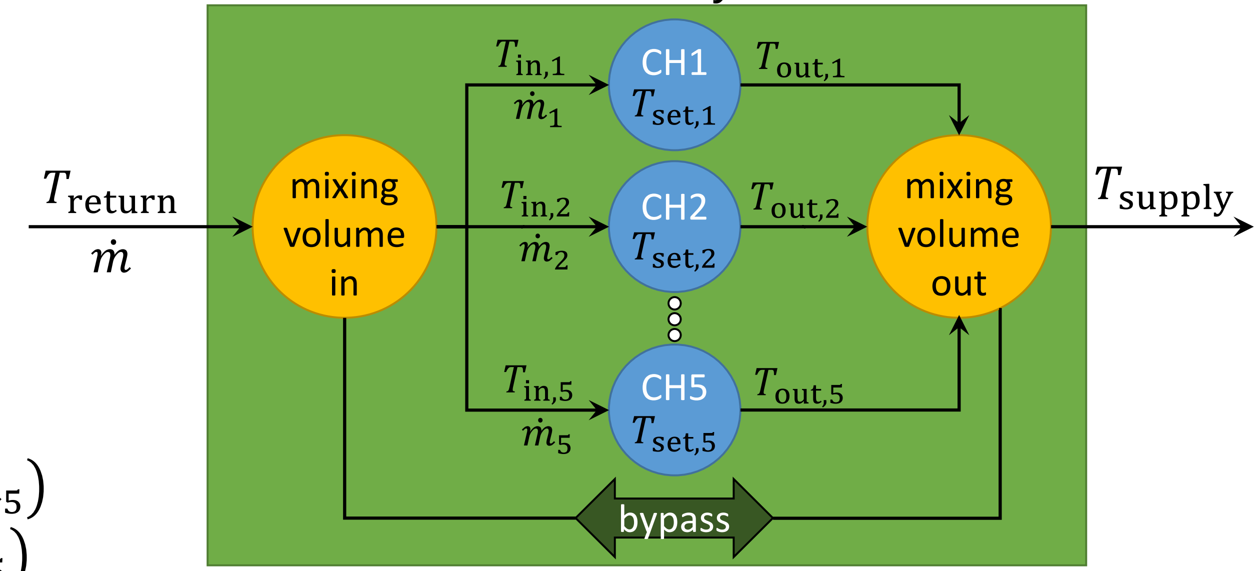
$$\boxed{\begin{aligned} \dot{z} &= f(z,u) \\ w &= h(z,u) \end{aligned}}$$

→ **FMU or Python**

  - 5 states $z = (T_{\mathrm{out},1}, \dots, T_{\mathrm{out},5})$
  - 10 inputs $u = (T_{\mathrm{set},1}, \dots, T_{\mathrm{set},5},\ \dot{m}_1, \dots, \dot{m}_5)$
  - 7 outputs $w = (P_{\mathrm{el}}, T_{\mathrm{supply}},\ T_{\mathrm{in},1}, \dots, T_{\mathrm{in},5})$

Symbol picture of building with multiple chillers, Bosch plant Bamberg

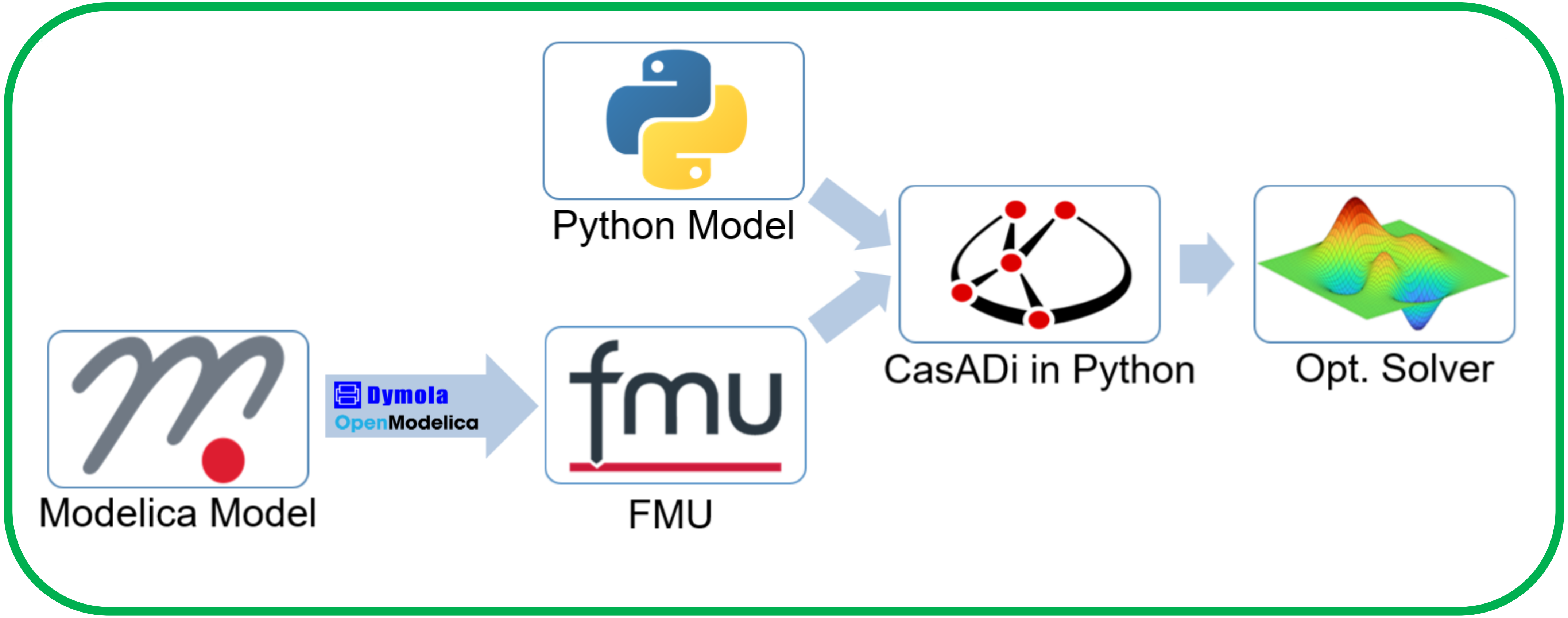
Chiller subsystem



Python: 250 lines of code

A. Pfeiffer, C. Winter, DLR FK | F. Jarmolowitz, C. Bertsch, Bosch

Framework


Python Model
CasADi in Python
Opt. Solver
Modelica Model
Dymola
OpenModelica
fmu
FMU

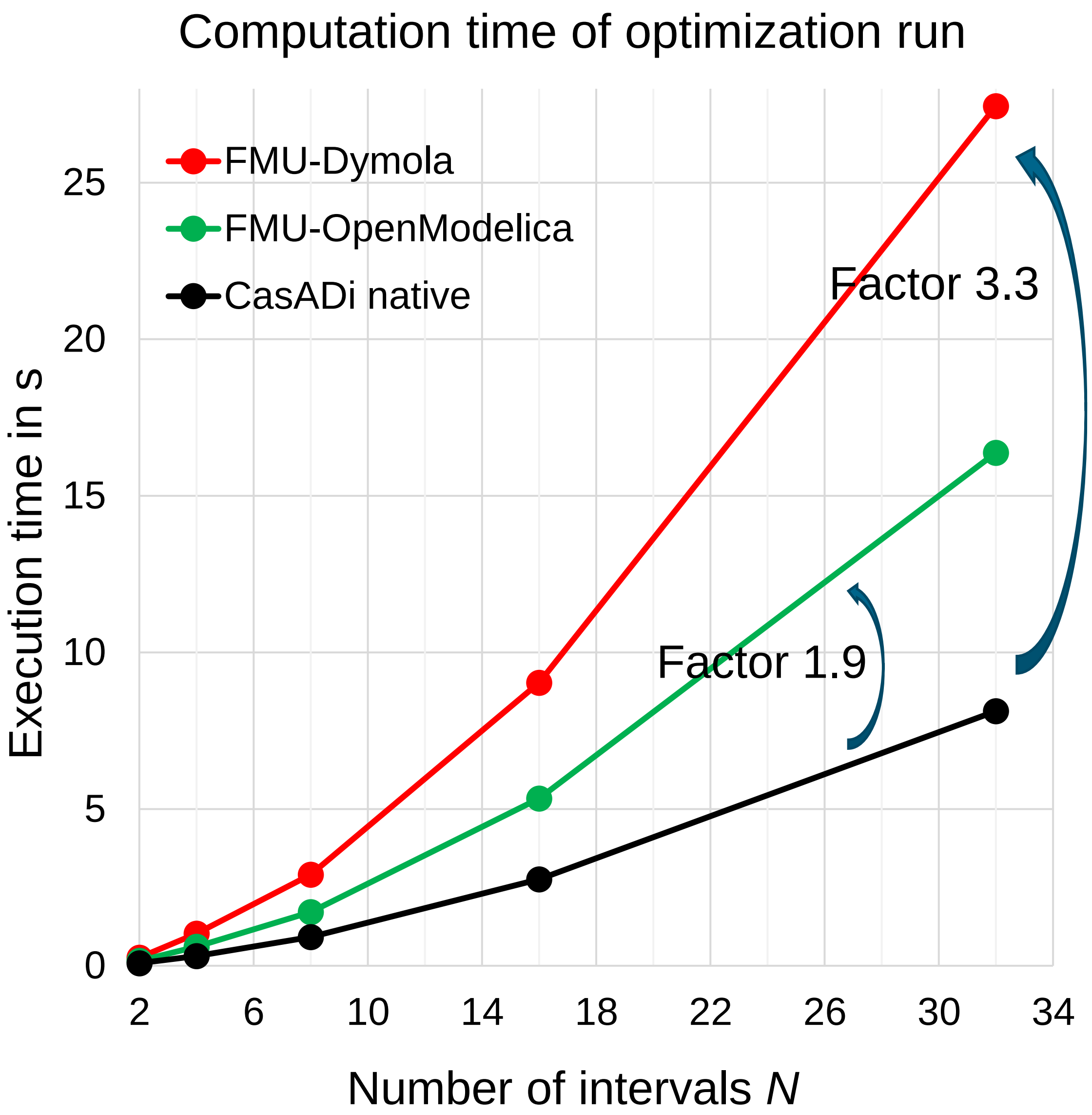
Use Case – Chiller subsystem

Results

Collocation variant:

- Variation of number of prediction intervals ($N = 2, 4, 8, 16, 32$)
- Up to 3500 optimization variables
- Up to 3400 constraints

- For fixed N:
  - Prediction model from different sources
  - Solution deviations < 10^{-10} ✓



Split of computational time

Detailed Time Analysis

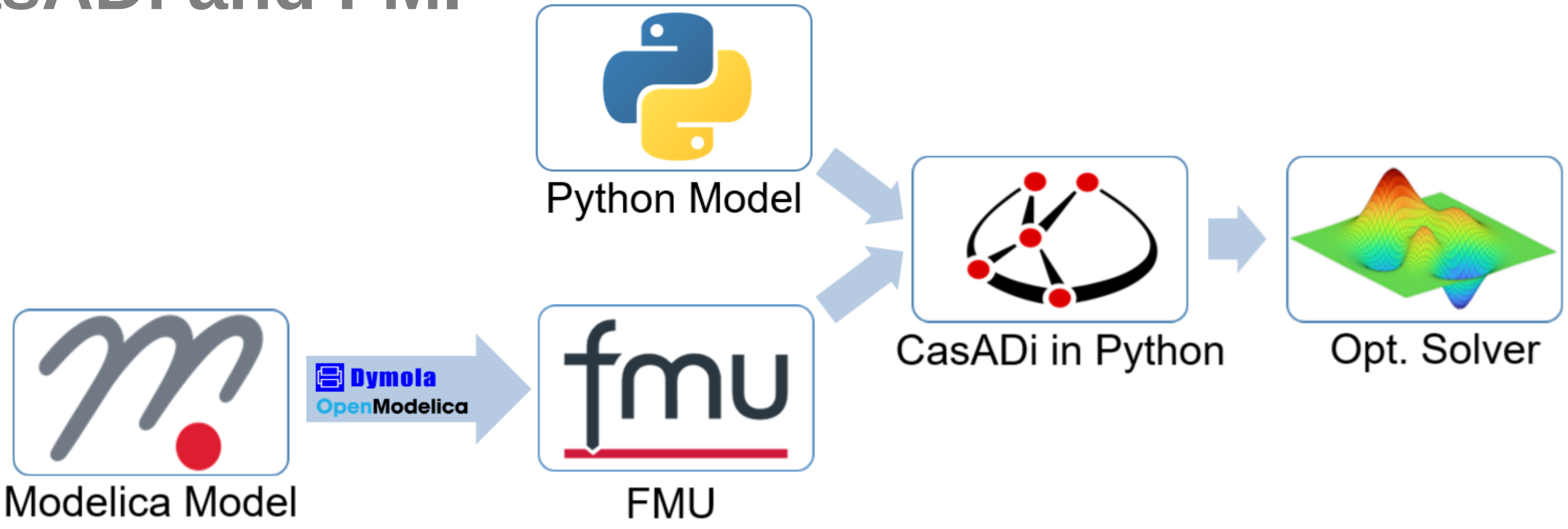
$$\min_x J(x)$$
$$\text{s.t.} \quad g(\text{x}) \leq 0$$

	Time factor: FMU-based vs. CasADi native	
▪ Objective J, constraints g	~1	
▪ First derivatives $\frac{\partial J}{\partial x}, \frac{\partial g}{\partial x}$	~1 – 2.5	
▪ Second derivatives $\frac{\partial^2 J}{\partial x^2}, \frac{\partial^2 g}{\partial x^2}$	3 – 4.5	Single Track (collocation, N = 2500)
	5 – 10	Chiller (collocation, N = 16)
	25	Chiller (single shooting, N = 16)
▪ Optimization algorithm	~1	

→ Second derivatives with FMUs: 50% – 98% of total time

Optimization with CasADi and FMI

Summary



- Use cases:
  - Trajectory following of a single track model
  - Chiller subsystem
- Main results:
  - Main bottleneck: second-order derivatives for IPOPT
  - CasADi native faster with factor 1.5 – 3 vs. CasADi + FMI (collocation)
  - Advantage: use **existing model libraries** before FMU export
- Outlook:
  - Continue the approach using more complex Modelica models in the building domain

Optimization with CasADi and FMI

Summary

- Acknowledgments:
  - Tilman Bünte, DLR Institute of Vehicle Concepts
  - Joel Andersson, FMIOPT AS
  - Financial support by the German Federal Ministry of Economic Affairs & Climate Action (grant number 13IPCEI021)