Tutorial

District heating network simulations using the TransiEnt Library

If you have any questions regarding this tutorial please feel free to contact us at transientlibrary@tuhh.de

Introduction:

The TransiEnt Library offers a variety of models for the simulation of integrated energy systems. One important use case for the TransiEnt models is the simulation of district heating networks. Several investigations can be done using these models, for example:

- the investigation of heat losses,
- the design of control strategies for the supply temperature,
- the design of control strategies for pump stations,
- the integration of renewable heat sources like solar thermal heat,
- the investigation of efficiencies of the heat producers for different operational conditions,
- the optimization of parameters like the capacity of storages or the heating power of heat producers.

This tutorial shall help the user to create district heating network models and show how investigations using the TransiEnt can be made. Two detailed examples of how the TransiEnt models can be used are described in [1] and [2].

For the simulation of a district heating network (DHN), at least five components are needed. A heat producer, pump stations, pipes, junctions and consumers. However, for the simulation to run, some auxiliary components are necessary. In the case of this tutorial, the following TransiEnt models are used:

1. The predefined heating grid typology **`TransiEnt.Grid.Heat.HeatGridTopology.TopologyA_Ports`** built of nine energy consumers that are interconnected by pipes and junctions
2. The heat pump model **`TransiEnt.Producer.Heat.Power2Heat.Heatpump.Heatpump_DHN`**
3. The boiler model **`TransiEnt.Producer.Heat.Gas2Heat.SimpleGasBoiler.SimpleBoiler`**
4. The fluid sink model **`TransiEnt.Components.Boundaries.FluidFlow.FluidSink`**
5. Two pump models: **`TransiEnt.Components.Heat.SimplePump`** (providing a predefined mass flow) and **`TransiEnt.Components.Heat.Pump_Dp`** (providing a predefined pressure difference)
6. **`TransiEnt.Basics.Adapters.FluidPortAdapter`** as an adapter between the two different fluid port types currently used in the TransiEnt library
7. As a heat storage, the model **`TransiEnt.Storage.Heat.HotWaterStorage_constProp_L4.HotWaterStorage_constProp_L4`** is used
8. Junctions in the grid are modelled with the two models **`TransiEnt.Components.Heat.VolumesValvesFittings.Fittings.Split`** and **`TransiEnt.Components.Heat.VolumesValvesFittings.Fittings.Join`**

Tutorial model step 1:

A good starting point for a DHN simulation is the model TopologyA_Ports, which includes a set of consumers that are connected via pipes and junctions. This topology model (TopologyA_Ports), the

model of the heat pump (Heatpump_DHN), four real expressions and a model of a sink need to be instantiated in the graphical editor of a model. The model of the sink is used to dictate the pressure in the grid because every other component only defines a pressure difference between the in- and outlet. The pressure input of the sink has to be connected with a real expression. The value of the real expression has to be set to a realistic grid pressure, like 3e5 Pa. The inputs of the heat pump have to be connected to real inputs as well. For the input of the supply temperature (T_set_variable), a temperature of 60 ° C can be set. The supply temperature input is only used to calculate the COP of the heat pump. The supply temperature flowing into the grid is set via the Q_flow input at a given mass flow rate and a given return temperature. In general, the consumer models calculate the mass flow rate inside the grid and the return temperature. The supply temperature is set by the producer and affects the mass flow rate. For example, if the supply temperature is increased, the consumers need less mass flow rate to transfer the same amount of heat to the building.

A sensible pressure difference has to be chosen for the dp-Input. 1.8e5 Pa is recommended for this simulation. For the heat flow rate input (Q_flow), an equation can be used and written into the real expression:

$$Q\_flow = inlet.m\_flow(cf * T_{supply} - inStream(inlet.h_{outflow}))$$

with cf as the specific heat capacity of the fluid. The operator instream() is used for stream variables like the specific enthalpy that is connected to a flow variable like the mass flow. It is used to determine the specific enthalpy flowing into a component. The variable inlet.h_outflow is defined by the developer of the component model and is only valid if the mass flow is flowing out of that fluid port. At last, the fluid ports have to be connected. The inlet of the heat pump should be connected to the outlet of the TopologyA, and the outlet of the heat pump with the inlet of the TopologyA. The inlet of the pressure sink should be connected to the outlet of the heat pump.

The finished model can be seen in Figure 1 and can be found in the TransiEnt Library (`TransiEnt.Examples.Heat.DHN_Tutorial_Step1`).

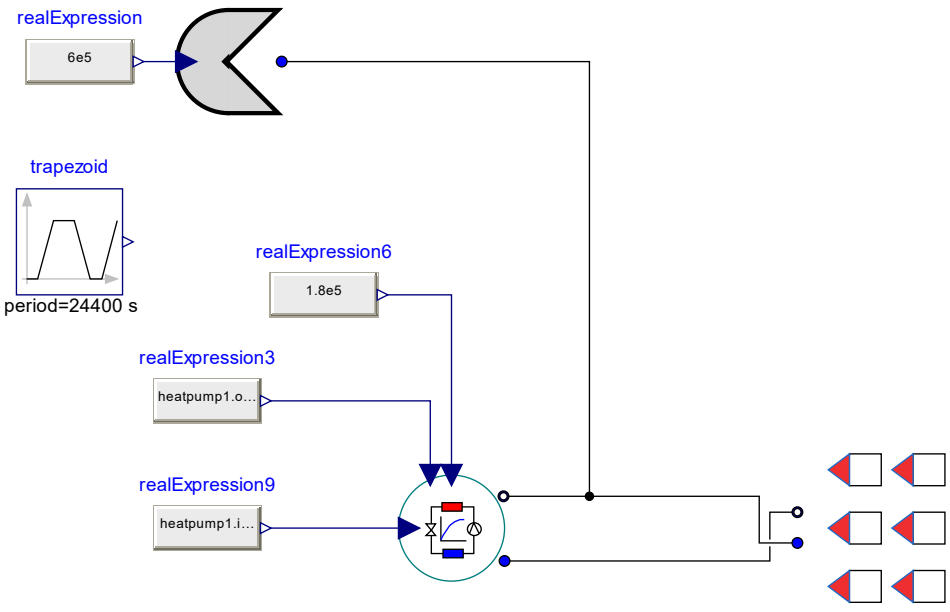

Figure 1: Tutorial model step 1 (TransiEnt.Examples.Heat.DHN_Tutorial_Step1)

The results of the simulation can be seen in Figure 2. There are two main inputs for the simulation of a DHN. The first one is the supply temperature, which is normally set by the DHN operator. The second one is the heat demanded by the consumers. In this case, the supply temperature was set in the input for Q_flow at the producer. The demanded heat flow rate can be set either in the topology model by activating an input in each of the consumer model instances or directly in the class of the consumer models. In this case, the heat flow rate is set inside the consumers by a trapezoid signal that defines the ambient temperature and leads to an increased heat demand. Accordingly, the heat flow rate produced by the heat pump increases after 86400 seconds. The electric power used for the circulation pumps is also increased because the consumers need more mass flow to cover the heat demand while the supply temperature is kept constant at 60 °C. In the TransiEnt model the supply temperature is set with a trapezoid signal used in the next step of this Tutorial.

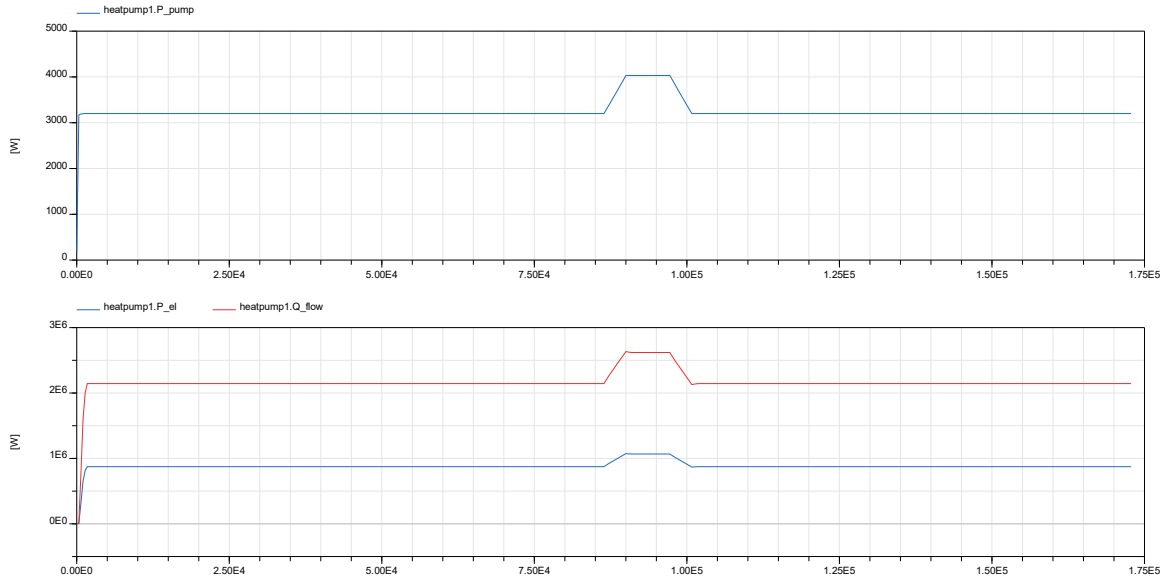

Figure 2: Simulation results step 1

Supply temperature trapezoid input:

Modelica is a standard for dynamic simulation. One important dynamic that is often investigated in DHN is the thermal inertia of the piping system. When changing the supply temperature at a certain point during the simulation, the propagation of the temperature step can be observed. In Figure 3, the supply temperature at the producer (blue) and the supply temperature at the consumer farthest away from the producer (red) can be seen. Therefore, the model can be used to determine the time it takes for a temperature step to reach the consumers.

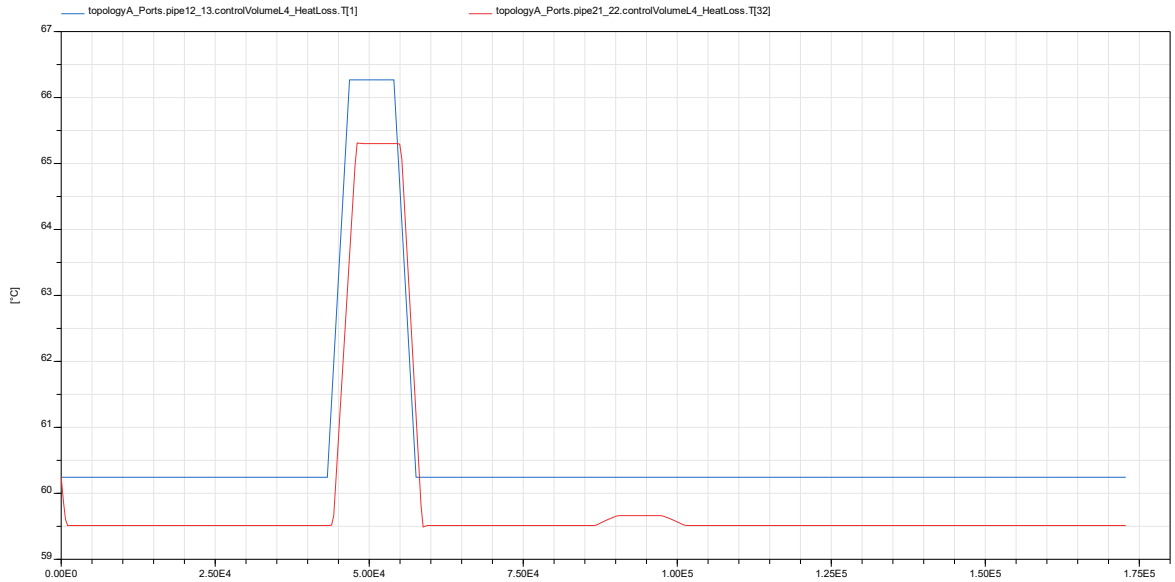

Figure 3: Supply temperature comparison

In Figure 4, the heat losses of the DHN, the electric power of the circulation pumps, the COP of the heat pump and the electric power of the heat pump can be seen. After 43200 seconds, the increase of the supply temperature takes place. This causes three different effects:

- The electric power of the circulation pump decreases because the mass flow rate in the DHN decreases.
- The COP of the heat pump decreases because the efficiency of the thermodynamic process decreases.
- The heat losses in the DHN increase because the temperature difference to the surrounding soil increases.

These effects have been explained in more detail in [1].

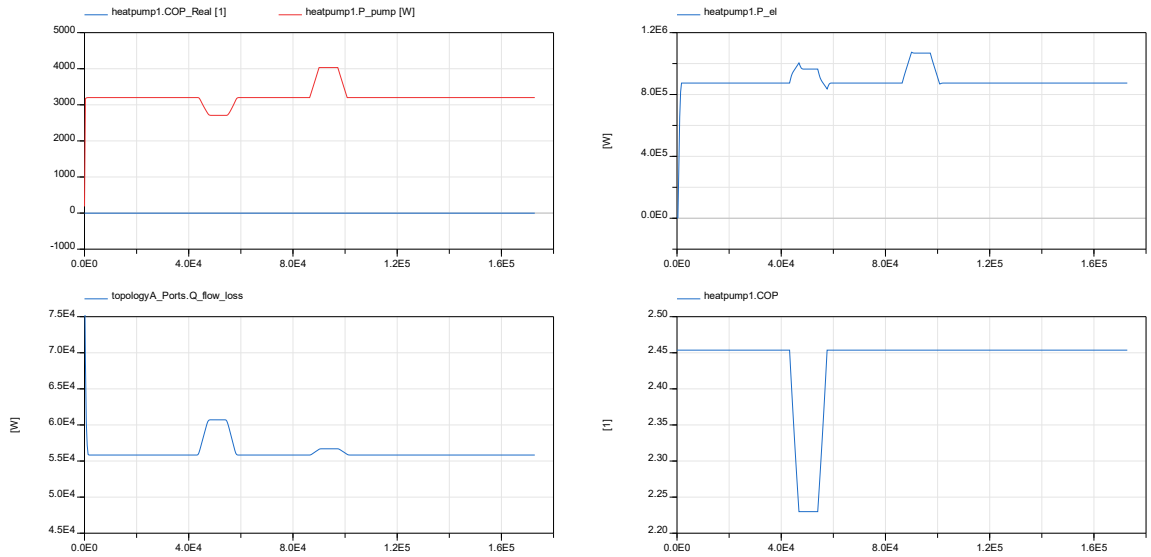

Figure 4: Simulation Results for a trapezoid supply temperature increase

Tutorial model step 2 (pressure control):

So far, the pressure difference at the producer has been set constant to 1.8e5. However, the pressure difference at the producer must be set so that the pressure difference at the consumer furthest away from the producer does not fall below a minimum required. Even so, this pressure difference depends on the mass flow rate inside the DHN and therefore has to be variable to achieve an optimal operation. A simple way to improve the electric power consumption of the circulation pump is to use a PI-Controller for setting the pressure difference at the consumer. For this, drag a model for a PI-Controller inside your simulation model. Use for example`Modelica.Blocks.Continuous.LimPID`.

The PI-Controller needs two inputs that can be set by connecting real expressions. To input the measured signal, the pressure difference at the bad point of the grid must be entered. For TopologyA, this is consumer 9. Therefore,

```
topologyA_Ports.consumer_9.inlet.p -
topologyA_Ports.consumer_9.outlet.p
```

can be entered into the real expression. The other input is the set value for the pressure difference at the critical point. In this example, this could be 0.1e5 Pa. The output of the PI-Controller has to be connected to the dp-input of the heat pump. Moreover, different parameters can be set in the PI-Controller, like the proportional gain k (k=10) and the integrator constant Ti (Ti=1000), as well as upper and lower limits for the output of the controller (y_min=0.0001, y_max=10e5). The full setup of the simulation can be seen in Figure 5.

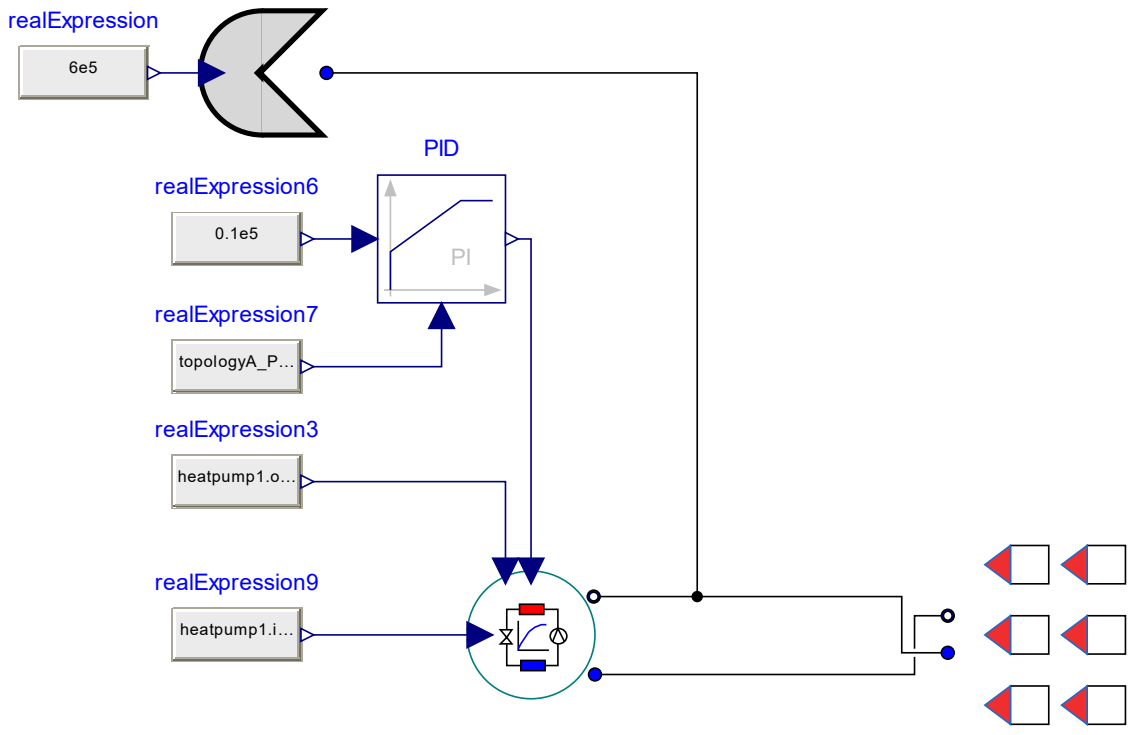

Figure 5: Tutorial model step 2 (TransiEnt.Examples.Heat.DHN_Tutorial_Step2)

In the first plot of Figure 6, the electric power of the circulation pump is presented for both scenarios. One with a constant pressure difference (red) and one with a pressure difference set by

the PI-Controller (blue). It can be concluded that pressure control leads to a lower electric energy consumption of the pump station, although the peak power is the same.

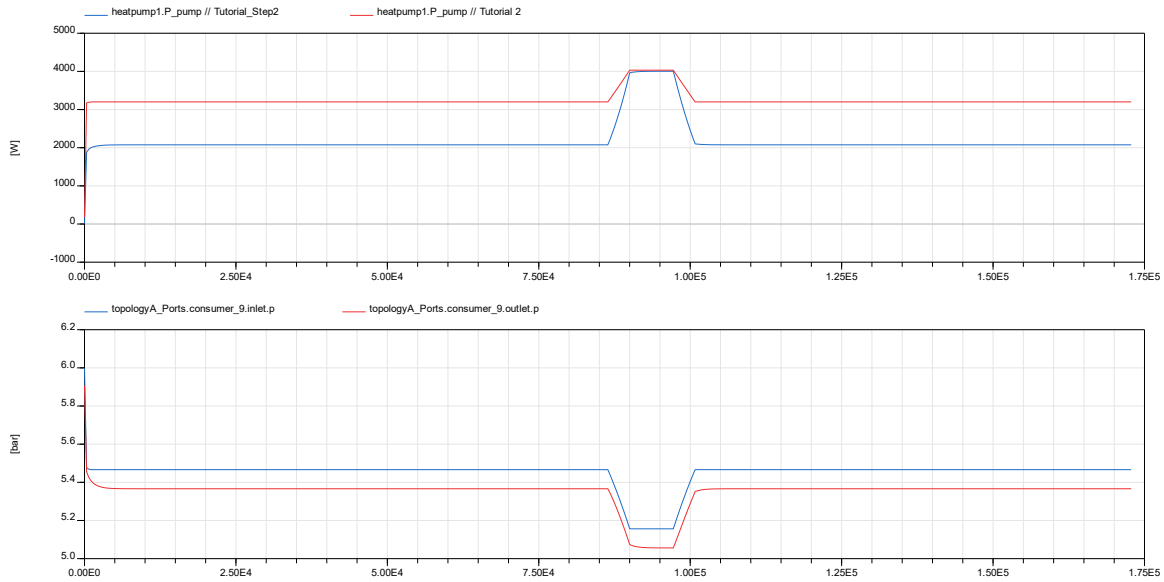

Figure 6: Electric power needed by the circulation pump and pressure difference at bad point controlled by the PI-Controller

Tutorial model step 3 (adding a heat storage):

Next, a heat storage shall be added to the simulation. For the integration of a heat storage four models are needed. A suitable model of a heat storage (`hotWaterStorage_constProp_L4_1`), an adapter for the fluid ports, a heat flow boundary (or a temperature boundary) and two pump models (`SimplePump` and `Pump_dp`) for setting a mass flow and a pressure difference. For this step, an adapter for the fluid ports is needed because currently there are two kinds of fluid ports used in the TransiEnt Library. One fluid port is equal to the fluid port used in the ClaRa Library and features a variable for the mass fraction and a parameter for the medium. The other fluid port only features a pressure, a mass flow and a specific enthalpy. Connecting both fluid ports is not possible, so for the following steps, the adapter has to be used. The inlet of the heat storage (waterPortIn_prod) needs to be connected to the outlet of the pump, while the outlet of the heat storage (waterPortOut_prod) needs to be connected to the inlet of the heat pump. The inlet of the pump model is connected with the outlet of the heat pump. The mass flow input of the circulation pump is set with a real expression (y=18). The outlet of the heat storage has to be connected to another pump model that sets a pressure difference between the supply and return pipes (`Pump_dp`). The dp-input for this pump can be connected to the output of the PI-controller. The dp-input of the heat pump can be set constant (dp=0.3e5). The second inlet of the heat storage (outlet_grid) can be connected to the output of the Topology_A. The heat port of the heat storage has to be connected to a fixed heat flow boundary. The parameter of the heat flow can be set to Q_flow=0 for an adiabatic heat storage. Using a temperature boundary would be better for the determination of the heat losses and also for the numerical efficiency of the simulation, but shall not be considered here. In practice, pipes should be used to connect the heat pump to the heat storage in order to account for occurring pressure losses. However, these pressure losses are neglected for this simulation.

With this new setup, the control of the DHN changed because the heat production is now decoupled from the heat demand. The consumer in the TopologyA sets the mass flow inside the network

dependent on the heat demand and the supply temperature. Therefore, the producer (in this case, the heat storage) has to set a pressure difference. The mass flow between the producer and the heat storage has to be set manually in the pump model.

At last, the TransiEnt SimCenter needs to be dragged into the model. The set-up of the finished model can be seen in Figure 7 and can be found in the TransiEnt Library.

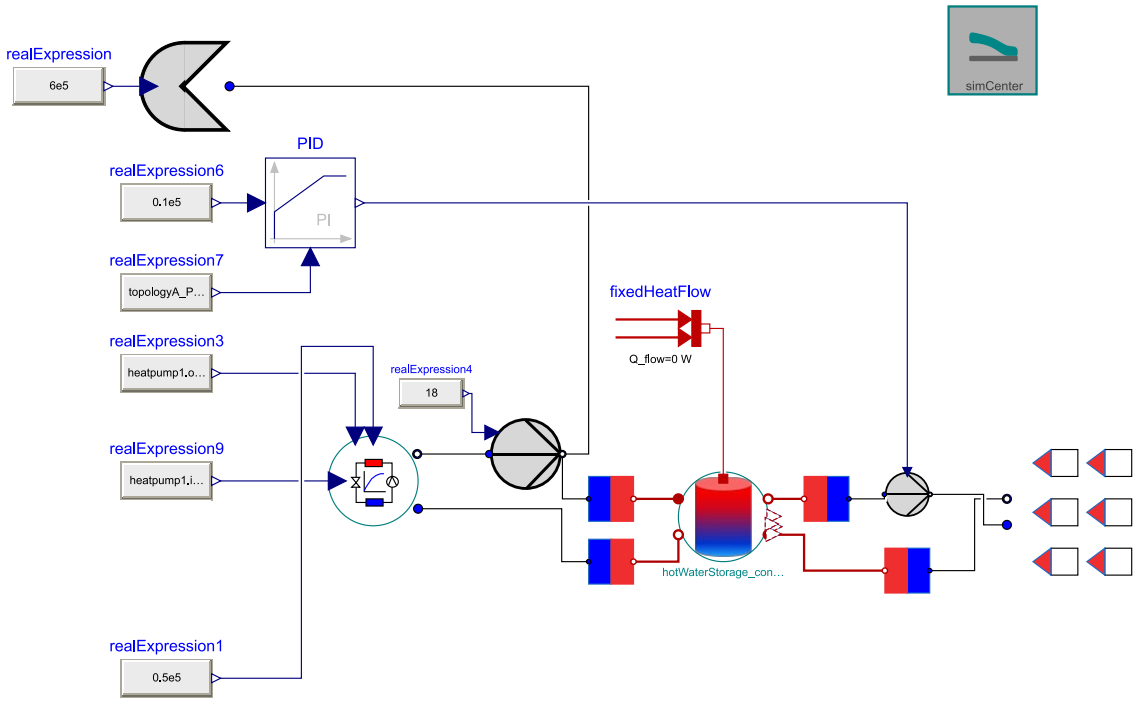

Figure 7: Tutorial model step 3 (TransiEnt.Examples.Heat.DHN_Tutorial_Step3)

In Figure 8, the temperature of the stratified heat storage can be seen. The supply temperature is kept constant during this simulation. At 86400 seconds, the heat demand of the consumer is increased, but the produced heat flow is kept constant. That is why temperatures in the storage decline and increase again after the heat demand decreases.

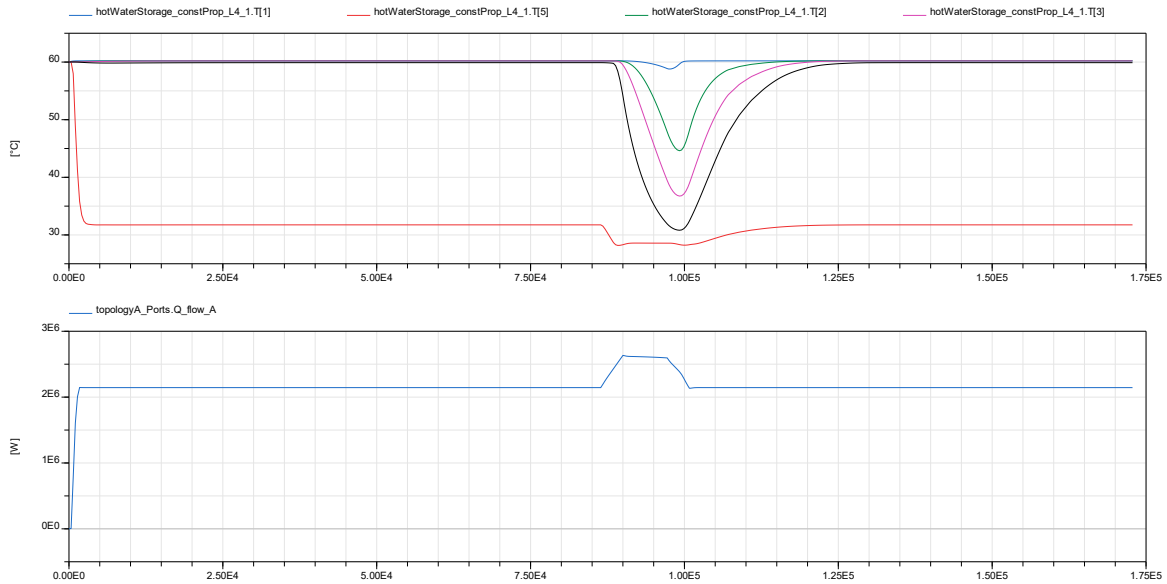

Figure 8: Temperatures inside the heat storage

Tutorial model step 4 (Gas boiler):

In the last step of this Tutorial, another heat producer shall be integrated into the simulation. For this, a gas boiler model (`SimpleBoiler`) has to be dragged inside the model. Moreover, two fluid adapters, a split, a join and another pump for setting a mass flow rate are needed. The pump is connected to the outlet of the gas boiler via the fluid port adapter. The inlets of the join are connected to the two pump models, and the outlet of the join is connected to the inlet of the heat storage. The inlets of the heat pump and the gas boiler are connected to the outlets of the split, while the inlet of the split is connected to the outlet of the heat storage. The mass flow inputs of the two pump models are set to 9 kg/s. Another real expression is needed to set the heat flow rate (Q_flow_set) of the gas boiler. The heat flow rate can be set similarly to the heat flow rate of the heat pump. However, this input needs a negative sign for heat production.

The value of the real expression could look like this:

$$-gasBoiler.inlet.m\_flow * (T\_supply * cf \; - \; inStream(gasBoiler.inlet.h\_outflow))$$

The setup of the model can be seen in Figure 9, and the model can be found in the Examples package of the TransiEnt Library.

For further investigations, other (and larger) DHN topologies can be used, which can be found in `TransiEnt.Grid.Heat.HeatGridTopology`.

To replace the topology, the path of the TopologyA_ports model can be exchanged with a path of a different topology model.

For larger topologies with a higher number of states, it is advisable to activate a sparse solver. In Dymola, this can be done with a command:

```
Advanced.Translation.SparseActivate=true
```

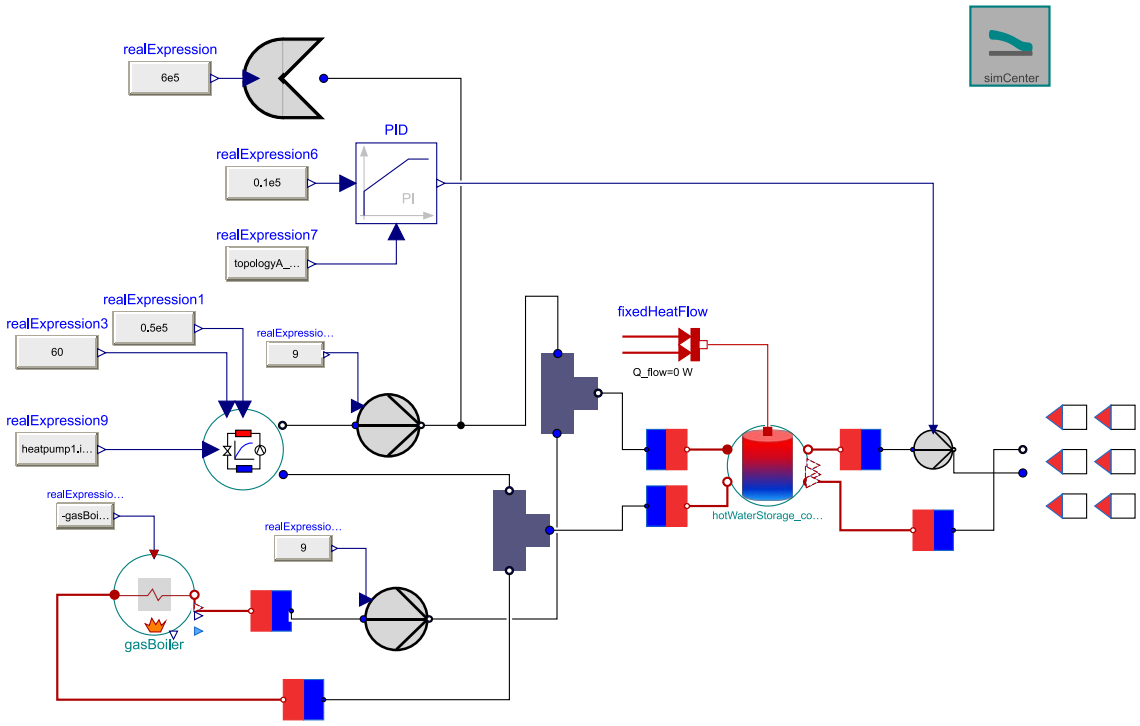

Figure 9: Tutorial model step 4 (TransiEnt.Examples.Heat.DHN_Tutorial_Step4)

References:

[1] Westphal, J.; Brunnemann, J.; Speerforck, A.; Enabling the dynamic simulation of an unaggregated, meshed district heating network with several thousand substations; Energy; Volume 322, 2025, 135434, ISSN 0360-5442, https://doi.org/10.1016/j.energy.2025.135434. (https://www.sciencedirect.com/science/article/pii/S036054422501076X)

[2] Vieth, J.; Westphal, J.; Speerforck, A.; District heating network topology optimization and optimal co-planning using dynamic simulations; Advances in Applied Energy; Volume 19, 2025, 100233, ISSN 2666-7924, https://doi.org/10.1016/j.adapen.2025.100233. (https://www.sciencedirect.com/science/article/pii/S2666792425000277)