



A framework for simulation and model predictive control of hybrid district heating systems

Authors: Gerald Schweiger ^a, Per-Ola Larsson ^b, Stéphane Velut ^b, Patrick Lauenburg ^c

(a) AEE – Institut für Nachhaltige Technologien (AEE INTEC)
A-8200 Gleisdorf, Feldgasse 19
AUSTRIA



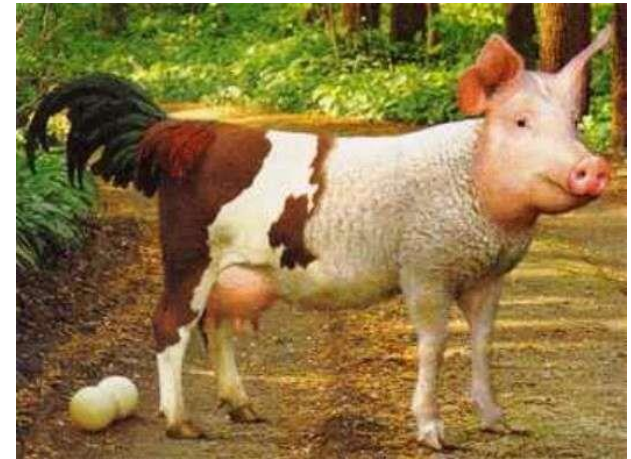
(b) Modelon AB
223 70 Lund, Ideon Science Park Beta 6 building Scheelevägen 17
SWEDEN



(c) Lund University Division of Efficient Energy
3223 63 Lund, Kårhuset, John Ericssons väg
SWEDEN

Motivation

"Another new framework for simulation and MPC of hybrid Energy system"

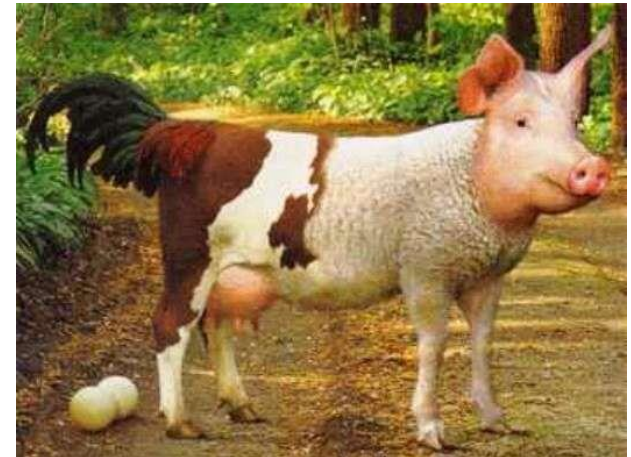


<http://blog.inkyfool.com>

Motivation

"Another new framework for simulation and MPC of hybrid Energy system"

We can't do everything; not everything is new
...but...



<http://blog.inkyfool.com>

[...] joint effort of industry, applied research and university to further develop tools and methods for **detailed** simulation and optimization of (future) city (district) energy systems.

Motivation

Simulation, Optimal control, MPC (**M**odel **P**redictive **C**ontrol)

Typical use-cases

- To investigate design of new city districts
- To find optimal control strategies for different system settings

Requirements for the simulation/optimal control method given by the real world

- **Dynamics**: The simulation and optimal control must capture all important dynamics
- **Multi-domain**: The framework must be designed to analyze multi-domain problems

Requirements

- ➔ given by mathematics and physics
 - Nonlinear models
 - Stiff systems
 - Hybrid systems: event handling & variable time steps

- ➔ given by simulation/optimization method
 - Simulation: “acceptable” ratio of eigenvalues
 - Optimal control: reduce complexity of models or network topology
 - Optimal control : C^2 (Twice continuous differentiability)
 - Optimal control: Handle MINLP [**M**ixed **I**nteger **N**onlinear **P**rograms]

- ➔ given by the “user”
 - Robust, flexible, adoptable, extensible
 - “Master-algorithm”

Main contributions

First integrated framework for simulation and model predictive control of hybrid district heating system

- Uses a non-causal, equation based modelling language (Modelica) and a high-level, large-scale dynamic optimization method
- Possibility to impose constraints on physically relevant variables
- Multiphysics, multi domain
- Suitable for stiff systems
- Flexibility, modular expandability, reusability of models
- co-simulation possible

The framework

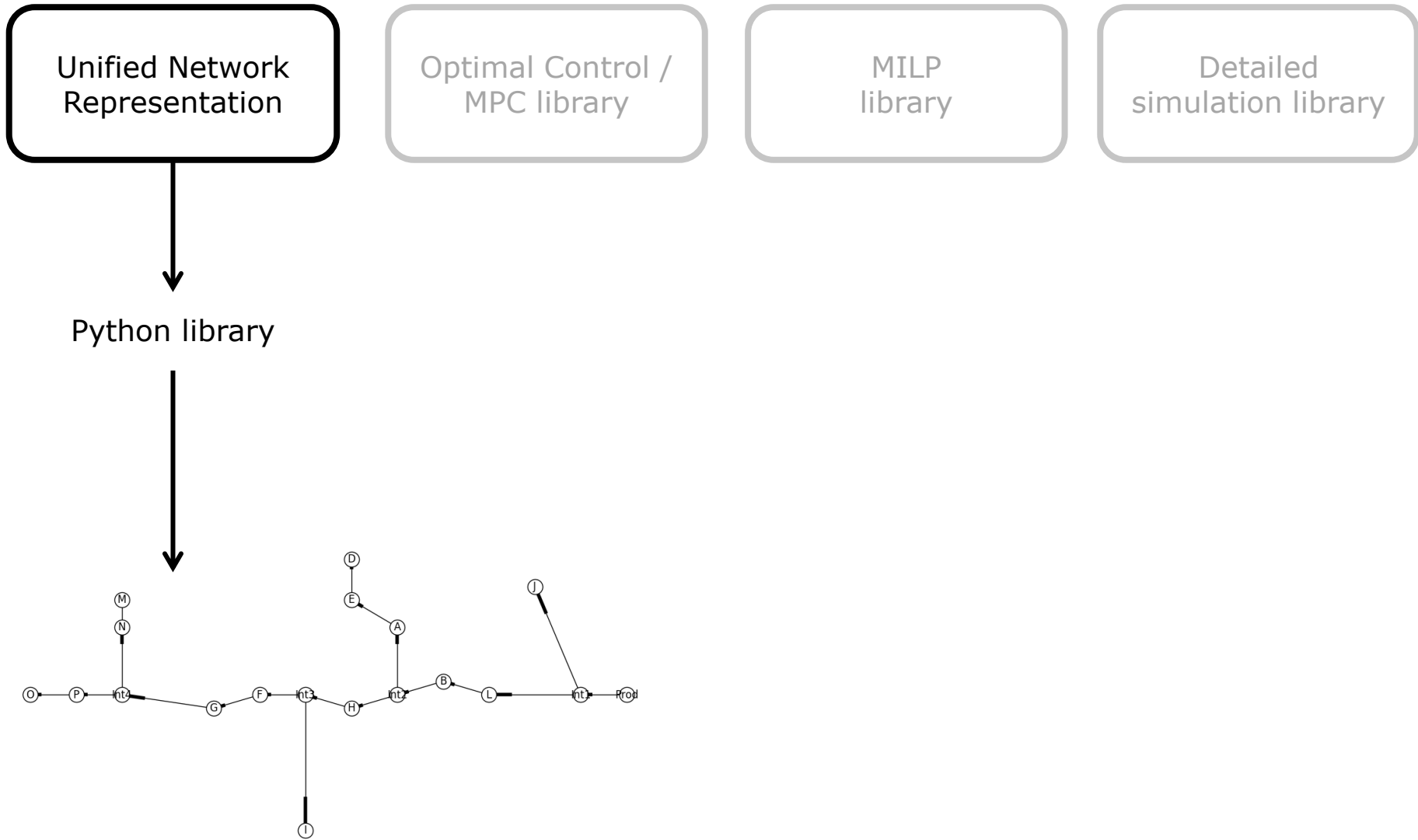
Unified Network
Representation

Optimal Control /
MPC library

MILP
library

Detailed
simulation library

The framework



The framework

Unified Network Representation

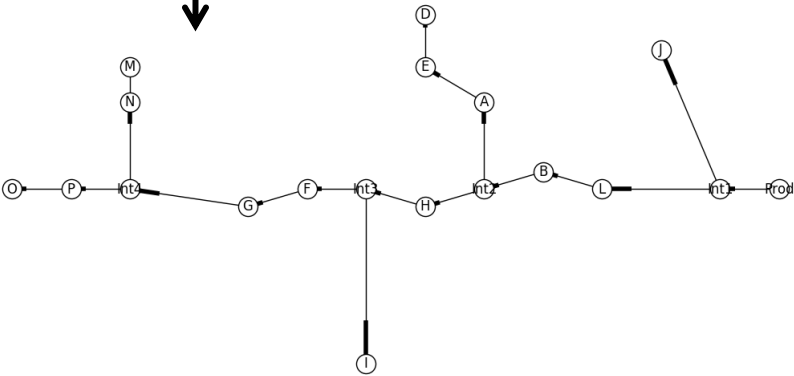
Optimal Control / MPC library

MILP library

Detailed simulation library



Python / NetworkX

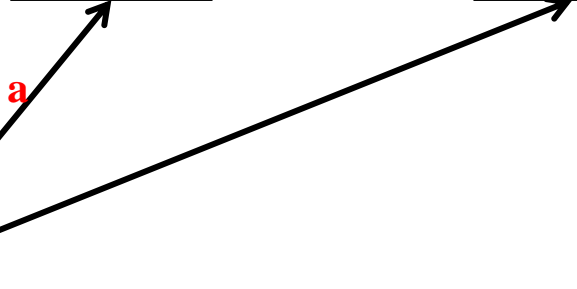


```
model PythonNetworkSim
replaceable model inputs = Modelica.Blocks.Interfaces.RealInput;
replaceable package Medium =
  OMNLibrary.Media.ConsumerPropertyLiquidWater;
replaceable model Producer =
  OMNLibrary.DistrictHeatingOptimization.HeatConnector.Tests.HeatProducer_gz;
replaceable model LoadPredictor =
  OMNLibrary.DistrictHeatingOptimization.LoadPredictor_greatInit;
replaceable model Pipe =
  OMNLibrary.DistrictHeatingOptimization.HeatConnector.Tests.HeatPipe_gz_1mMod;
replaceable model Consumer =
  OMNLibrary.DistrictHeatingOptimization.HeatConnector.Consumer.Customer_fixedQ_minCalcLive_gz;
Heater1:TepplyOev = Prod.TepplyOev;
Heater1:delayedSupplyO = dualPipe1_I_0.supplyI.delayedInput;
Heater1:supplySupplyO = dualPipe1_I_0.supplyI.inputToBeDelayed;
Heater1:delayedReturnO = dualPipe1_I_0.returnI.delayedInput;
Heater1:supplyReturnO = dualPipe1_I_0.returnI.inputToBeDelayed;
Heater2:delayedSupplyI1 = dualPipeProd_I1.supplyI1.delayedInput;
Heater2:supplySupplyI1 = dualPipeProd_I1.supplyI1.inputToBeDelayed;
Heater2:delayedReturnI1 = dualPipeProd_I1.returnI1.delayedInput;
Heater2:supplyReturnI1 = dualPipeProd_I1.returnI1.inputToBeDelayed;
equation
  Prod.TepplyOev = if time<6400 then 279.15+68 else qgas_load.y[2];
end
```

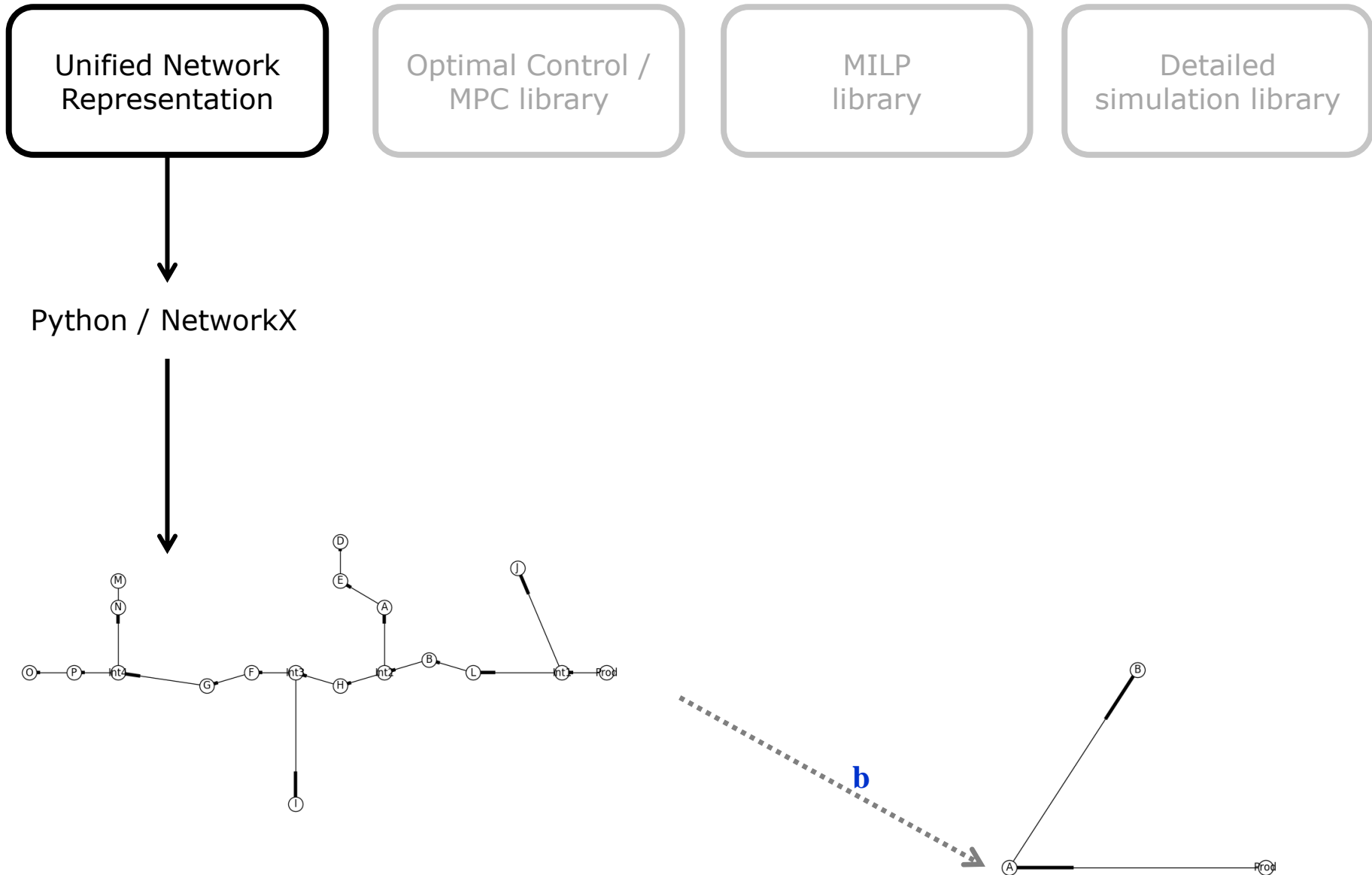
Library X

```
model PythonNetworkSim
replaceable model inputs = Modelica.Blocks.Interfaces.RealInput;
replaceable package Medium =
  OMNLibrary.Media.ConsumerPropertyLiquidWater;
replaceable model Producer =
  OMNLibrary.DistrictHeatingOptimization.HeatConnector.Tests.HeatProducer_gz;
replaceable model LoadPredictor =
  OMNLibrary.DistrictHeatingOptimization.LoadPredictor_greatInit;
replaceable model Pipe =
  OMNLibrary.DistrictHeatingOptimization.HeatConnector.Tests.HeatPipe_gz_1mMod;
replaceable model Consumer =
  OMNLibrary.DistrictHeatingOptimization.HeatConnector.Consumer.Customer_fixedQ_minCalcLive_gz;
Heater1:TepplyOev = Prod.TepplyOev;
Heater1:delayedSupplyO = dualPipe1_I_0.supplyI.delayedInput;
Heater1:supplySupplyO = dualPipe1_I_0.supplyI.inputToBeDelayed;
Heater1:delayedReturnO = dualPipe1_I_0.returnI.delayedInput;
Heater1:supplyReturnO = dualPipe1_I_0.returnI.inputToBeDelayed;
Heater2:delayedSupplyI1 = dualPipeProd_I1.supplyI1.delayedInput;
Heater2:supplySupplyI1 = dualPipeProd_I1.supplyI1.inputToBeDelayed;
Heater2:delayedReturnI1 = dualPipeProd_I1.returnI1.delayedInput;
Heater2:supplyReturnI1 = dualPipeProd_I1.returnI1.inputToBeDelayed;
equation
  Prod.TepplyOev = if time<6400 then 279.15+68 else qgas_load.y[2];
end
```

Library Y



The framework



The framework

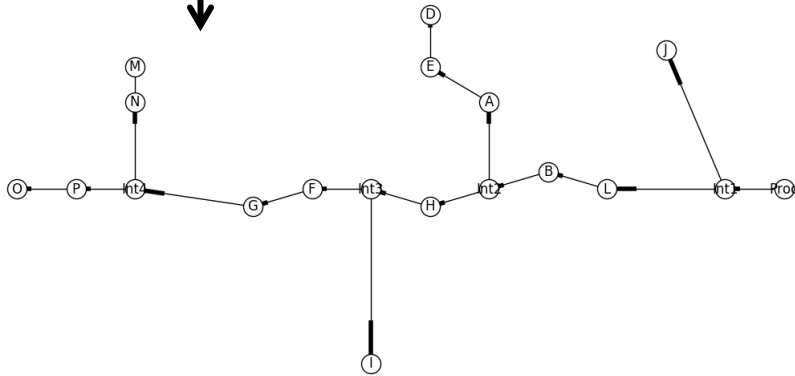
Unified Network Representation

Optimal Control / MPC library

MILP library

Detailed simulation library

Python / NetworkX



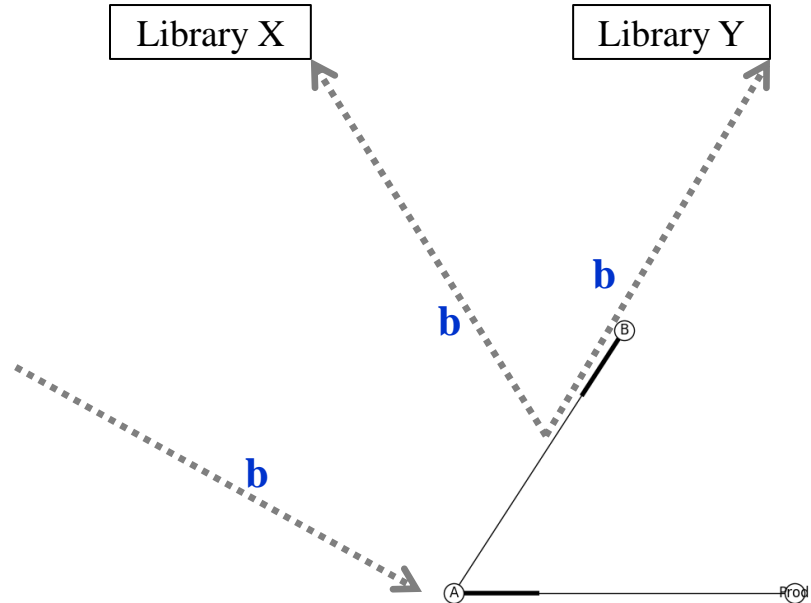
```
model PythonNetworkSim
replaceable model inputs = Modelica.Blocks.Interfaces.RealInput;
replaceable package Medium =
  OMNLibrary.Media.ConstantsPropertyLiquidWater;
replaceable model Producer =
  OMNLibrary.DistrictHeatingOptimization.NewConnector.Tests.SimProducer_gv;
replaceable model LoadPredictor =
  OMNLibrary.DistrictHeatingOptimization.LoadPredictor_greatIn;
replaceable model Pipe =
  OMNLibrary.DistrictHeatingOptimization.NewConnector.Tests.SimPipe_gv_1ambda;
replaceable model Consumer =
  OMNLibrary.DistrictHeatingOptimization.NewConnector.Consumer.Customer_fixedQT_refinCalcInLive_gv;
end

Pipe TeppalyOev = Prod.TeppalyOev;
Pipe delayedSupplyO = dualPipe1_0.supp1.y1.delayedInput;
Pipe toDelayReturnO = dualPipe1_0.return1.inputToBeDelayed;
Pipe delayedSupplyI = dualPipeProd_11.supp1.y1.delayedInput;
Pipe toDelayReturnI = dualPipeProd_11.return1.inputToBeDelayed;
Pipe toDelayReturnII = dualPipeProd_11.return2.delayedInput;
Pipe toDelayReturnIII = dualPipeProd_11.return1.inputToBeDelayed;

equation
  Prod.TeppalyOev = if time<6400 then 279.15+68 else qgas_load.y[2];
end
```

Library X

Library Y



The framework

Unified Network
Representation

Optimal Control /
MPC library

MILP
library

Detailed
simulation library

JModelica &
OPTIMICA
Compiler Toolkit

```
graph TD; A[Unified Network Representation] --- B[Optimal Control / MPC library]; B --- C[MILP library]; B --- D[Detailed simulation library]; B --> E[JModelica & OPTIMICA Compiler Toolkit];
```

The diagram illustrates the framework's architecture. It consists of five main components arranged in a horizontal line: 'Unified Network Representation', 'Optimal Control / MPC library', 'MILP library', and 'Detailed simulation library'. The 'Optimal Control / MPC library' component is highlighted with a thick black border, while the others have a thin grey border. An arrow points from the 'Optimal Control / MPC library' component down to a separate box labeled 'JModelica & OPTIMICA Compiler Toolkit', which also has a thick black border. This indicates that the compiler toolkit is a foundational or supporting component for the optimal control and MPC library.

The framework

Unified Network Representation

Optimal Control / MPC library

MILP library

Detailed simulation library

JModelica &
OPTIMICA
Compiler Toolkit

$$\underset{u(t)}{\text{minimize}} \quad \phi(t_f, x(t_f)) + \int_{t_0}^{t_f} L(t, x(t), u(t)) dt$$

$$\begin{aligned} \text{s.t. } & F(t, \dot{x}(t), x(t), u(t)) = 0 \\ & x(0) = x_0 \end{aligned}$$

$$\text{Path constraints: } g_i(x(t), u(t)) \leq 0$$

$$\text{Point constraints: } g_e(x(t), u(t)) = 0$$

Collocation
method
Convert infinite to a
finite dimensional
optimization problem

Solve NLP
[Nonlinear programm]

The framework

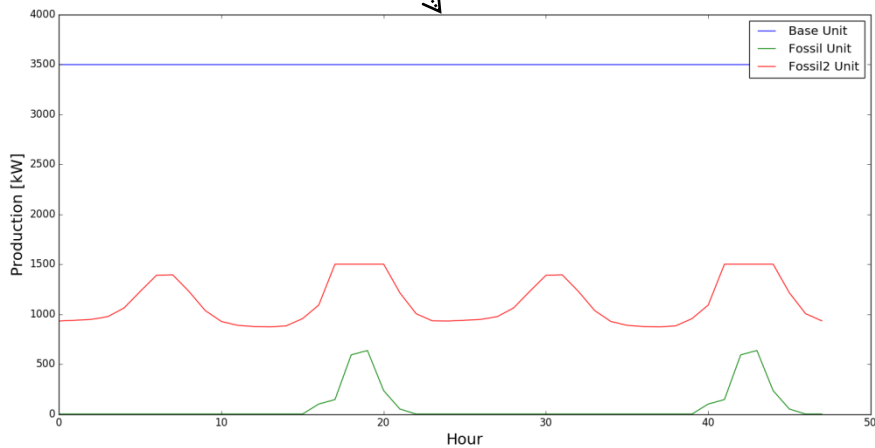
Unified Network Representation

Optimal Control / MPC library

MILP library

Detailed simulation library

Solve MILP



The framework

Unified Network Representation

Optimal Control / MPC library

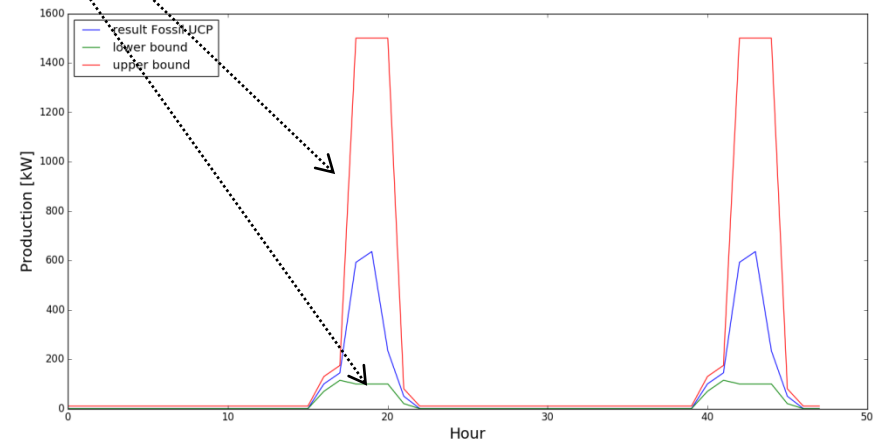
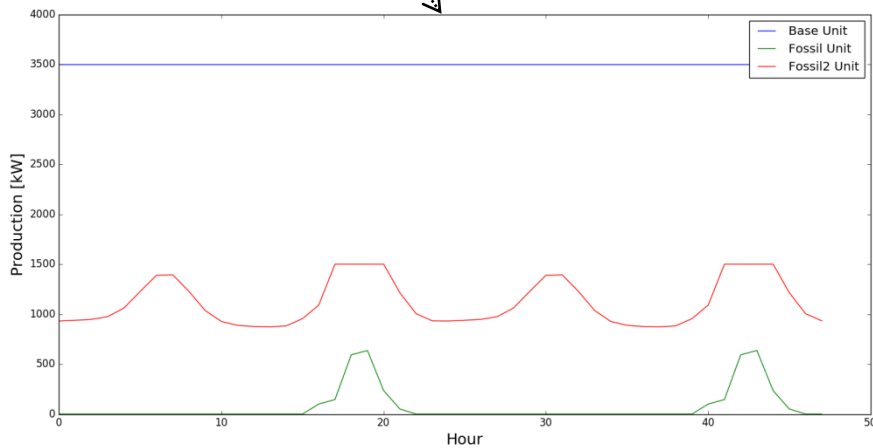
MILP library

Detailed simulation library

Solve MILP

Calculate upper /lower constraints

Translate into optimal control/MPC constraints



The framework

Unified Network
Representation

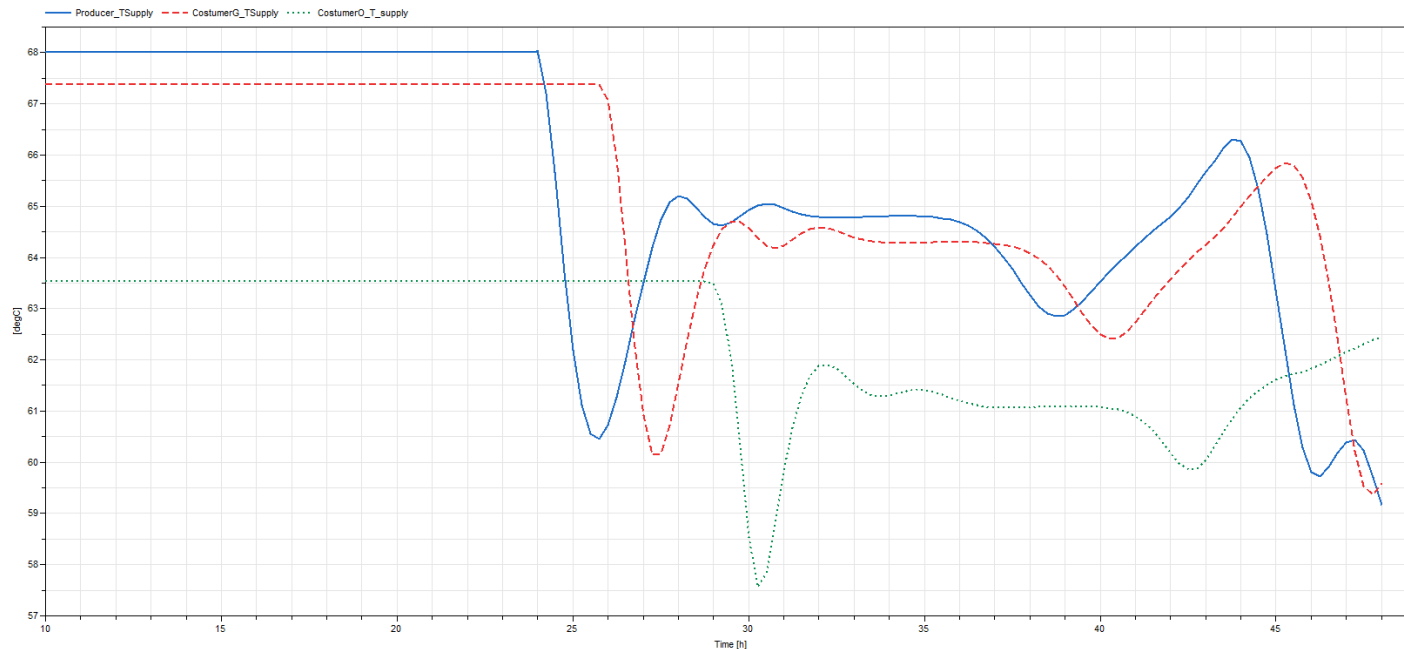
Optimal Control /
MPC library

MILP
library

Detailed
simulation library

Development Highlight

- Precise, numerically robust district heating pipe model (plug-flow approach based on Modelica spatialDistribution operator)



Use case „ScaleTest“

System details

- Consumer: 103
- Total length: 14.000m
- Loop

Model details

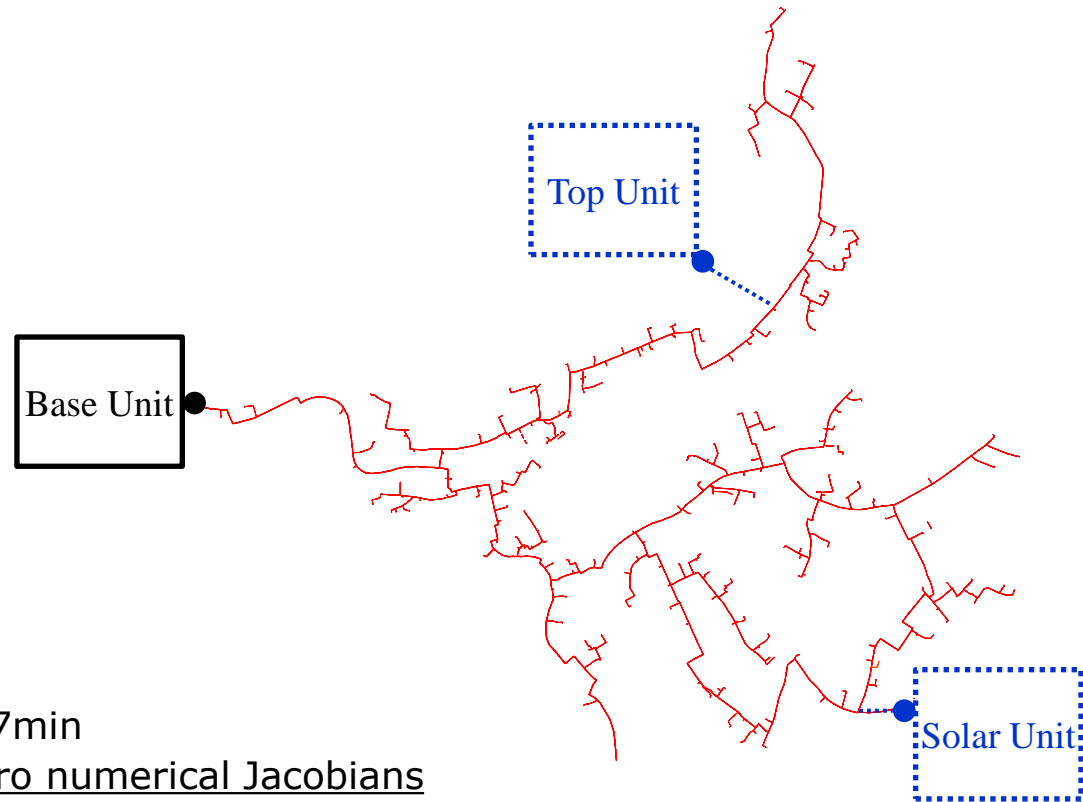
- Components: ~ 2.800
- Variables: ~ 50.000
- Equations: ~22.000

Case „base“

- One „Base Unit“ 4MW
- Computation time 1day → 7min
- Zero nonlinear systems, zero numerical Jacobians

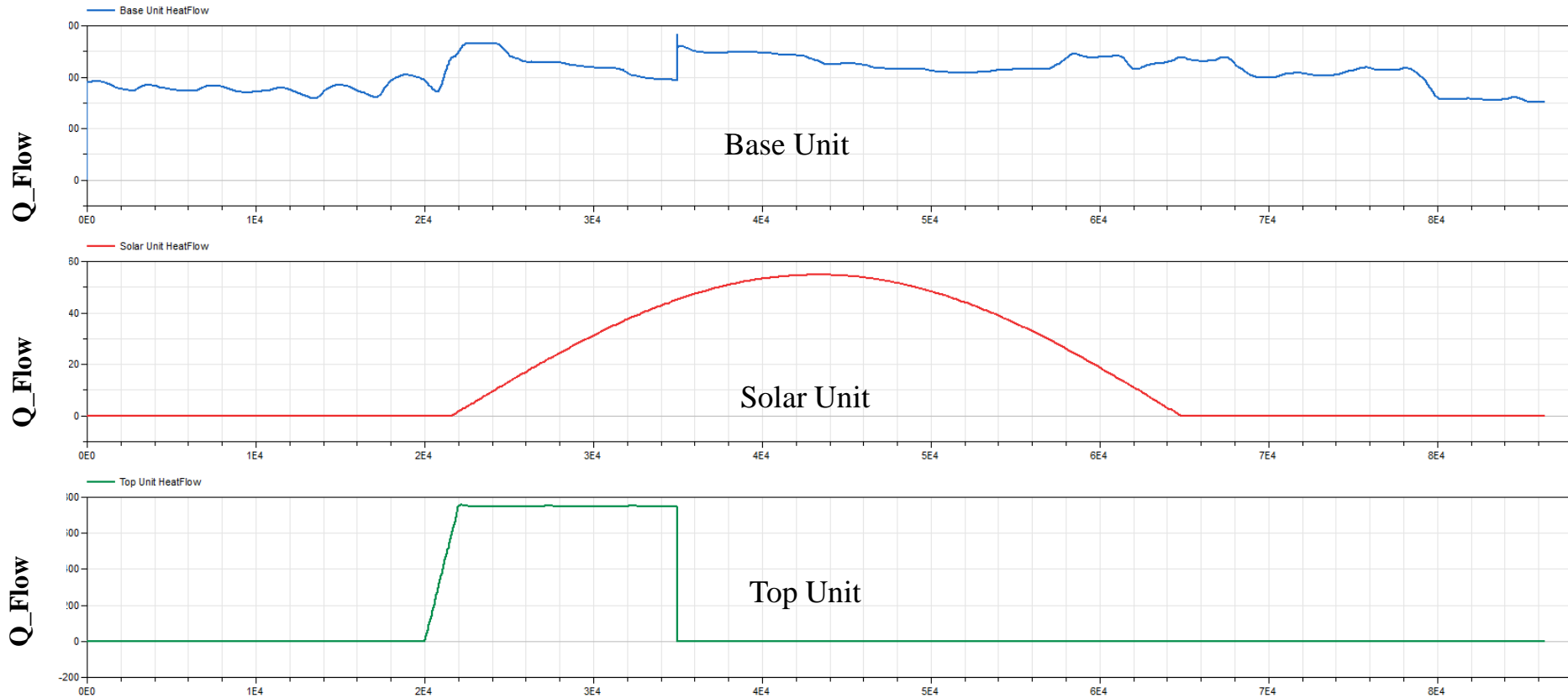
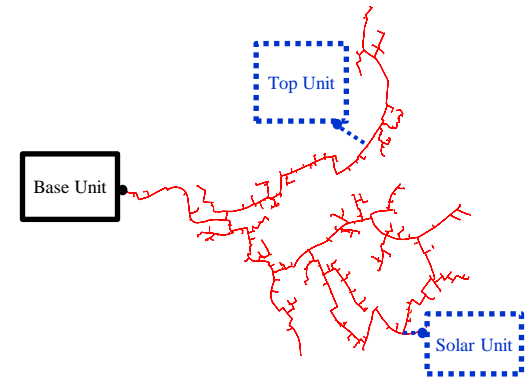
Case „modified“

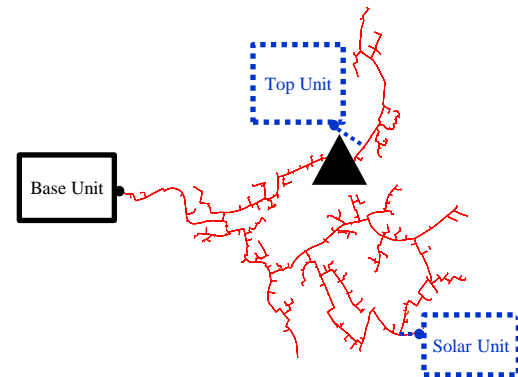
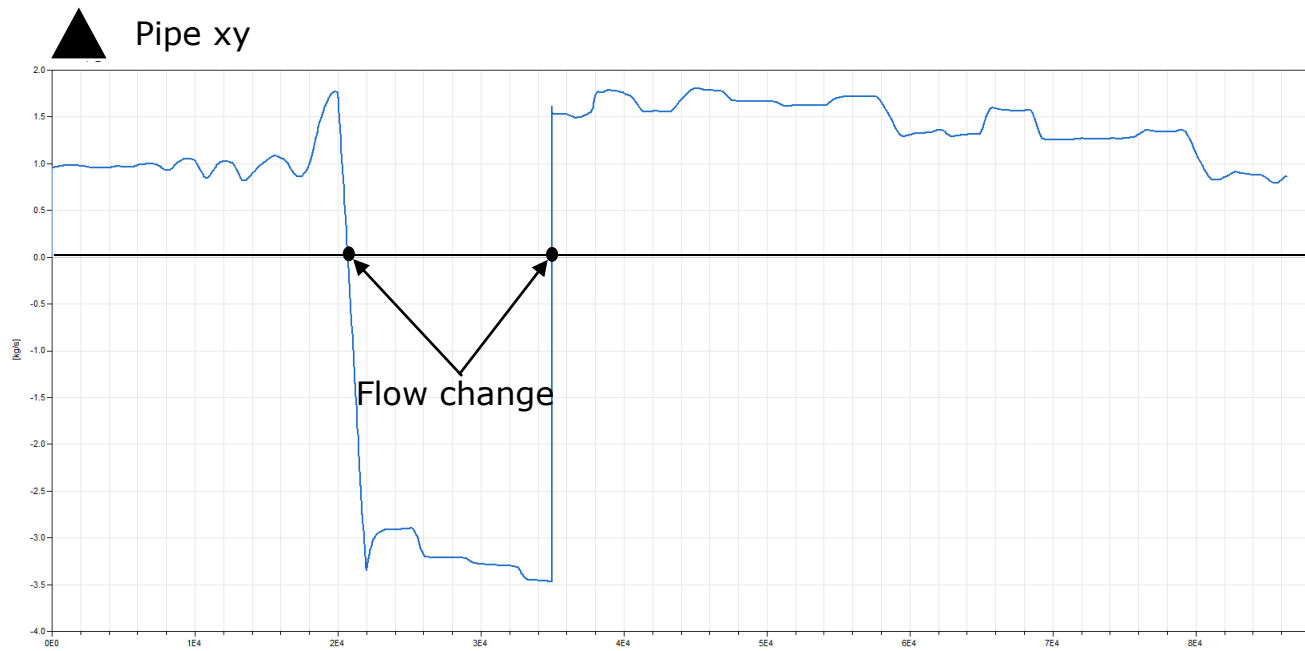
- „Base Unit“ 4MW, „Top Unit“ 750kW, „Solar Unit“ 55kWp
- Computation time 1day → 8min
- Zero nonlinear systems, zero numerical Jacobians

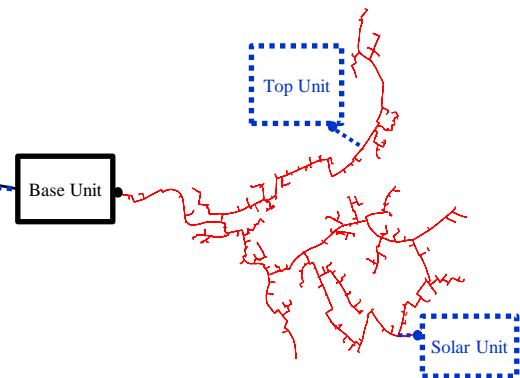
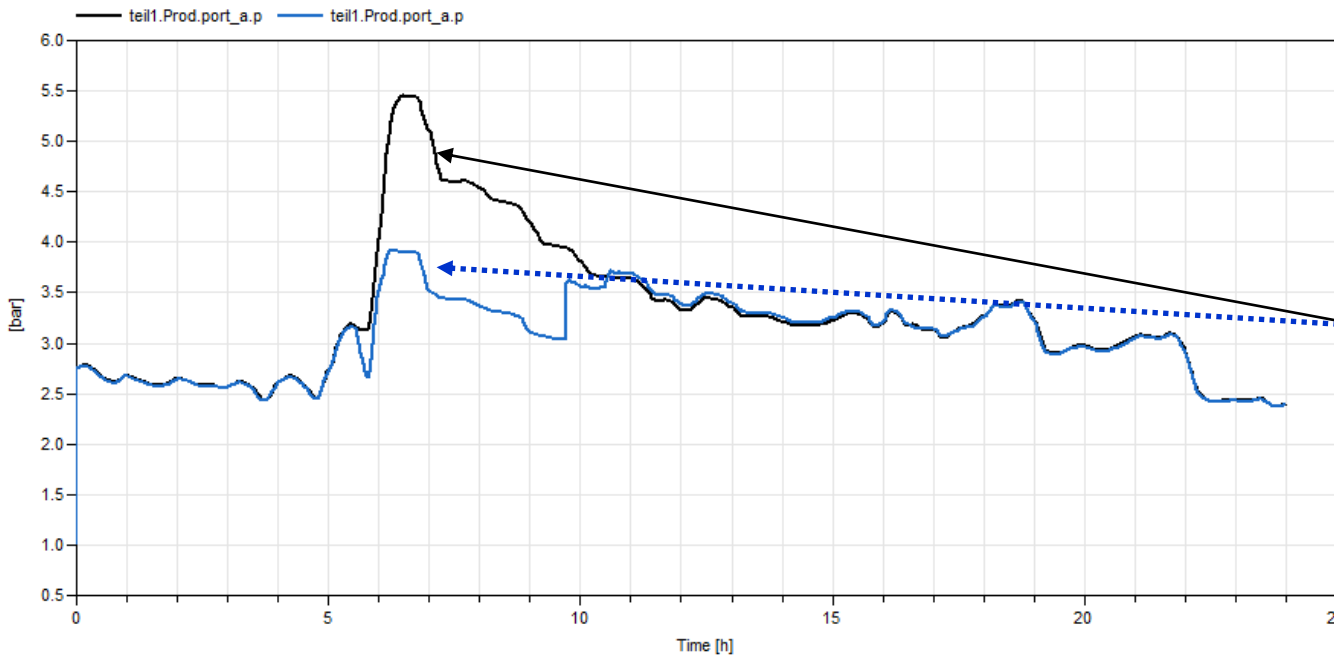
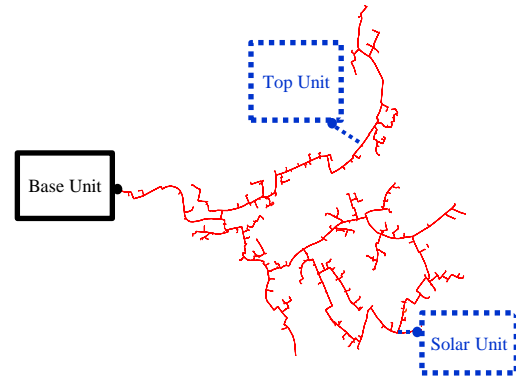
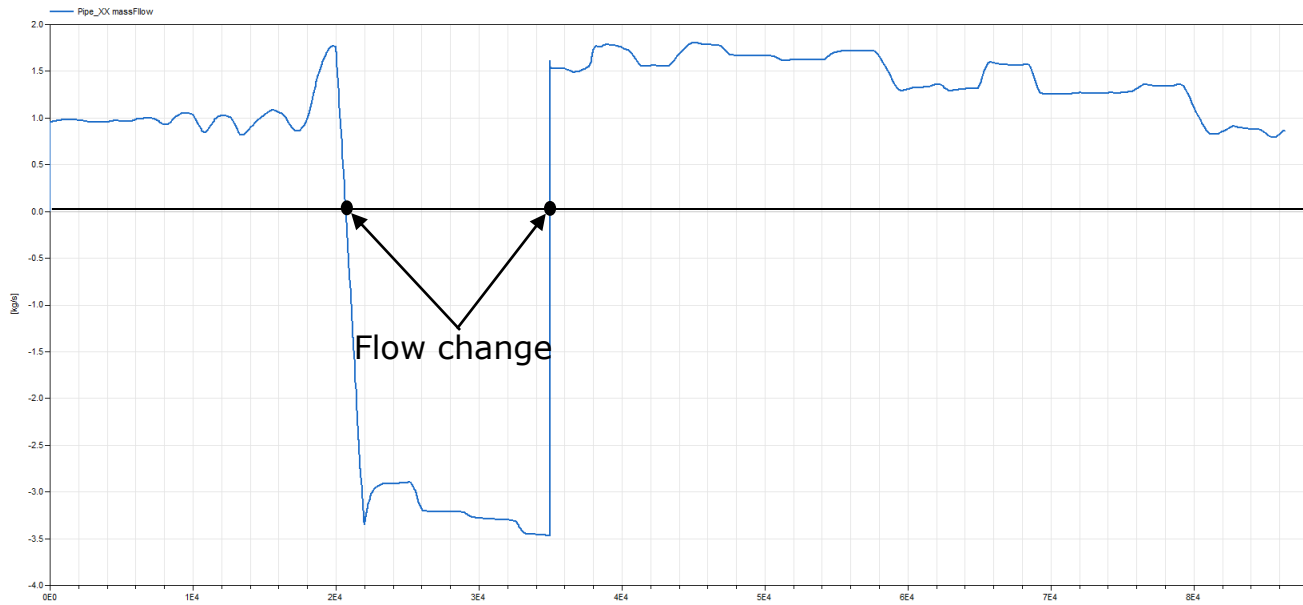


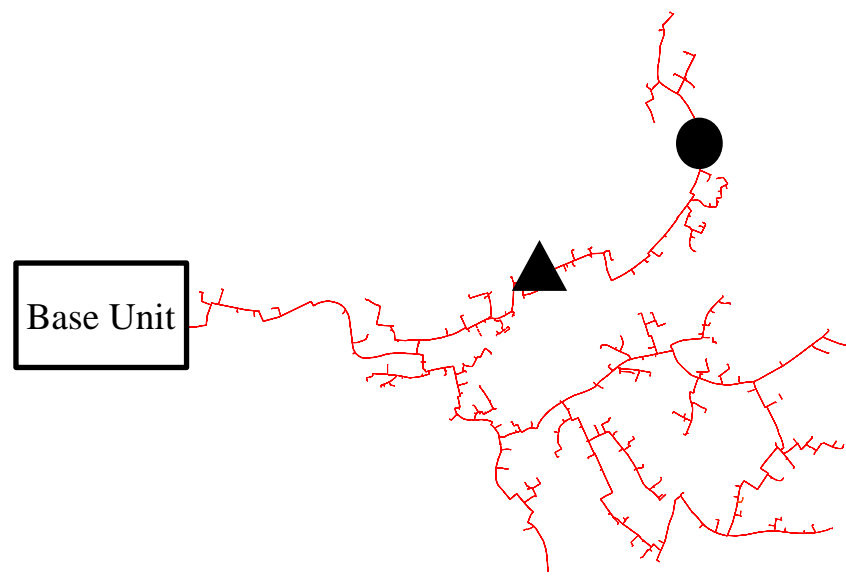
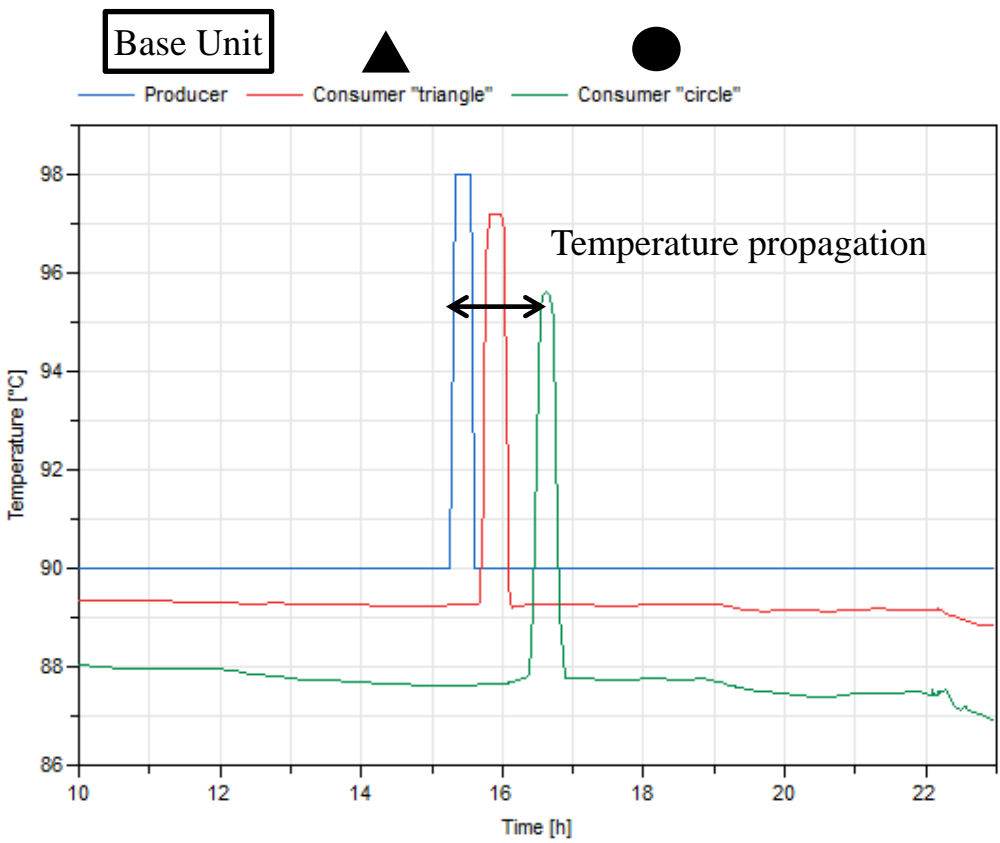
Numerical challenges

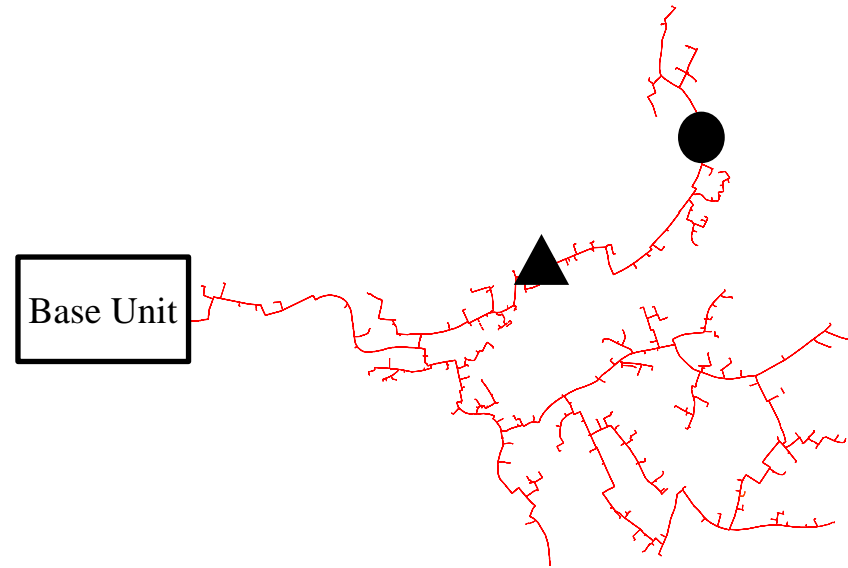
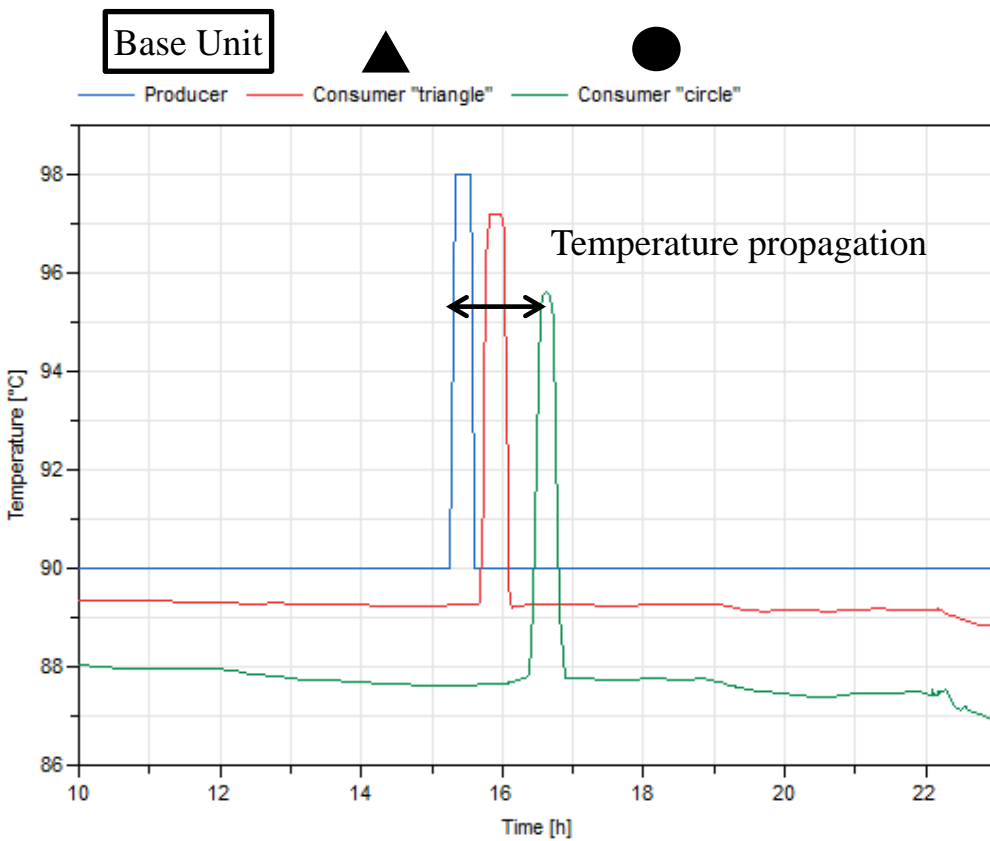
- Fast transients
- Reverse flow & zero flow states
- Temperature-wave propagation











Insights

- Robust library
- Focus on numerical improvements!!! → Address large scale problems
- Accurate delay propagation
- Accurate heat-loss model
- Flexible hydraulic models

Use case „Optimal-control / MPC“

System details

- Consumer: 16
- Total length: 4194m

Model details

- Variables: ~ 10.100
- Equations: ~ 4.500
- Computation time (**Sim**) 1day \rightarrow 26sec

Goal:

- Challenge robustness
- Aggregation: 16 \rightarrow 2 consumer
- Validate temperature propagation
 - mass flow dependent

Assumptions & constraints

- $\max(\text{Prod_mflow}) = 50 \text{ kg/s}$
- Objective: $\min T_{\text{supply}}$



Numerical challenges

- Mass flow dependent transport delay for optimization
- Twice continuously differentiable approximations of several functions

Validation

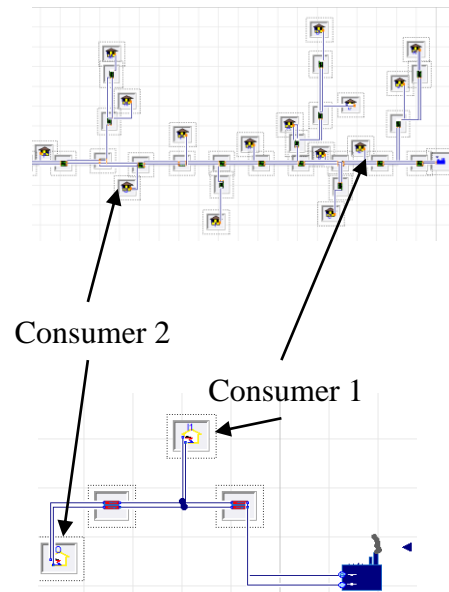
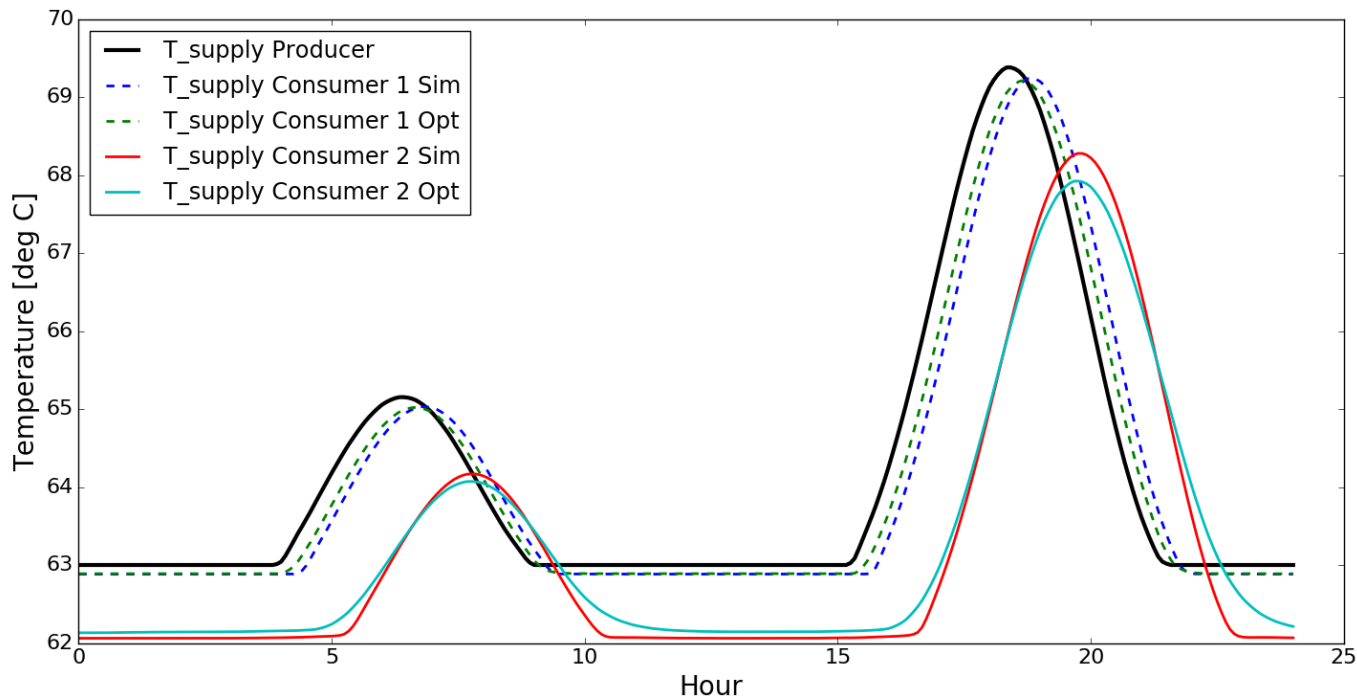
- Result optimal control vs. apply optimal trajectories on complex models & network topology
- Validate mass flow dependent delay
- Validate heat-loss models

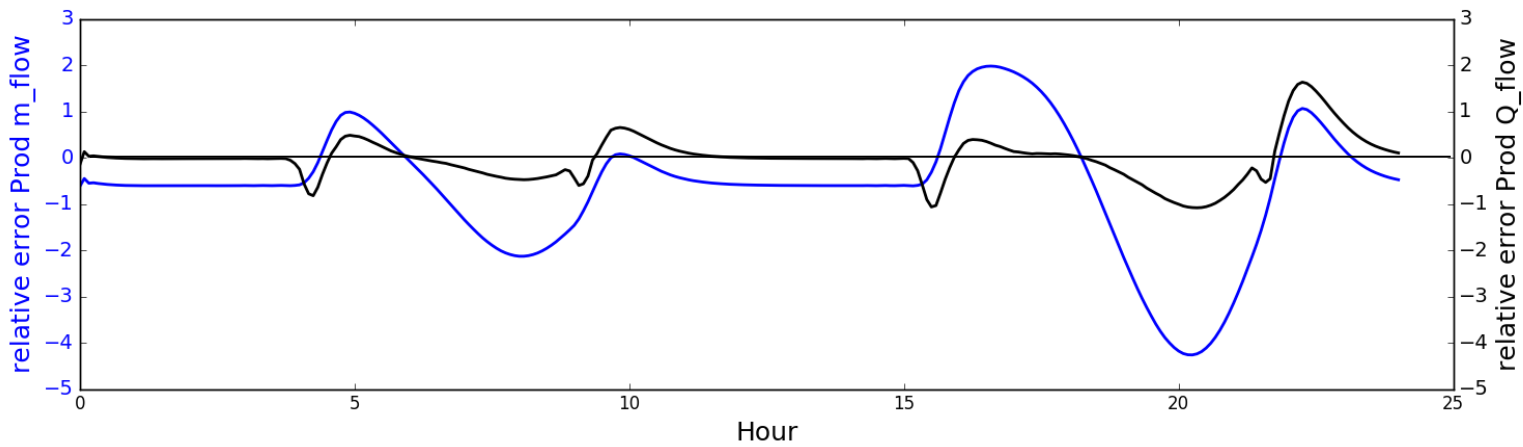
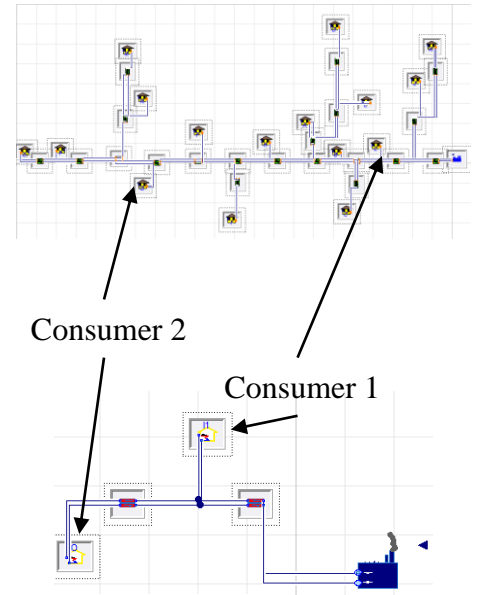
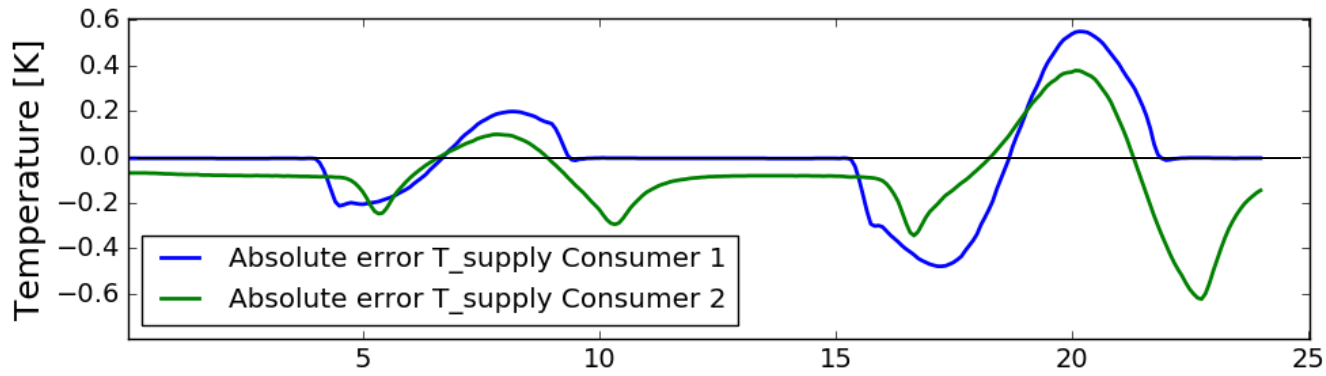
Numerical challenges

- Mass flow dependent transport delay for optimization
- Twice continuously differentiable approximations of several functions

Validation

- Result optimal control vs. apply optimal trajectories on complex models & network topology
- Validate mass flow dependent delay
- Validate heat-loss models





Scale of optimal control / MPC problem

- 41.000-90.000 variables [2 vs. 5 consumer left]
- 40.000 -89.000 constraints [2 vs. 5 consumer left]
- Medium sized problem for NLP Solver
- Computation time for the whole chain 1day: 4-8min

Validation

- Low max/min errors → precise mass flow dependent delay within optimal control/MPC
- Mean errors over a optimization horizon → „neglectable“
- Accurate aggregation method

Outlook

- MILP → units at different geographical positions
- Optimal control → handle reverse flow
- Integrate detailed models of power system into the optimal control library
- MPC Big-Scale-Validation based on measured data