University of South-Eastern Norway

Advanced Control Implementations with Modelica

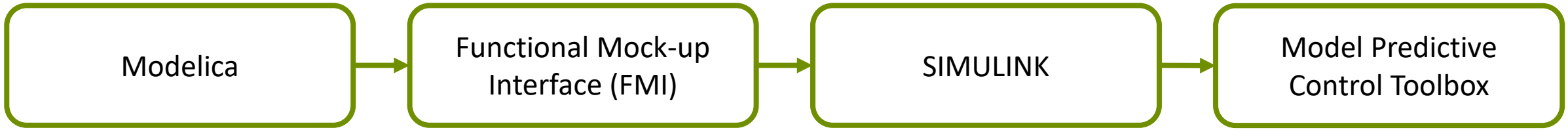
FMH606 Master's Thesis 2023
Industrial IT and Automation
Carl Magnus Bøe

Supervisor USN: Finn Haugen
Supervisor Yara: Anushka Perera

Repository: https://github.com/cmbNor/Advanced-Control-Implementation-with-Modelica

Introduction

Background information



- Open-source equation-based modelling language
- Graphical and text based
- Multi domain support
- Standard library
  - 1600 components
  - 1350 distinct function

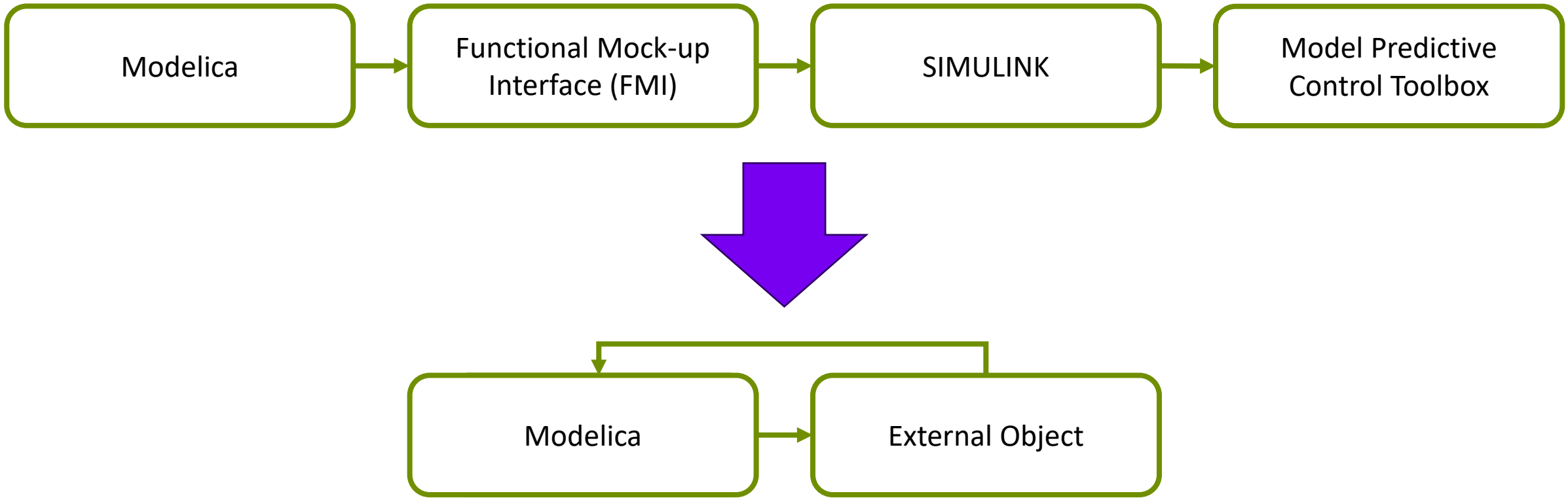
- Vendor neutral standard for model exchange
- Defines a container and an interface
- Open standard

- Commercial tool from MathWorks
- Graphical extension of MATLAB
- Used to simulate and analyze dynamic models
- Block based

- Commercial toolbox for SIMULINK
- Example code
- SIMULINK blocks

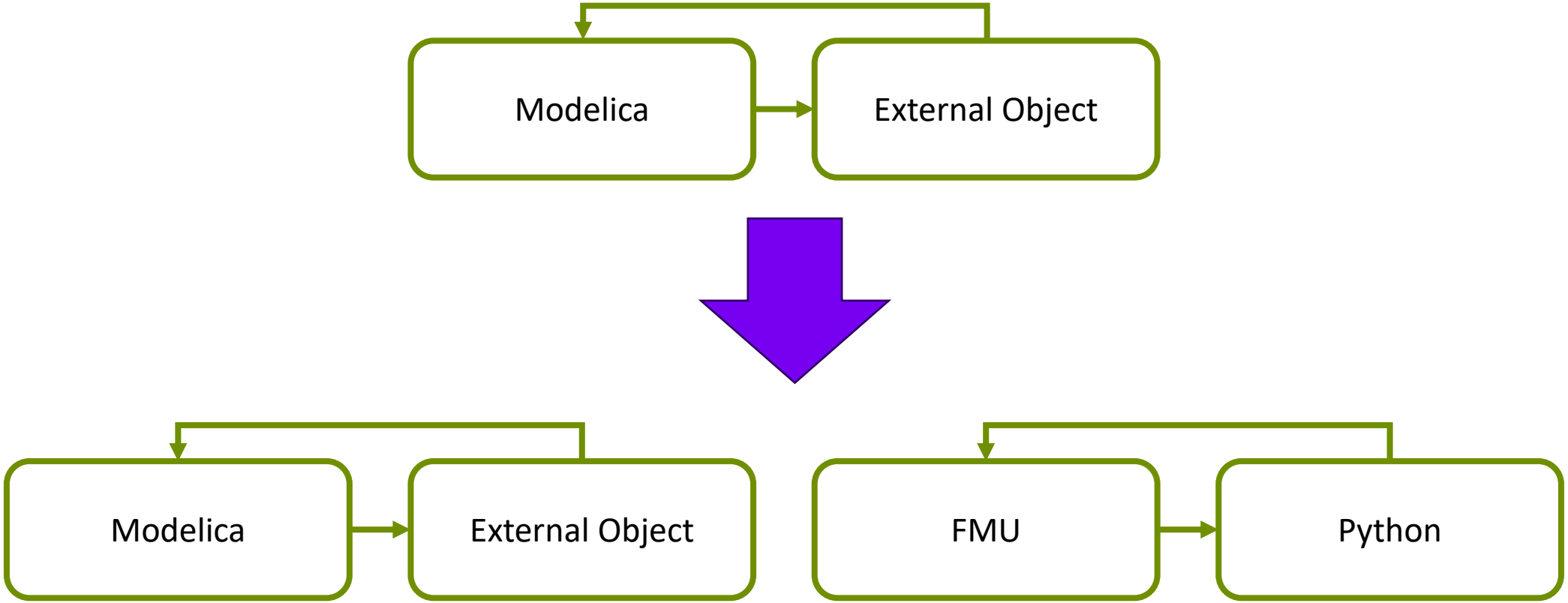
Introduction

Background information



Introduction

Revised scope



Existing Options

Linear MPC Modelica Library

- Open-source
- Last updated 2016
- Not compatible with Modelica 4.0

JModelica

- Discontinued in 2019
- Assimulo
- PyFMI
- FMI Library

FMPy - Dassault Systèmes

- Free Python FMI library
- Graphical user interface
  - Validate FMUs
  - Export to Jupyter Notebook, Modelica and Cmake
  - Web App

NLopt

- Open-source optimization library
- C, C++, MATLAB, Fortran, Python among others
- Collection of several algorithms from different contributors

Air Heater Model

First-principles, transfer function, and state space models

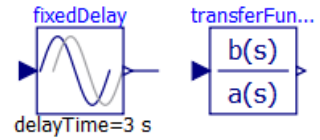
$$t_{const} \cdot \frac{dT}{dt} = (T_{amb} - T) + K_h \cdot u(t - t_{delay})$$

First-principles:

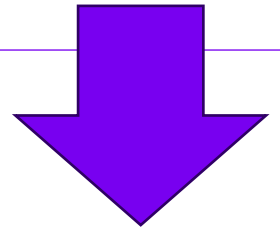
$$t_{CONST} \cdot der(T_{Out}) = (T_{amb} - T_{Out}) + Kh \cdot delay(u, 3)$$

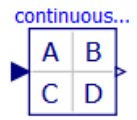
Transfer Function:

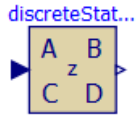
$$H(s) = \frac{K}{Ts+1} \cdot e^{-\tau \cdot s} \longrightarrow H(s) = \frac{3.5}{23s+1} \cdot e^{-3 \cdot s}$$



State Space:

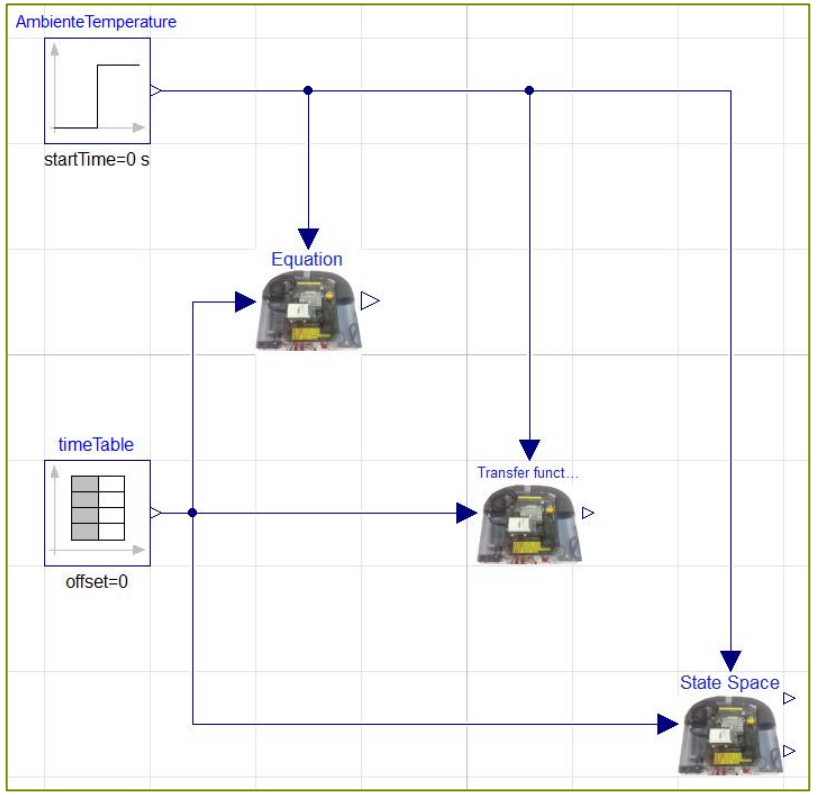






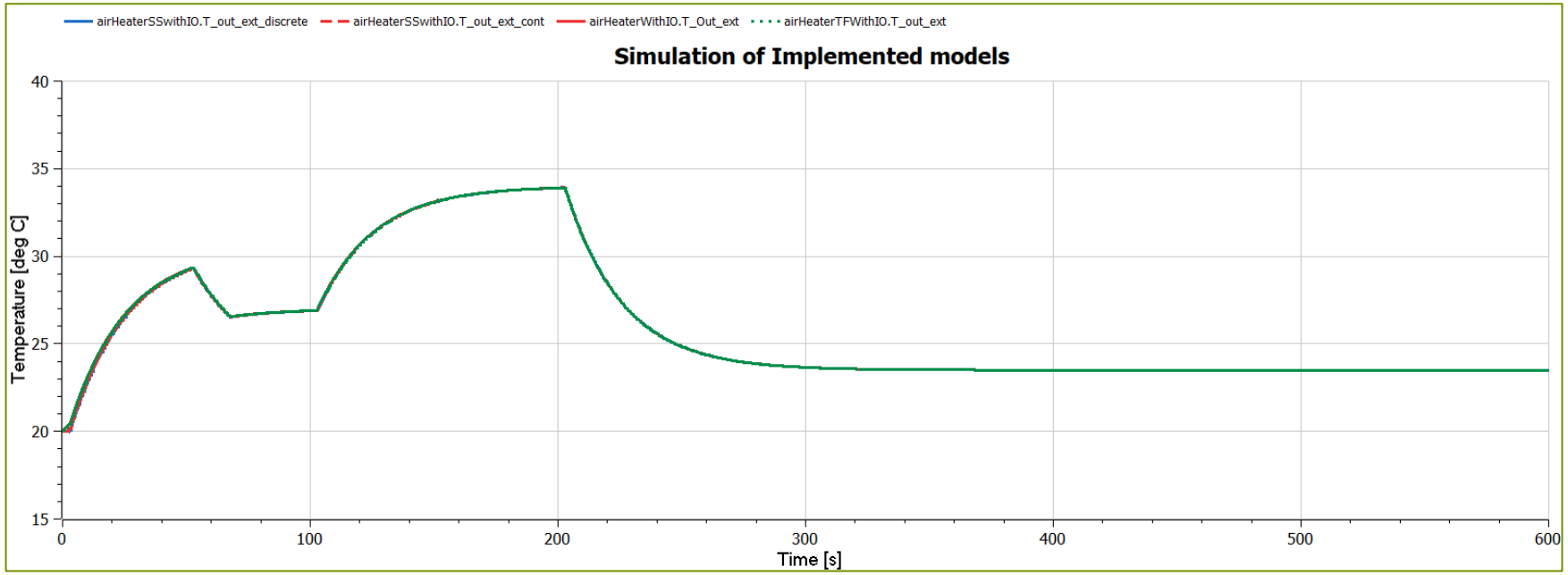
Air Heater Model

Simulation of first-principle, transfer function, and state space models



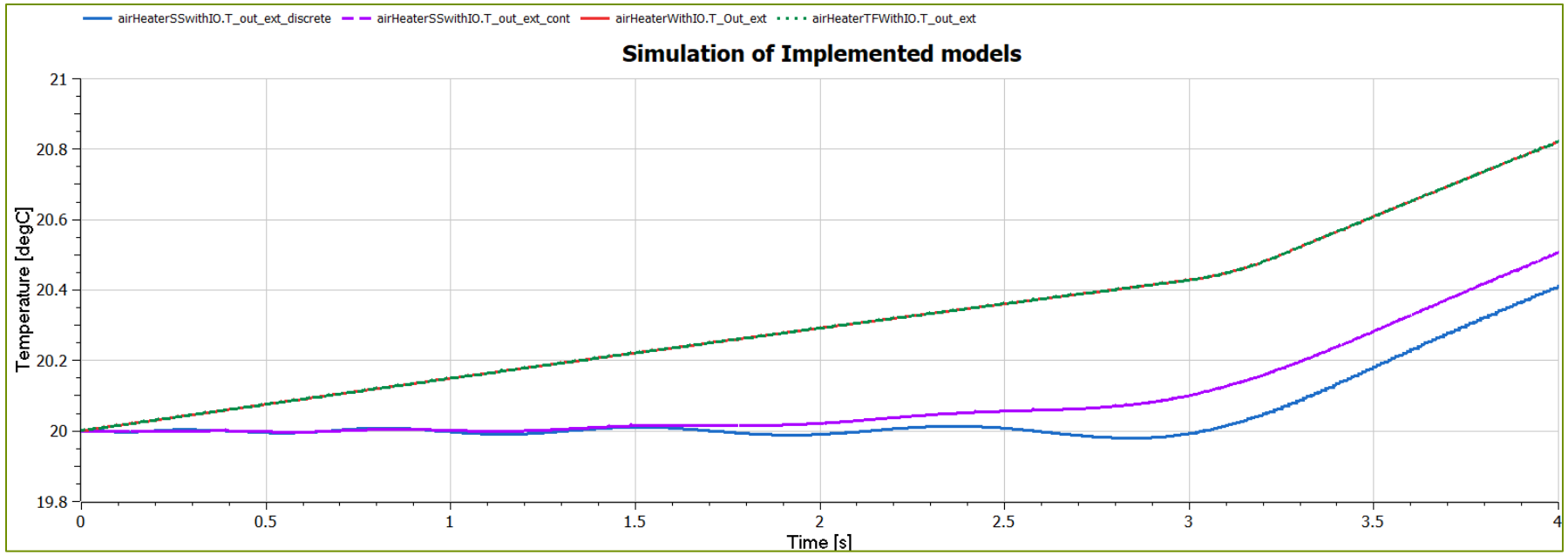
Air Heater Model

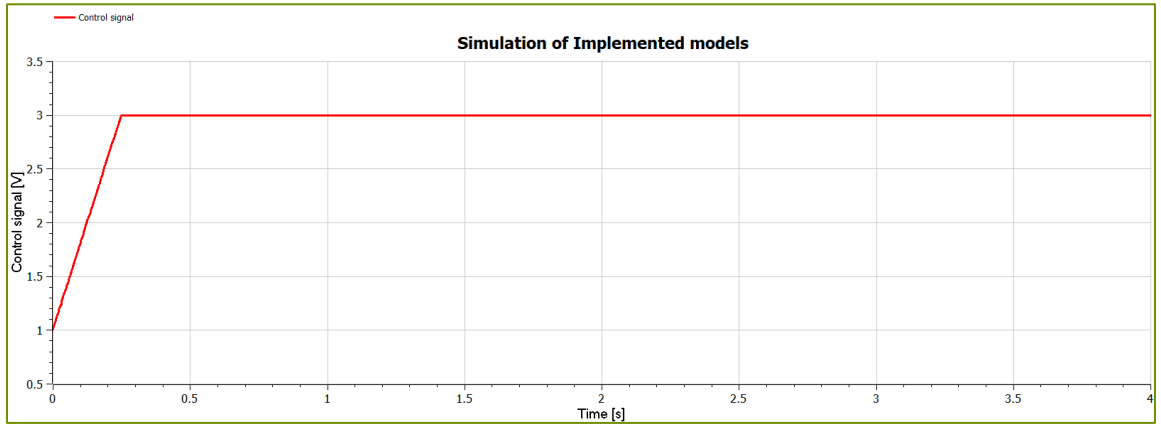
Simulation results for first-principle, transfer function, and state space models



Air Heater Model

Simulation results for first-principle, transfer function, and state space models





University of South-Eastern Norway

Calling External C Code in Modelica

Overview

Modelica Function

- Basic functions with one return variable
- Needs a Modelica block or model as a wrapper

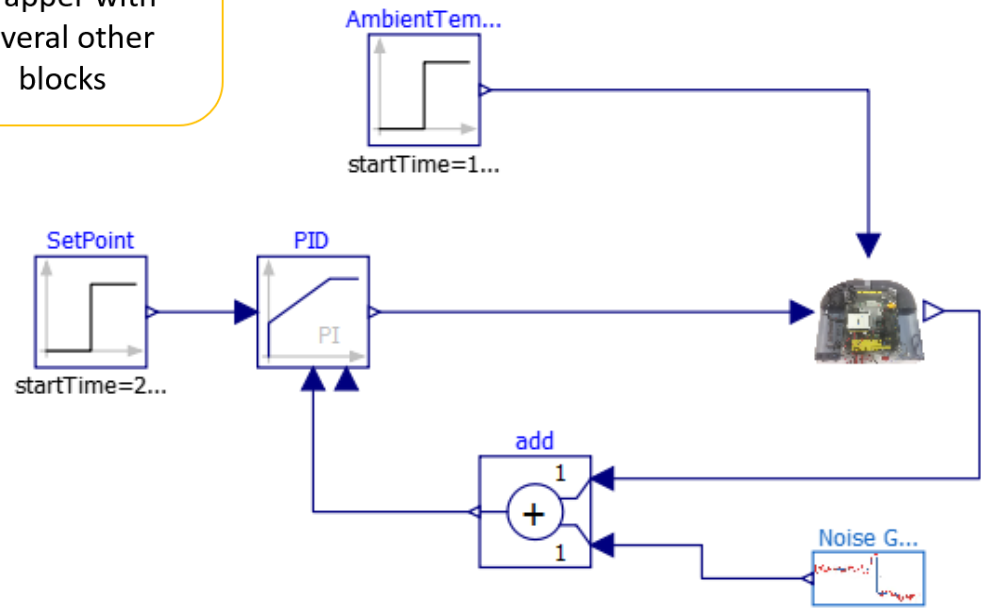
Modelica External Object

- Extended functionality and complexity
- Enables reading and writing of variables from external memory across C and Modelica

Calling External C Code in Modelica

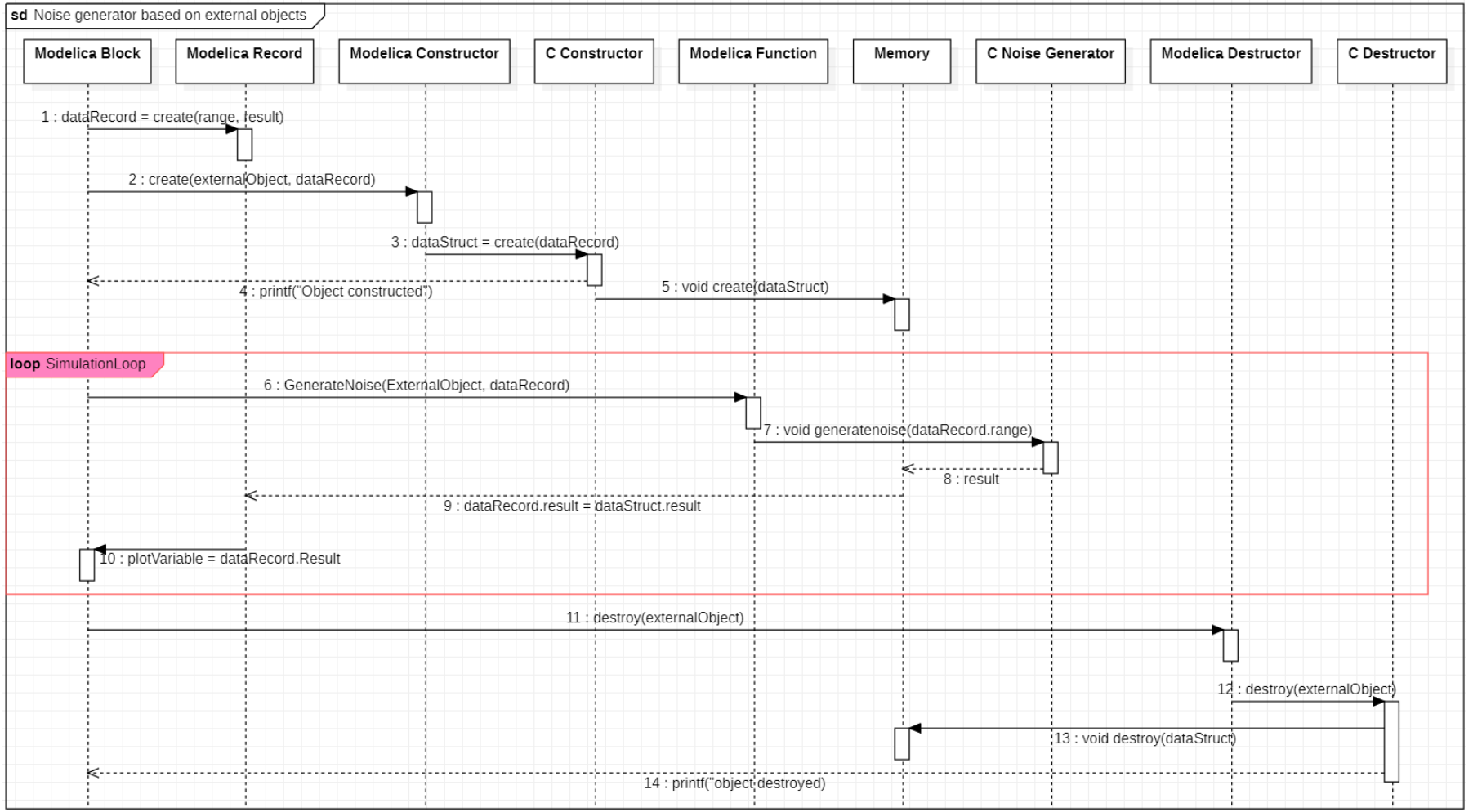
Implementation with the use of Modelica function

External C Function	Modelica Function	Modelica Model	Modelica Simulation Model
External file that is based on C code and is stored in the Modelica folder structure	Function in Modelica that calls the external C Function	A wrapper for the Modelica function to generate a block with inputs and outputs	Model that uses the Modelica wrapper with several other blocks



Calling External C Code in Modelica

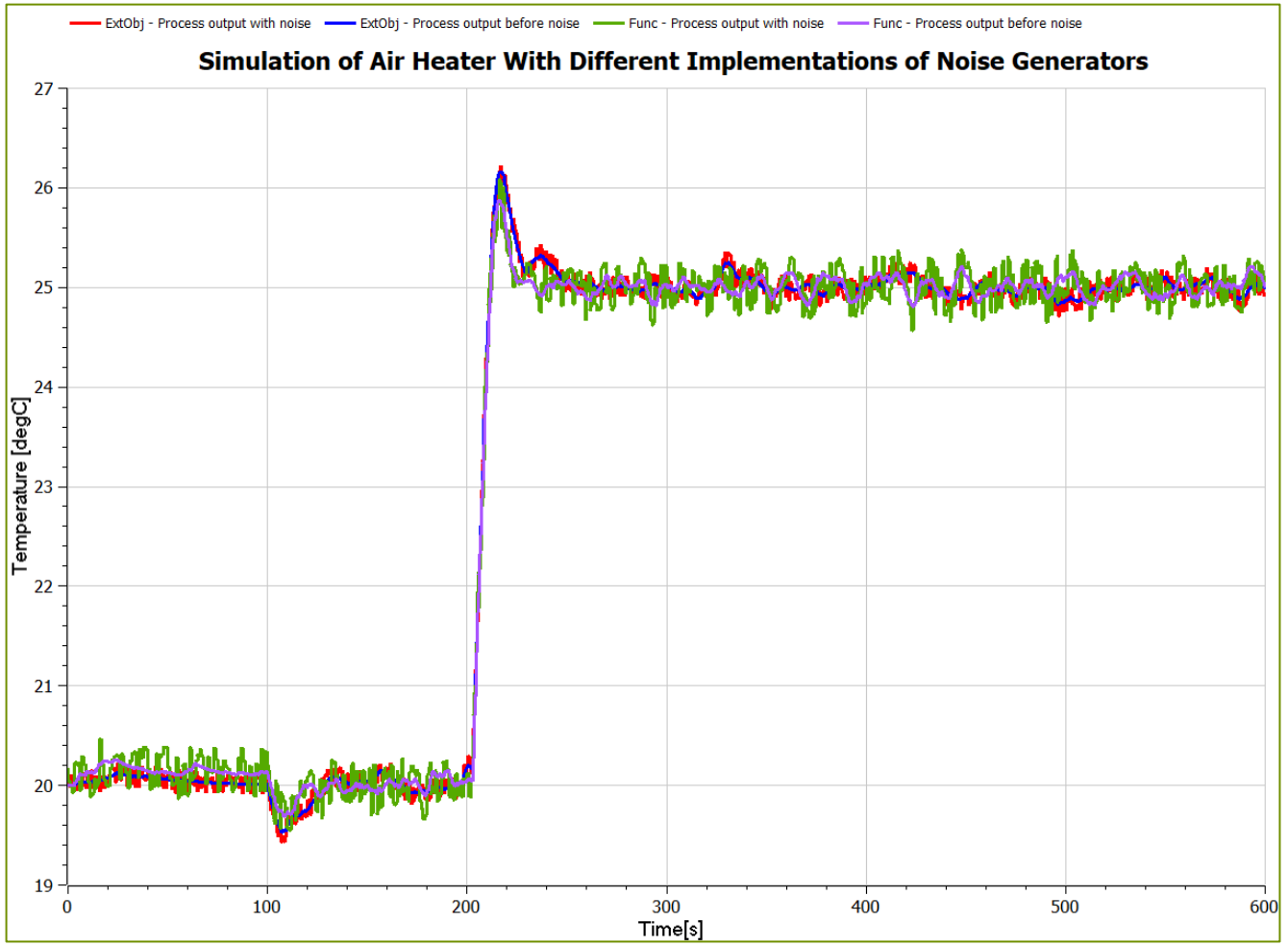
Implementation with the use of external objects



* Error in interaction diagram is corrected in this version

Calling External C Code in Modelica

Results



NLopt

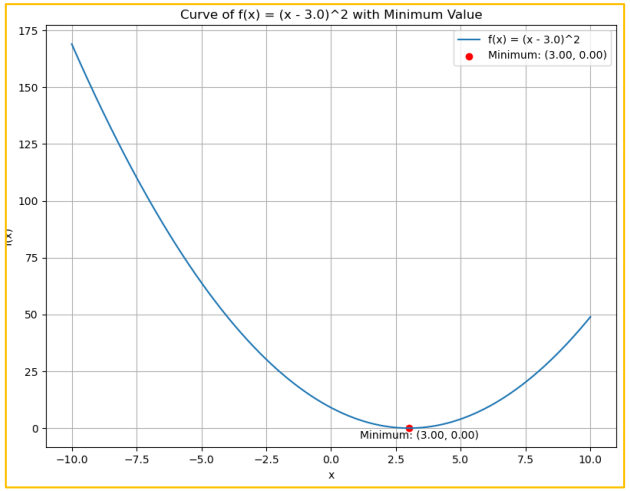
Overview

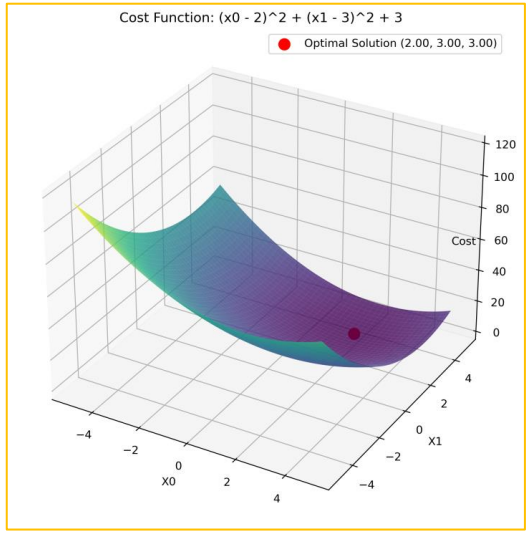
Basic example for calling external C functions in NLopt from OpenModelica and returning values to OpenModelica

Examples of solving a univariate optimization problem in OpenModelica by calling a NLopt function in C

Examples of solving a multivariate optimization problem in OpenModelica by calling a NLopt function in C

NLopt Version

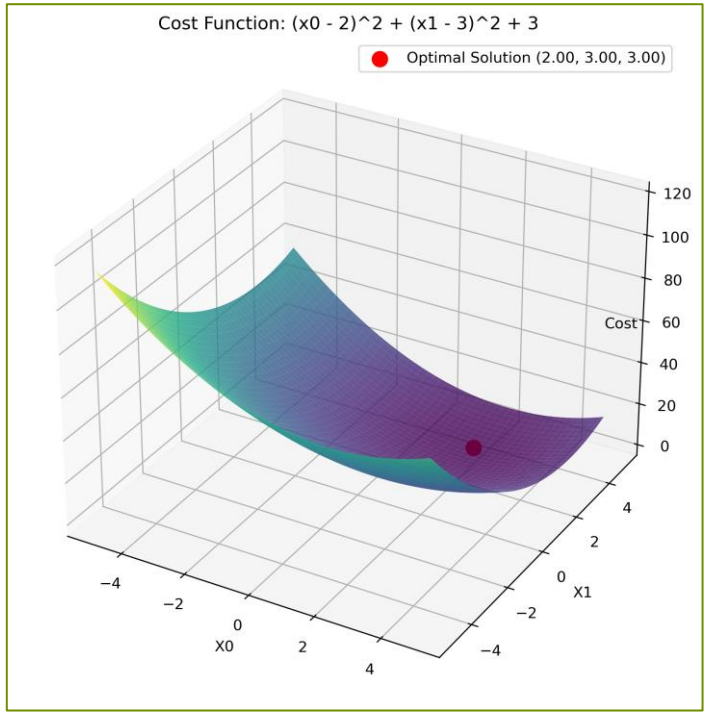




NLopt

Multivariate optimization

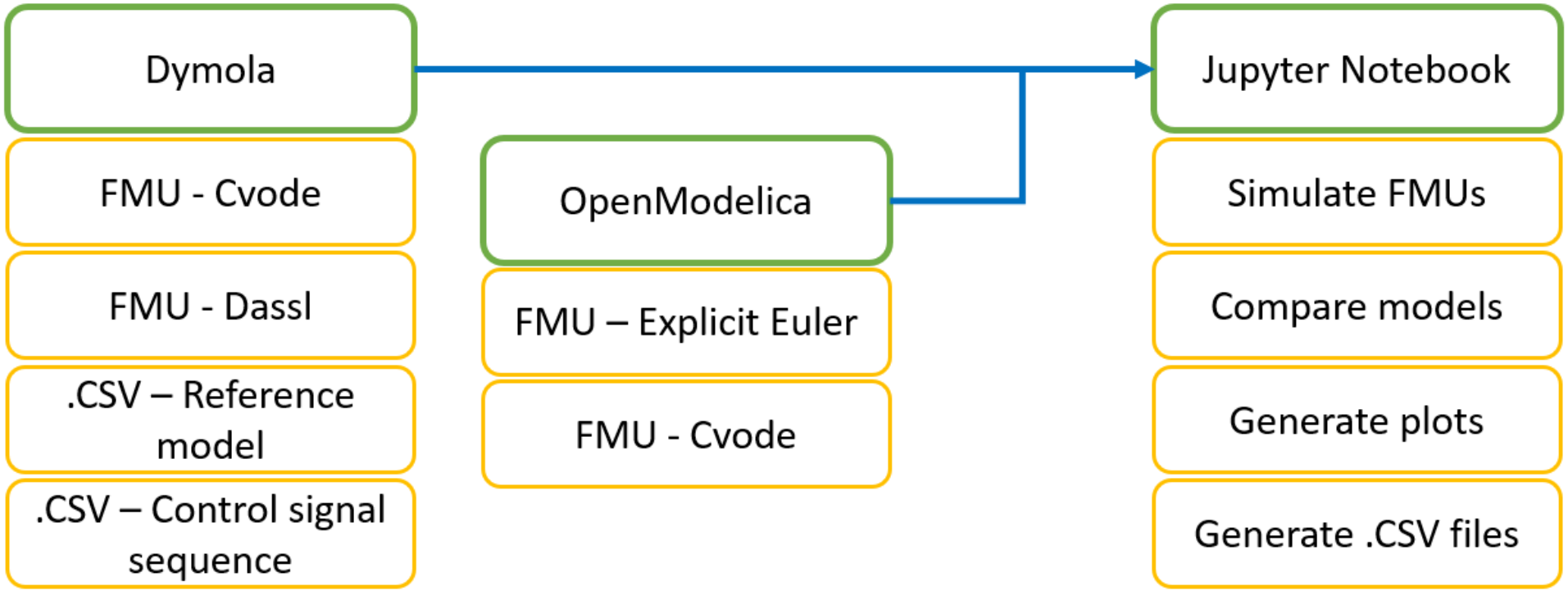
$$f(x) = (x_0 - 2)^2 + (x_1 - 3)^2 + 3$$



Variables	Value	Description
(Active) NloptMultiOptiBlock		
Tol	1e-06	Optimizer termination tolerance
max_iter	101	Maximum number of iterations for the optimizer
n	2	Number of optimization variables
optimizeData		
summary		
min_cost	3	Minimum value of the objective function after optimization
x1	1.99992	Optimization variable
x2	2.9999	Optimization variable
x1Lb	-5.0	Lower bound of x1
x1Ub	5.0	Upper bound of x1
x2Lb	-5.0	Lower bound of x2
x2Ub	5.0	Upper bound of x2

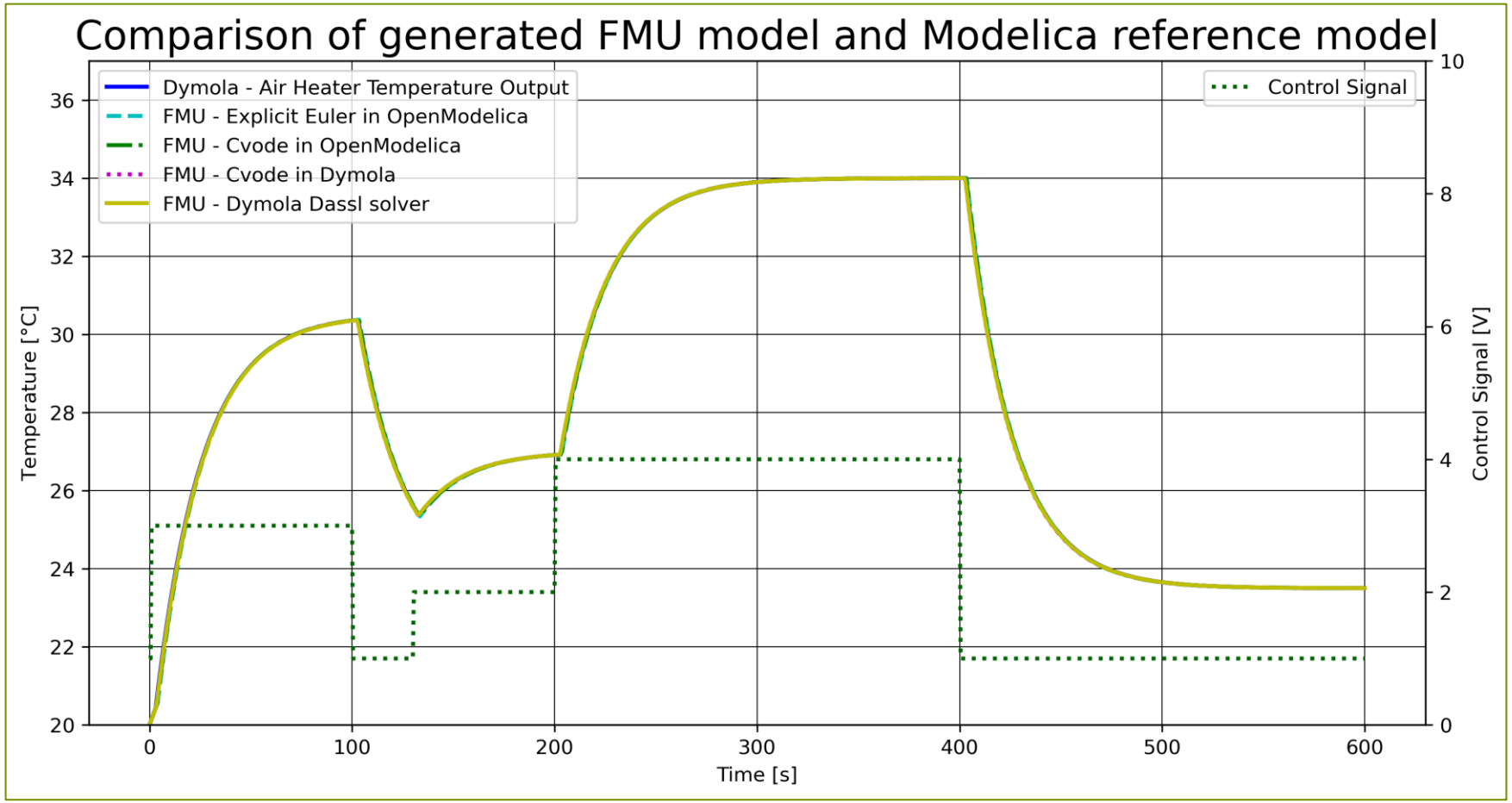
FMU of Air Heater in Python

Overview



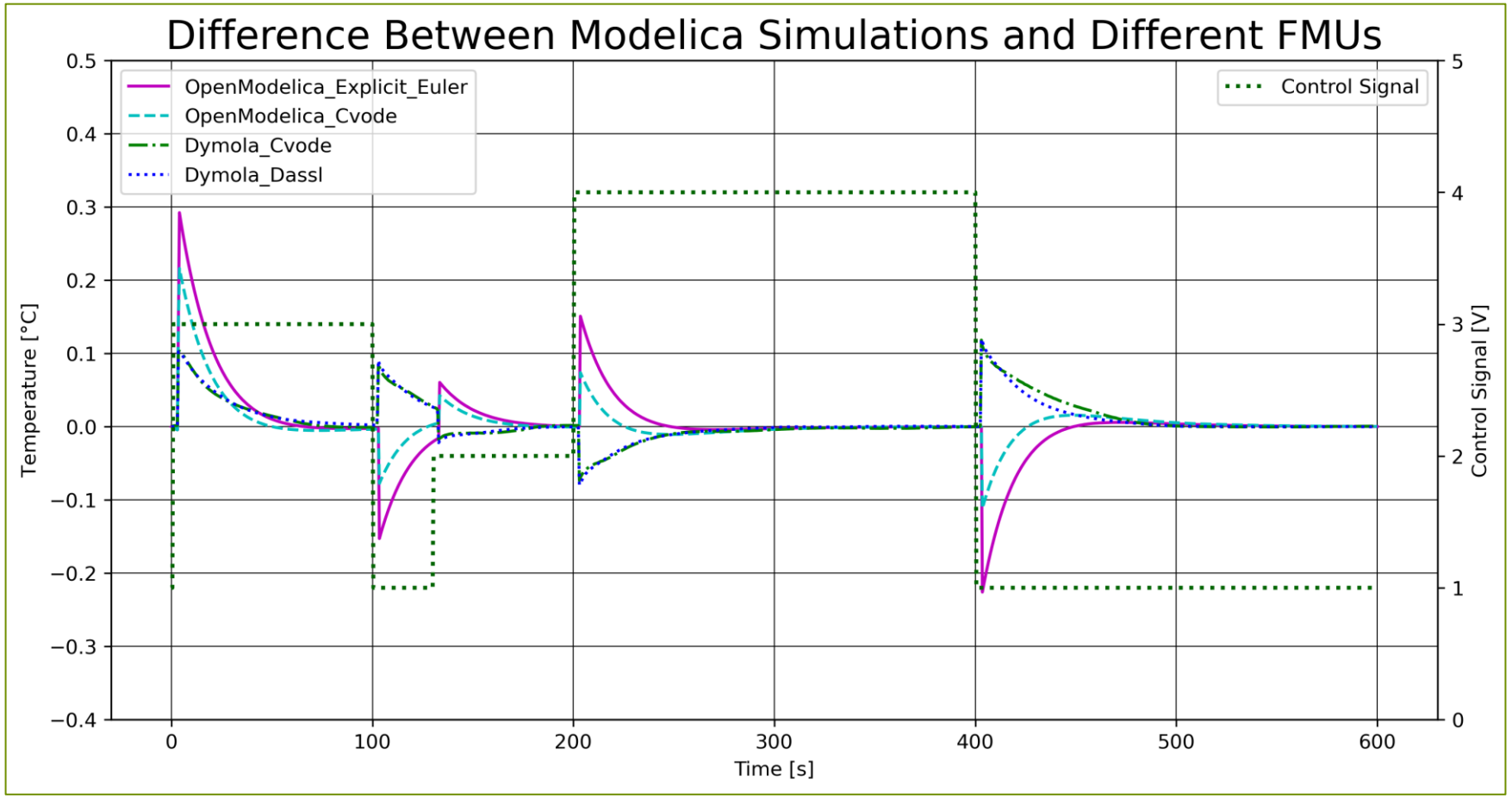
FMU of Air Heater in Python

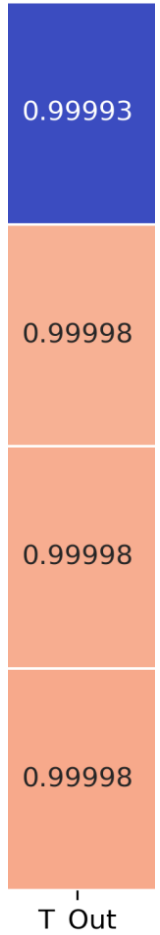
Results



FMU of Air Heater in Python

Results





FMU of Air Heater with MPC in Python

Overview

General

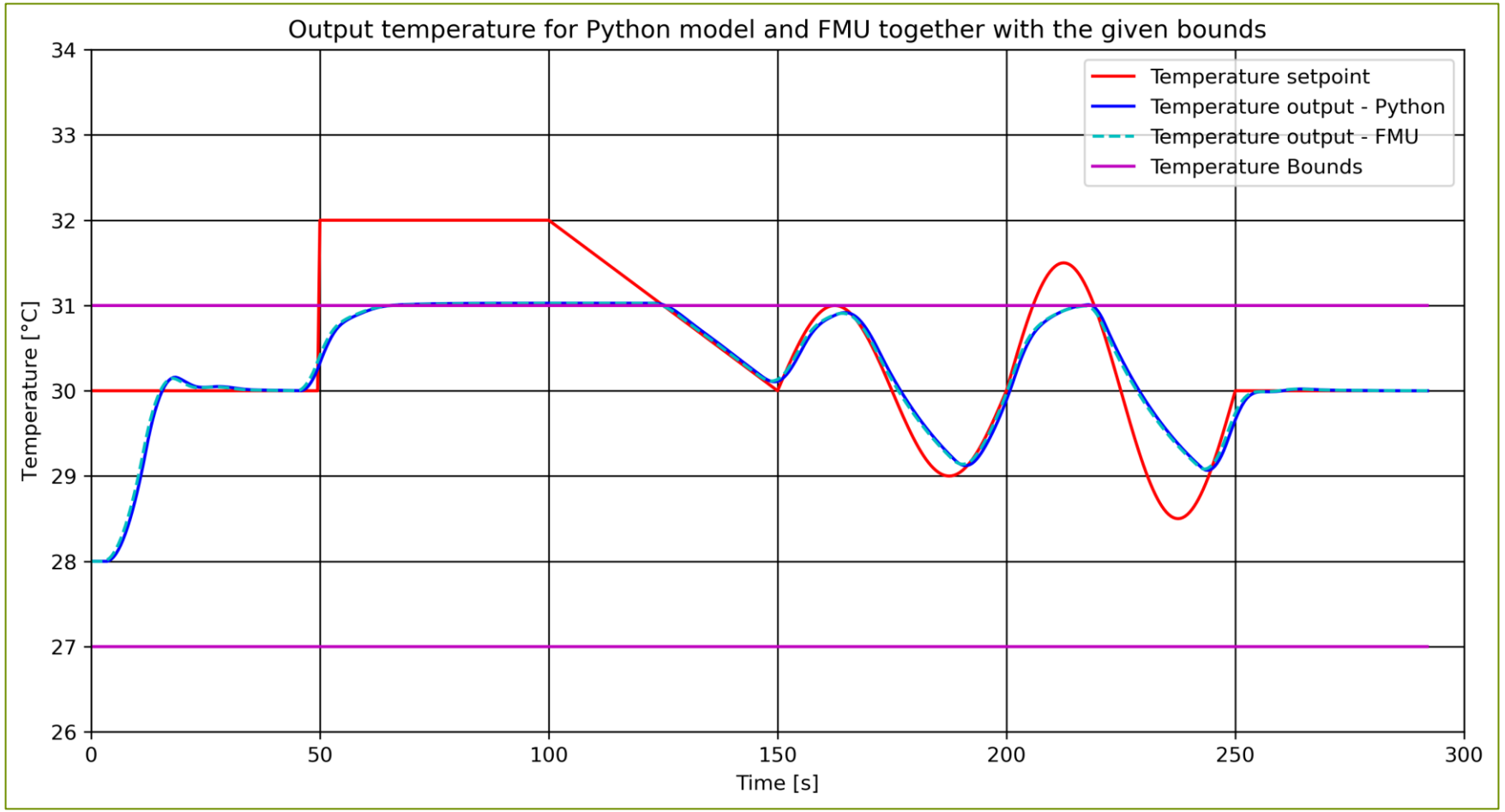
- Based on Python MPC source code from TechTeach.no
  - Nonlinear model
- FMU runs in parallel with nonlinear model of the air heater

Python - FMPy

- Set the FMU simulation parameters
- Connect model value references to variables
  - Enables get/set in simulation
- Import the air heater FMU object
- Set initial values for the set values
- Simulate system with doStep()
- Get and set variables for each timestep in simulation

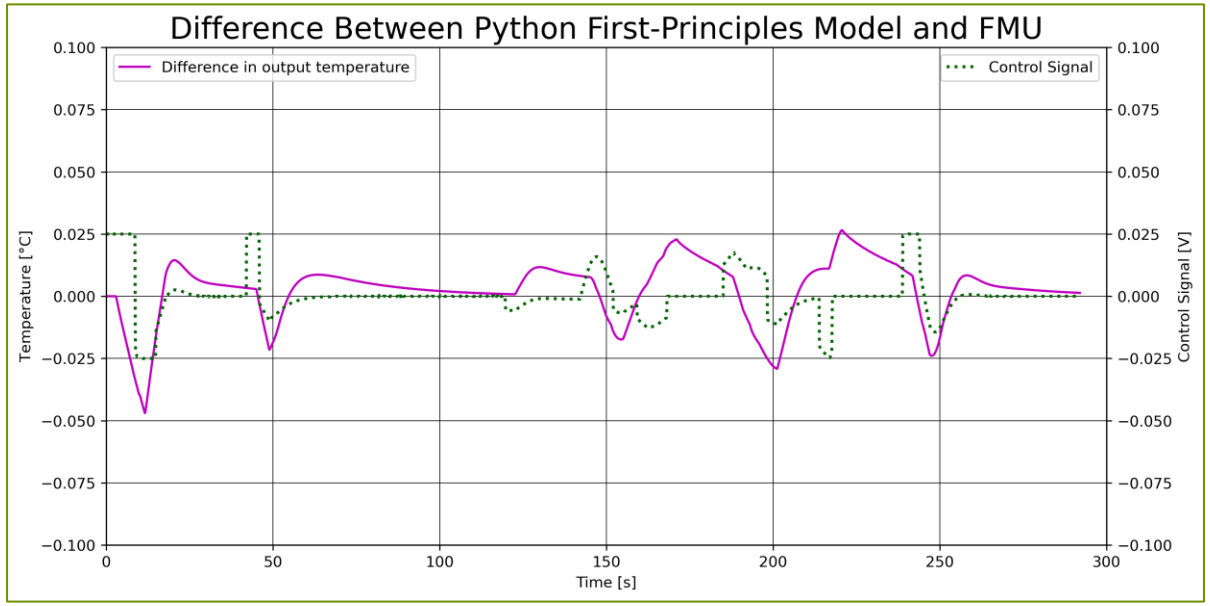
FMU of Air Heater with MPC in Python

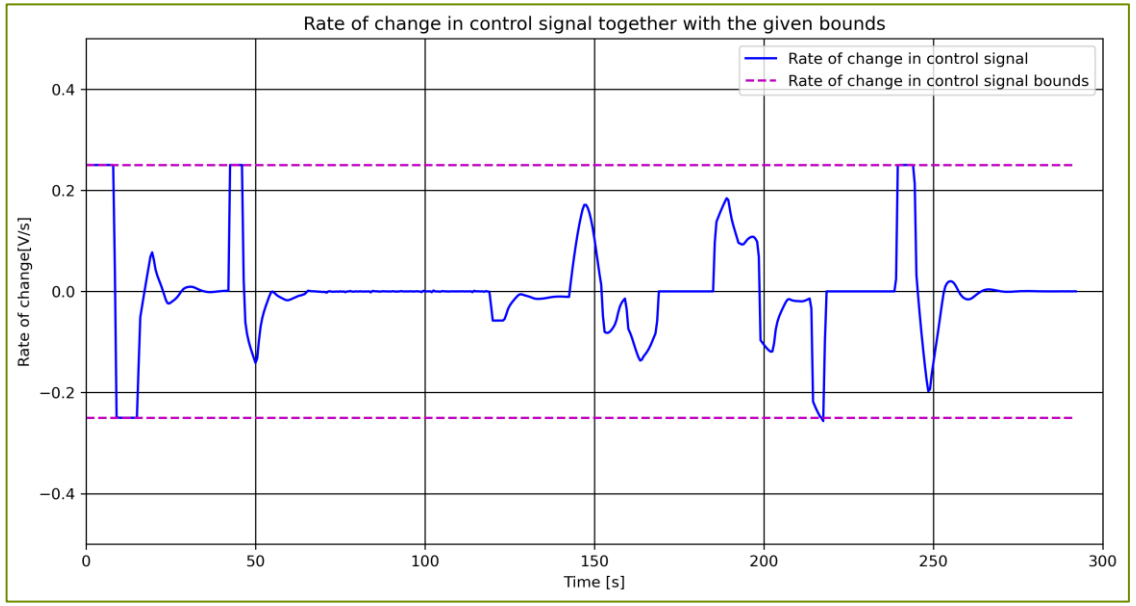
Results



FMU of Air Heater with MPC in Python

Results

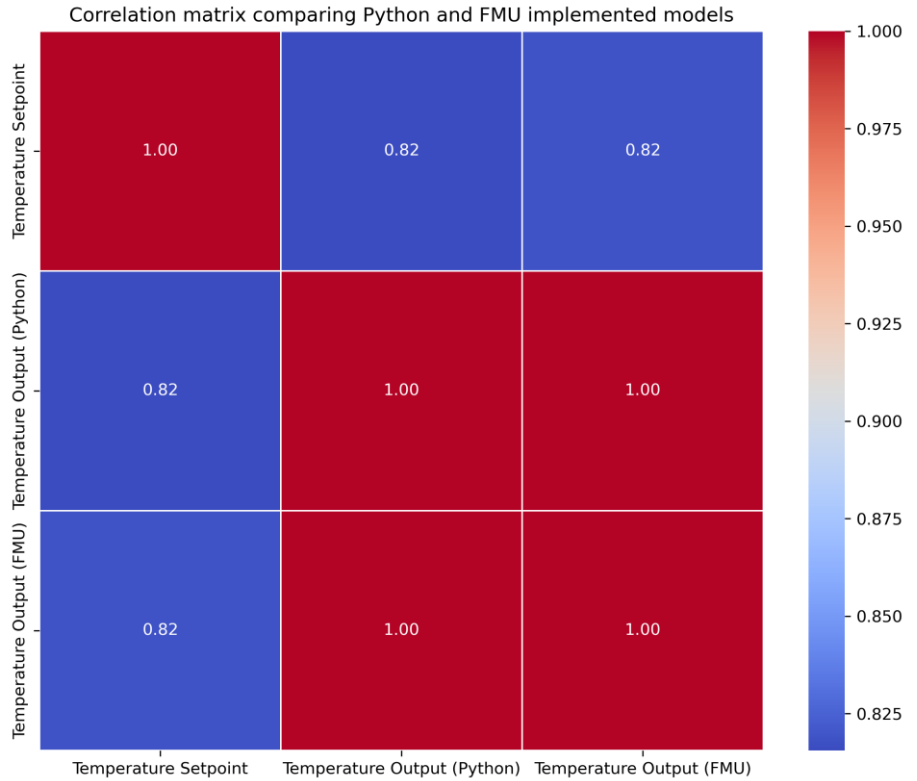




FMU of Air Heater with MPC in Python

Measures

Measures	Control Signal [V]			Output Temperature [°C]		
	Python	FMU	Diff	Python	FMU	Diff
Data points	2921	2921	0	2921	2921	0
Mean	0.696	0.696	0.000	30.259	30.256	0.003
Standard dev.	0.417	0.417	0.000	0.692	0.688	0.004
Min	0.000	0.000	0.000	28.000	28.000	0.000
Max	2.175	2.175	0.000	31.005	31.004	0.001



* Error in measures is corrected in this version

Conclusion

Both methods of implementation have their strengths and weaknesses

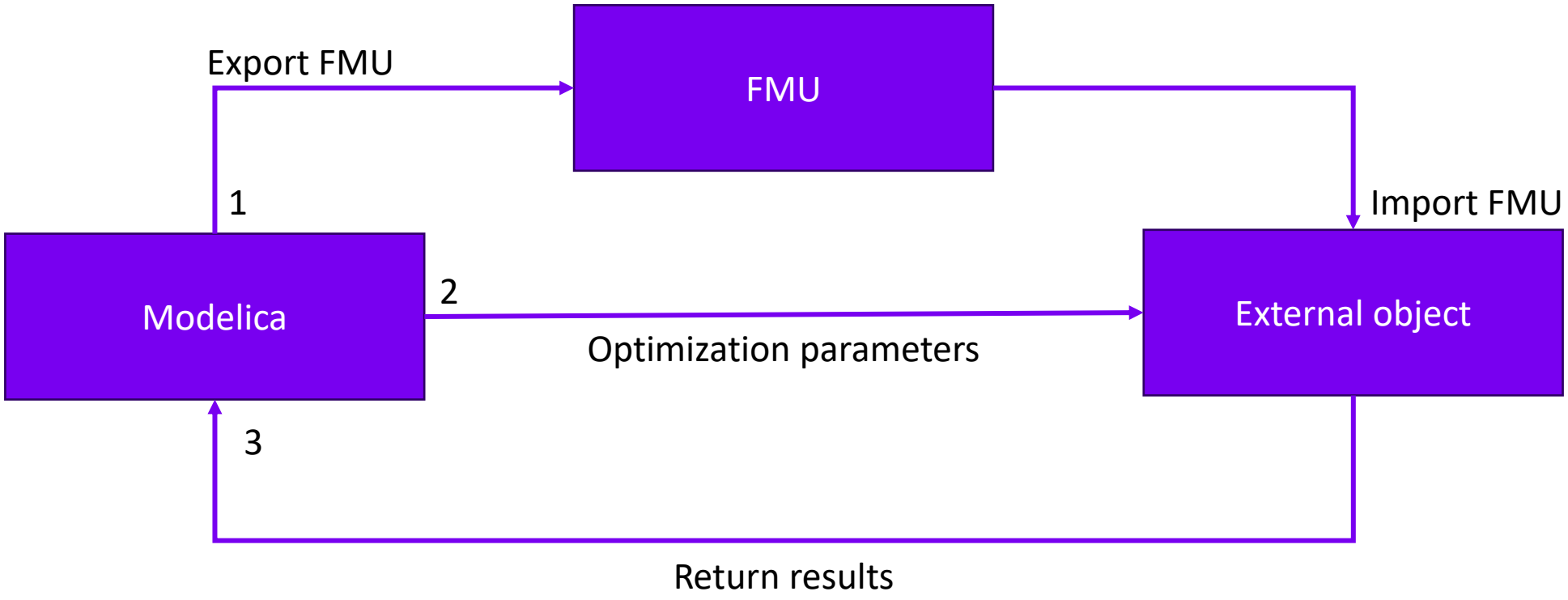
External objects in Modelica

- Relatively fast calculations
- Complex implementation

FMU in Python

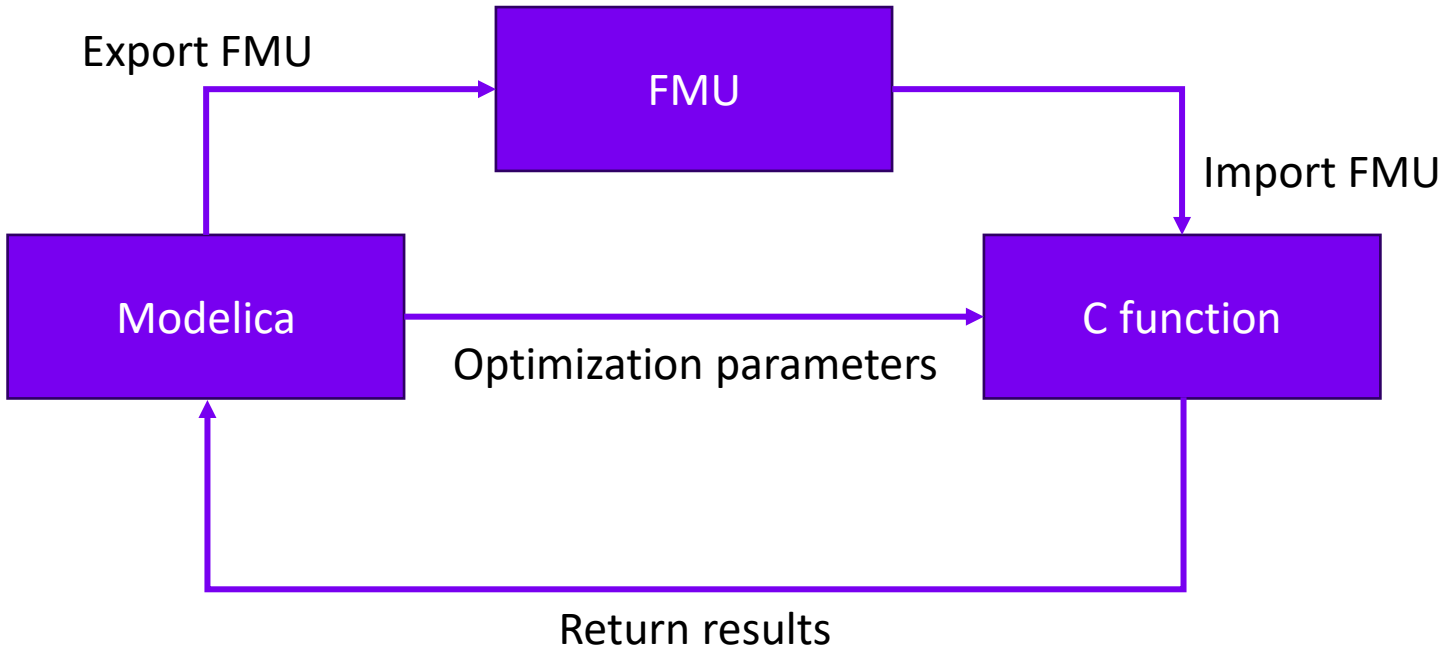
- Versatile solution
- Common and highly adopted language

Future Work


Export FMU
FMU
Import FMU
1
Modelica
2
External object
Optimization parameters
3
Return results

Thank you for the attention

University of
South-Eastern Norway


Export FMU
FMU
Import FMU
Modelica
Optimization parameters
C function
Return results