ISRN LUTMDN/TMHP-16/5373-SE

ISSN 0282-1990

Framework for district heating optimization

Jasir Sharif
Mohammad Hamid

Thesis for the degree of Master of Science in Engineering
Division of Efficient Energy Systems
Department of Energy Sciences
Faculty of Engineering | Lund University

Framework for district heating optimization

Jasir Sharif
Mohammad Hamid

Aug 2016, Lund

This degree project for the degree of Master of Science in Engineering has been conducted at the Division of Efficient Energy Systems, Department of Energy Sciences, Faculty of Engineering, Lund University, and at Modelon AB.

Supervisor at the Division of Efficient Energy Systems was Associate Professor Urban Persson.

Supervisors at Modelon AB was Stéphane Velut and Per-Ola Larsson.

Examiner at Lund University was Associate Professor Patrick Lauenburg.

Thesis for the Degree of Master of Science in Engineering

ISRN LUTMDN/ TMHP-16/5373-SE

ISSN 0282-1990

Efficient Energy Systems

Department of Energy Sciences

Faculty of Engineering, Lund University

Box 118, 221 00 Lund

Sweden

www.energy.lth.se

Abstract

The objective of this thesis was to develop a flexible optimization framework for district heating networks (DHN) and to introduce pressure in the network models. This was done by creating a network representation (NR) of the networks using the Python package NetworkX. Each node represents a customer or producer and the lines between the nodes are the corresponding pipes.

The method of the project was to first set up a large DHN using the NR based on geometries of the pipes and minimum and maximum supply temperatures for each customer. This model is then simulated to obtain different profiles for the network such as mass flows, loads and supply temperatures. Based on this information, the large network can be aggregated by using a method called “The German method”. This simplified network is then used for optimization, since large networks cannot be optimized due to the large computational time and memory consumption. This gives optimal trajectories of mass flow and supply temperature that are used as inputs in the original, complex network.

The results show that the framework is indeed possible to be used on real cases. By introducing pressure in the framework, one can introduce constraints on differential pressure on different types of nodes and still obtain optimal solutions for the producer unit. The framework is also flexible if the components to be changed contain the same parameters.

This framework could be a suitable tool to be used when performing production planning and model predictive control (MPC) if a discretized optimization procedure is introduced.

Keywords: District Heating, Network Representation, Model Predictive Control

Acknowledgments

We would like to thank Modelon AB for providing the necessary resources and effort for this thesis, as well as a great working environment. Our special gratitude goes to the supervisors at Modelon AB, Stéphane Velut and Per-Ola Larsson, along with our supervisor at LTH, Urban Persson. Lastly, we would like to thank our deputy supervisor Gerald Schweiger from TU-Graz.

Contents

1) Introduction

In future energy systems there will be a high share of fluctuating energy from wind and photovoltaic and until now, the driving force in energy systems is a high demand. This will change when the share of renewable energy will increase, because energy from wind and photovoltaic is limited to certain time frames in which electricity can be produced. The main challenge of future energy supply is to match the available energy from the renewable resources with the energy demand in place, time and quantity. To design intelligent systems, there is a need for dynamic modeling tools to simulate and optimize complex energy systems.

According to SETIS (Strategic Energy Technologies Information System) in terms of utilizing essential resources such as Combined Heat and Power (CHP), geothermal energy, industrial surplus heat, waste and biomass, recent studies show that district heating has the potential to be a key factor. This means that the district heating faces a significant challenge in terms of optimization and further developing the technological concept to increase the efficiency (SETIS, 2013).

The overall objective of the project is to develop a framework for a flexible and compact network representation for different types of pipes, producers, storages and consumer models. For the representation of the district-heating network, a method for aggregating the network to a suitable size is also of interest. The simplified model will then be used to generate a Modelica model useful for optimal control.

The optimal control and the full-scale network representation allows for further studying of production planning.

1.1 Problems to examine

The main goal of the project is to develop a framework for district heating optimization. This means that several steps have to be performed automatically with one main Python script. In order for this to be successful, three problems have been examined:

- Is it possible to optimize differential pressure driven flows in the framework?
- Is it possible to fulfill customers load profiles after aggregation, based on the developed framework?
- Is the framework flexible enough to test differential assumptions, component models and cost functions?

1.2 Limitations

1. Optimization in JModelica.org is a powerful way of dealing with industrially relevant problems such as complex dynamic systems. Constraints are easily set and cost functions can be simply changed by changing a few lines of code. However, small adjustments in both models and constraints make an impact on the result of the optimization. Therefore, the expertise of the supervisors at Modelon was vital in understanding the overall behavior of our results. Several meetings were held to overcome difficulties. In this thesis the main cost function that is minimized is the *supply temperature* and the motivation for this is explained in section "2.4 Motivation of low supply temperature".

2. In the beginning of the project, two aggregation methods were planned to be used – the "Danish" and the "German" method to compare results for optimization. The main priority was to make sure that the framework was operational, and to do this, only one method was required. In the initial phase, the German method was fully implemented and therefore used in the framework. The reason that the Danish method was excluded from the thesis was because the German method can handle loops and several producers.

3. For the framework to be flexible, the translator codes that translate the codes from Python to Dymola (where the simulation is done) have to be more general. The way we implemented the translation codes in this thesis requires that the components to be changed needs to have the same parameters in the modelling. In addition, the networks to be used needs to have the same design.

4. Some production units are assumed to be available at all times in the framework. In reality, some producers are used only when needed. For example, the oil boiler or the accumulator are activated when the heat demand has increased and the combined heat and power cannot fulfill the demands.

1.3 Thesis outline

The thesis begins with a background in section 2) Background, which describes district heating and its main features. This is followed up with a brief explanation of production planning, importance of aggregation and why future district heating systems are focusing on low supply temperature. Thereafter there is an overview of the process conducted in the thesis.

Thereafter, in section 3) Methods, the two aggregation methods used are explained with their corresponding theory. This is then followed up with an illustration of the aggregation procedure. Lastly in this section, the different software and programming languages are briefly described.

In section 4) Network Representation, the network representation is explained, describing the component models and how the translation from different software is performed. In section 5) Process in detail, a detailed explanation of the framework is described.

The last part of the report presents the main results (section 6) Result of the profiles for the full framework. The results are then analyzed in section 7) Discussion, conclusions in section 8) Conclusion and future work aspects in section 9) Future work.

2) Background

In this section, a brief description of how district heating works and the difference between the aggregation methods will be explained.

2.1 District heating

The main goal of district heating is to distribute heat efficiently in urban areas, mainly used for space heating and to heat up tap water. The sources for the heat are usually renewable natural resources that are usually excess heat from power generation, industrial processes, or renewable natural resources that will be lost if not used (Frederiksen & Werner, 2013). The heat is generated at one or a few central heating plants, where the most common types are combined heat and power (CHP), and generating electricity as well. There is also the possibility of using excess heat from industries.

In the grid there are distribution pipes containing water. To stay competitive, the length of the distribution pipes is minimized, a condition that is fulfilled mostly in dense urban areas. The technology behind district heating consists of four major parts:

1. Supply units
2. Distribution networks
3. Customer substations
4. Customer heating and cooling systems

The supply unit can be a thermal power station, a waste incineration plant or an industrial process whereas the distribution network consists of double pipes running side by side entering each building. Usually the pipes are insulated to minimize the distribution heat loss. The customer substation is used to cool down the incoming hot water. If the customer demands increase, the heat addition is also increased in the heat supply unit.

Figure 1 illustrates the concept of a district-heating network.

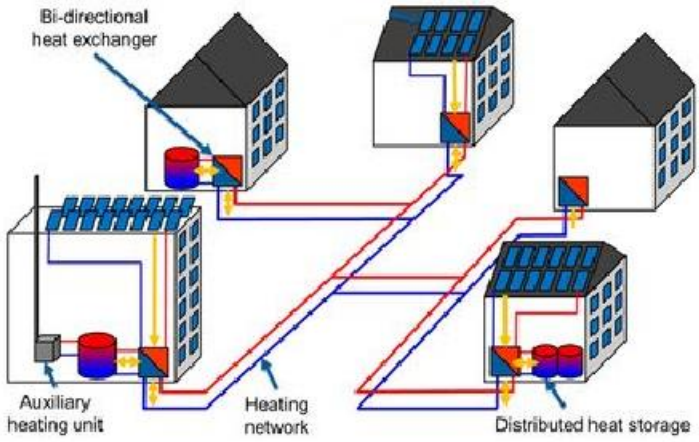

Figure 1 - Typical district heating network. Picture taken from (EESC, 2013).

From Figure 1 it can be seen that the customers are connected to the main pipes in a parallel manner. In the network representation section, the models built for the project look like they are connected in series, but this is not the case. The short pipes between the main pipes and the customers are simply not included in the models graphically.

2.2 Production planning

Imagine a network with a couple of producers such as a combined heat and power (CHP), an oil boiler and an accumulator behaving as a producer. Optimally, the goal is to fulfill the demand of the customers while minimizing the cost, i.e. maximizing the profit. Each time a production unit is started, the cost will increase due to the fuel costs, maintenance costs and also due to the start-up procedure costs for the unit. Basically, to avoid these unnecessary start-ups it is important to include an accumulator, which will be connected to the main producer (CHP). What the accumulator does is storing energy/heat inside of it. Whenever the customer demand increases (before people go to work and after they come home from work), the accumulator assists the main producer by sending out the heat stored inside of it to the customers instead of starting up another unit due to high costs.

But sometimes it is necessary to startup another producer unit if the main producer and accumulator is not enough to supply the customers. To decide which of the multiple deactivated production units to activate, when to start up the deactivated producer and how much this producer will produce is hard to optimize since it is a mixed integer nonlinear problem. To simplify this kind of problem, the problems are divided into two separate optimizing problems: Unit Commitment Problem (UCP) and Economic Dispatch Problem (EDP).

The UCP is the first step done in the optimization and the EDP is the second step done in the optimization. The UCP is constructed using linear models in discrete time only, and the results of the UCP contains production unit statuses, i.e. when the different units are on and off. These results from the UCP is then used as input values to the EDP, which uses physics based models in continuous time. Since the EDP uses nonlinear dynamic optimization techniques, the load of each unit is decided (Larsson, o.a., 2014). However, this is not used in this project.

So by combining these two different optimization results, it is possible to know when each unit will be on and how much heat each unit will produce, i.e. the mixed integer nonlinear problem is solved.

The goal of the production planning is to maximize the profit for the producers, and the mathematical formulation of the EDP in the time interval $[t_0,t_f]$ can be seen in (1):

$$\max_{u} \int_{t_0}^{t_f} \big(R(t) - C(t)\big)dt \qquad (1)$$

where *u* is the selected control signal(s) for the production plant(s), the function R(t) describes the revenue and the function C(t) describes the production cost. As mentioned above, the production cost is a combination of fuel costs, maintenance costs, start-up costs, taxes and similar fees. The revenue for CHP plants is generated by both heat and electricity production.

2.3 Importance of aggregation

A real district heating system could consist of thousands of customers and several producers. If this system were to be simulated to illustrate the dynamics, it would require a huge amount of time. However, if the same network is to be optimized instead regarding supply temperature costs for example, this is not a good way to do it. When aggregating a network, it is proven that it is possible that simplified networks have an 85% saving of computational time for dynamic simulations compared with non-simplified networks (Bøhm, Larsen, & Wigbels, A comparison of aggregated models for simulation and operational optimisation of district heating networks, 2004). The time and memory required for this is simply not possible, therefore, the original network has to be aggregated first and still preserve the majority of its dynamics. Mathematical models of such networks can be applied for this purpose, either for general understanding of the district heating networks or in combination with production planning and optimization (Larsen, Pålsson, Bøhm, & Ravn, 2002)

To do this aggregation there are two main methods that has been used in this thesis: The "Danish" method, which was developed by The Department of Energy Engineering at the Technical University of Denmark together with the Systems Analysis Department at Risø National Laboratory, and the "German" method, which was developed at Fraunhofer Institute for Environmental, Safety and Energy Technology (UMSICHT) in Oberhausen together with the Technical University of Gdansk, Poland, the Gdansk district heating enterprise GPEC and the EVO Energy Supply Company of Oberhausen, Germany.

2.4 Motivation of low supply temperature

Low Temperature District Heating (LTDH) systems are defined differently between countries and have been changed through the years. LTDH in Denmark used to be defined for a supply temperature of 70 °C during the winter and 60 °C during the summer. However, according to the Danish District Heating Association (Olsen, Christiansen, Hofmeister, Svendsen, & Thorsen, 2014), a supply temperature at about 50°C is enough to fulfill the end user´s space heating and domestic hot water in central northern European climates

Therefore, LTDH is defined in accordance with *Dansk Fjernvarme*: “a system of district heat supply network and its elements, consumer connections and in house installations, which can operate in the range between 50-55°C to 60-70 °C supply and 25-30°C to 40°C return temperatures and meet consumer demands for thermal indoor comfort and domestic hot water”.

We want to model a 4th generation system because of several reasons such as a possibility to include renewable source and industrial waste heat and such. Our framework is not directly related to that topic 4th generation. The main objective of this study is to develop a method, where we want to minimize the supply temperature as a means to maximize the profit for daily operation (also for existing networks). Therefore, the framework could provide researchers a tool to test different system-configuration and provide suppliers to improve their existing systems.

There are two main reasons for low temperature district heating systems: (I) higher energy efficiency and (II) a higher share of renewable energy:

I) It ensures efficiency of energy supply since modern buildings set progressively lower limits on energy consumption for heating. In other words: the energy efficiency on the customer side increases considerably. However, the heat losses in the district heating supply network increase when the heat consumption in the buildings decreases. By lowering the supply temperature, the losses can be reduced considerably, which increases the supply side efficiency and competitiveness. It also makes it possibly to supply low energy building in low energy density areas.

II) Geothermal plants are more advantageous to satisfy the base load heat demand for temperatures below 60 °C. The efficiency of heat pumps is increased with lower district heating supply temperature, regardless of heat source. Also it opens up for a broader range of heat pump technologies. The surplus heat frøm industrial processes is better and cheaper the lower the supply temperature. Lastly, flue gas condensation is also a possibility with low supply temperature.

Another important aspect of LTDH systems is that the pipeline thermal stress is reduced, since the distributed temperature gradient along the pipeline is decreased. This minimizes the risk for pipe leakage because of the thermal stress, which makes the maintenance costs lower. Consequently, the lifetime of the DH network is prolonged because of the reduced thermal stress.

2.5 Overview of process

The main goal of this thesis is to optimize a large network regarding different variables such as the supply temperature, mass flows and introduce different constraints on the network. Figure 2 illustrates the overall approach of the thesis:

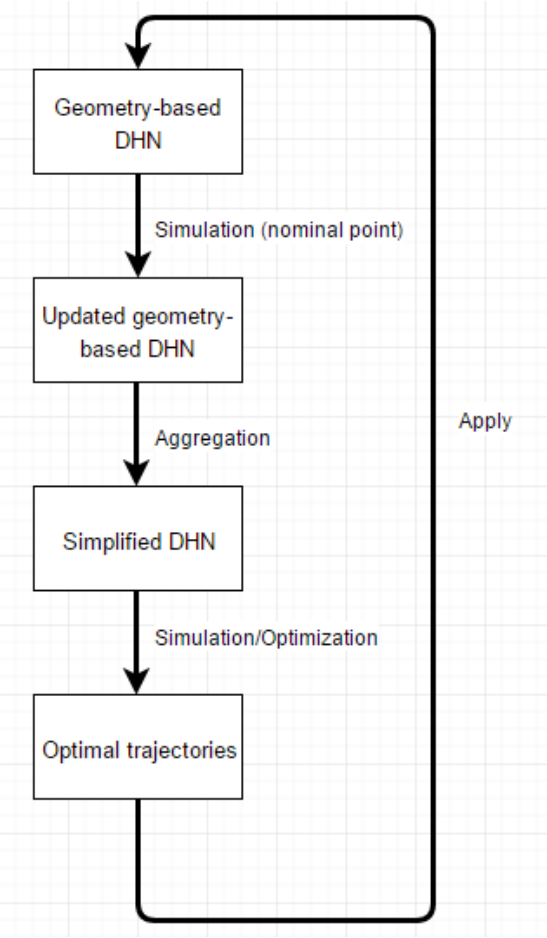

Figure 2 - Overall methodology of the project

The workflow of the thesis is briefly the following:

Firstly, a complex network based on geometries of pipes and limits on supply temperatures to customers is set up. This complex district-heating network represents a real district-heating network in Graz, Austria. Thereafter this complex network is updated with dynamic behavior based on a simulations made in Dymola. This gives a new network, which is then simplified using aggregation (with the Danish or German method). The simplified network is then optimized regarding supply temperature, which in the end will give optimal trajectories for different variables. These trajectories are then to be used in the original, complex network. If all the customers receive their load demand, the loop is successful.

3) Methods

The two aggregation methods used in this thesis are the German and the Danish method and a comparison of the two methods is described in this section. Both methods are defined for a steady state situation but can be used with time variations with a relatively low error (Bøhm, Larsen, & Wigbels, A comparison of aggregated models for simulation and operational optimisation of district heating networks, 2004). The building blocks for the two methods are the same:

- Changing a tree structure into a line structure (branching)
- Removing short branches (serial)

The German method however has a method for simplifying loops in the system.

The symbols used in the formulas are defined in Table 1 and Table 2:

Table 1 - Symbol definitions

Variable	Unit	Definition
$\dot{m}$	kg/s	Mass flow
τ	s	Time delay
L	m	Length of pipe
d	m	Diameter of pipe
V	m^3	Volume of pipe
T	K	Temperature of the medium
Q	W	Heat transfer

Table 2 - Indices used in the aggregation methods

Subscript	Definition
1	Pipe 1 (Before aggregation)
2	Pipe 2 (Before aggregation)
3	Pipe 3 (Before aggregation, for the Danish)
A	Pipe A (After aggregation)
B	Pipe B (After aggregation)
C1, C2, C3	Customers before aggregation
CA, CB, CC	Customers after aggregation

An illustration of branching and serial aggregation is shown in Figure 3, Figure 4 and Figure 5:

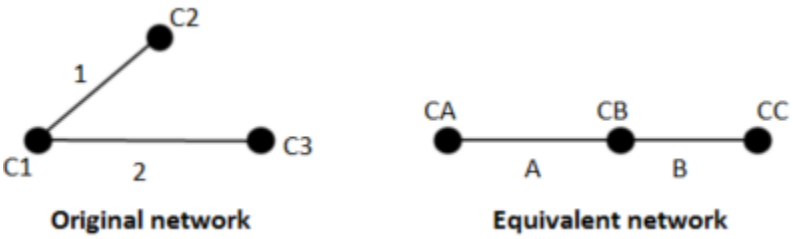

Figure 3 – Branch aggregation for Danish and German method

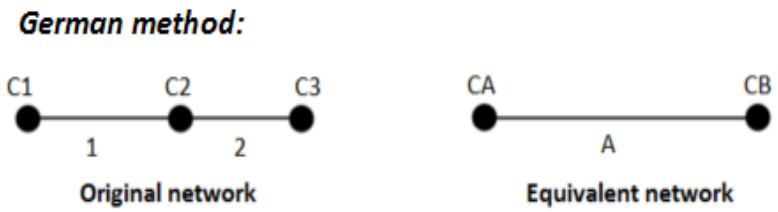

Figure 4 - German serial aggregation

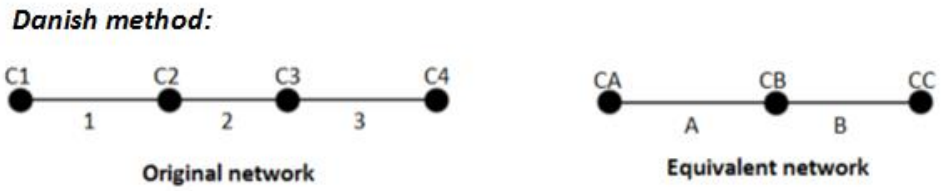

Figure 5 – Danish serial aggregation. NOTE: Each line represents 2 pipes (supply and return)

It is important to know that each line in the representation consists of 2 pipes (supply and return). The branching of the nodes is equivalent for the methods while the serial aggregation is different: The German method replaces two branches by one, whereas the Danish method replaces three branches by two. In the German method, the pressure drop is considered (not in the Danish) while the heat loss is considered in the Danish. The main differences between the two methods are shown in Figure 4 and Figure 5.

In the original network in Figure 3, we have customers C1, C2 and C3 and pipes 1 and 2. When you aggregate these branches into one structural line like in the equivalent network, you will get three "new" customers CA (earlier C1), CB (earlier C2) and CC (earlier C3) and two new pipes A and B with new geometries (volumes, mass flows, etc.).

In the original network in Figure 4 and Figure 5, you will have customers C1, C2, C3 (and C4 in Figure 5) and pipes 1, 2 (and 3 in Figure 5). When you aggregate the line structure into a smaller one, the load of the removed customer will be divided between the remaining customers which are CA, CB (and CC in Figure 5) and the remaining pipe/pipes A (and B in Figure 5) will be given new geometries and dynamics.

Table 3 shows the preservations of the two methods.

Table 3 –The significant preservations of the two methods.

Method	**Danish**	**German**
Preserves	Heat loss	Pressure drop
Can handle		Loops

The pressure drop in the pipes is considered in the German method in the sense that it adjusts the surface roughness or additional resistance caused by elbows etc. for each pipe in the aggregated grid. This preserves the pressure in each node.

The German method can also handle loops in the system in two ways:

1. Transforming a loop into serial pipes
2. Splitting a loop into two serial pipes and one branch

For case 1, small errors have to be accepted.

3.1 The German method

When working with the German method, the system will initially try to find three consumer-nodes in serial (see Figure 4)

When it has found three consumer nodes right next to each other, the system will serial aggregate the three nodes into two nodes. It will do so for the entire network. When the system eventually has aggregated all the "three consumer neighbors" in the network, it will look for two parallel branches and then aggregate these two into one branch. After the branch aggregation, the system will get more "three consumer neighbors" and it will aggregate them too into two nodes, and so on. The system will alternate between serial aggregation and branch aggregation until the entire system is fully aggregated.

3.1.1 Branch aggregation

When comparing the branch aggregation between the German method and the Danish method, the process is almost the same. Both methods convert the "parallel" pipelines into one serial pipeline, as seen in Figure 3. The only difference between the methods is the calculation of the geometries.

The new mass flows in the pipes are calculated below in (2) and (3) (Loewen, 2001):

$$\dot{m}_A = \dot{m}_1 + \dot{m}_2 \quad (2)$$

$$\dot{m}_B = \dot{m}_2 \quad (3)$$

where $\dot{m}_A$ and $\dot{m}_B$ represents the newly calculated mass flows in pipe A respectively pipe B in the equivalent network, and $\dot{m}_1$ and $\dot{m}_B$ represents the mass flows in pipe 1 respectively pipe 2 in the original network in Figure 3.

The delay τ is calculated in terms of volume in the pipe *V*, density of the object in the pipe (water) ρ and mass flow in the pipe $\dot{m}$. This can be seen in (4) and (5):

$$\tau_1 = \frac{V_1 \cdot \rho}{\dot{m}_1} \quad (4)$$

$$\tau_2 = \frac{V_2 \cdot \rho}{\dot{m}_2} \quad (5)$$

where index 1 and 2 represents the parameters for the corresponding pipe in Figure 3.

These parameters are used when calculating the new lengths of the pipes. (6) and (7) denotes the lengths of the new pipes:

$$L_A = L_2 \cdot \frac{\tau_1}{\tau_2} \quad (6)$$

$$L_B = L_2 - L_A \quad (7)$$

where L_A represents the length of the new pipe A and L_B represents the length of the new pipe B in the equivalent network in Figure 3.

The last geometries that need to be calculated is the inner diameter and volume of the new pipes (pipes A and B). By using the volumes, mass flows and delay factors of the two old pipes (pipes 1 and 2), the inner diameters of the new pipes can be calculated as seen below in (8) and (9):

$$d_A = \sqrt{\frac{4 \cdot V_1 \cdot (1 + \frac{\dot{m}_2}{\dot{m}_1})}{\pi \cdot L_2 \cdot \frac{\tau_1}{\tau_2}}} \quad (8)$$

$$d_B = \sqrt{\frac{4(V_2 - V_1 \frac{\dot{m}_2}{\dot{m}_1})}{\pi \cdot L_2 \cdot (1 - \frac{\tau_1}{\tau_2})}} \quad (9)$$

where d_A is the inner diameter for the new pipe A and d_B is the inner diameter for pipe B.

By using these new geometries of the new pipes A and B, the new volumes are then calculated in (10) and (11):

$$V_A = \frac{{d_A}^2 \cdot \pi}{4} \cdot L_A \quad (10)$$

$$V_B = \frac{{d_B}^2 \cdot \pi}{4} \cdot L_B \quad (11)$$

where V_A is the volume of the new pipe A and V_B is the volume of pipe B in the equivalent network in Figure 3.

Since nodes CA respectively CC in the new equivalent network in Figure 3 still has the same supply temperatures as nodes C1 respectively C3 in the original network, it is assumed that the middle node CB also has the same supply temperature as node C2 in the original network as can be seen in (12).

$$T^{CB}_{in\_supply} = T^{C2}_{in\_supply} \quad (12)$$

where $T^{CB}_{in\_supply}$ is the supply temperature for customer CB in the equivalent network in Figure 3.

The same applies for the input temperature in return pipe A from node CB and for the output temperature from supply pipe A to node CB, in (13) and (14):

$$T^{A}_{in\_return} = T^{1}_{in\_return} \quad (13)$$

$$T^{A}_{out\_supply} = T^{1}_{out\_supply} \quad (14)$$

where $T^{A}_{in\_return}$ is the input temperature to the return pipe in pipe A and $T^{A}_{out\_supply}$ is the output temperature from the supply pipe in pipe A.

The output temperature from return pipe A must be calculated, which can be seen below in (15):

$$T^{A}_{out\_return} = \frac{T^{1}_{out\_return} \cdot \dot{m}_1 + T^{2}_{out\_return} \cdot \dot{m}_2}{\dot{m}_1 + \dot{m}_2} \quad (15)$$

where $T^{A}_{out\_return}$ is the output temperature from the return pipe in pipe A.

The temperature parameters for pipe B looks a lot like the temperature parameters for pipe 2 with the exception of the supply temperature into pipe B, see (16), (17), (18) and (19) for pipe B:

$$T^{B}_{in\_supply} = T^{A}_{out\_supply} \quad (16)$$

$$T^{B}_{in\_return} = T^{2}_{in\_return} \quad (17)$$

$$T^{B}_{out\_supply} = T^{2}_{out\_supply} \quad (18)$$

$$T^{B}_{out\_return} = T^{2}_{out\_return} \quad (19)$$

These are the equations used for aggregating the parallel pipes into serial.

3.1.2 Serial aggregation

The illustration for serial aggregation can be referred to Figure 4. When serial aggregating, the system will be looking for three consumers as neighbors. When it does, it will aggregate the system from three nodes to two nodes. The parameters for the new pipes and new consumer nodes will be calculated below.

The geometries volume *V*, length *L* and inner diameter *d* of the new pipe will be calculated below in (20), (21) and (22):

$$V_A = V_1 + V_2 \quad (20)$$

$$L_A = L_1 + L_2 \quad (21)$$

$$d_A = \sqrt{\frac{{d_1}^2 \cdot L_1 + {d_2}^2 \cdot L_2}{L_1 + L_2}} \quad (22)$$

where V_A is the volume of the new pipe A, L_A is the length of pipe A and d_A is the inner diameter of pipe A in the equivalent network in Figure 4.

The mass flow in the new pipe will be calculated in (23):

$$\dot{m}_A = \frac{V_1 + V_2}{\frac{V_1}{\dot{m}_1} + \frac{V_2}{\dot{m}_2}} \quad (23)$$

where $\dot{m}_A$ is the new mass flow in pipe A in Figure 4.

Since the middle consumer node C2 in the original network is removed from the system, its load (and therefore its mass flow) will be divided between the two neighboring consumer nodes, C1 and C3 as CA and CB. That is why it is necessary to calculate the new mass flows at the two new consumer nodes, as we can see in (24) and (25):

$$\dot{m}_{CA} = \dot{m}_{C1} + \dot{m}_1 - \dot{m}_A \quad (24)$$

$$\dot{m}_{CB} = \dot{m}_{C3} + \dot{m}_A - \dot{m}_2 \quad (25)$$

where $\dot{m}_{CA}$ is the new mass flow in customer CA and $\dot{m}_{CB}$ is the new mass flow in customer CB in the equivalent network in Figure 4.

The temperatures for the pipe and consumer nodes and load for the consumer nodes will be calculated below (26) - (33):

$$T^{A}_{out\_supply} = T^{3}_{in\_supply} \quad (26)$$

$$T^{A}_{in\_supply} = T^{1}_{in\_supply} \quad (27)$$

$$T^{CA}_{in\_supply} = T^{1}_{in\_supply} \quad (28)$$

$$T^{CB}_{in\_supply} = T^{3}_{in\_supply} \quad (29)$$

where $T^{CA}_{in\_supply}$ is the input supply temperature in customer CA and $T^{CB}_{in\_supply}$ is the input supply temperature in customer CB in Figure 4.

$$Q_{CB} = \dot{m}_{CB} \cdot c_{P,w} \cdot (T^{C3}_{in\_supply} - T^{CB}_{out\_supply}) \quad (30)$$

$$Q_{CA} = Q_{C1} + Q_{C2} + Q_{C3} - Q_{CB} \quad (31)$$

where $c_{P,w}$ is the specific heat capacity for water, Q_{CB} is the new heat load for customer CB and Q_{CA} is the new heat load for customer CA from Figure 4.

$$T^{CA}_{out\_supply} = T^{C1}_{in\_supply} - \frac{Q_{CA}}{\dot{m}_{CA} \cdot c_{P,w}} \quad (32)$$

$$T^{A}_{out\_return} = \frac{T^{1}_{out\_return} \cdot \dot{m}_1 + T^{C1}_{out\_supply} \cdot \dot{m}_{C1} - T^{CA}_{out\_supply} \cdot \dot{m}_{CA}}{\dot{m}_A} \quad (33)$$

3.2 The Danish method

In this section, the Danish method will be described.

3.2.1 Branch aggregation

In the Danish method, when collapsing the network tree into one branch, there are a few assumptions that are considered (Bøhm & Larsen, Simple models of district heating systems for load and demand side management and operational optimisation , 2004):

- The water volume is conserved (34):

$$V_A + V_B = V_1 + V_2 \quad (34)$$

- The total mass flow in the customers is conserved (35):

$$\dot{m}_{CA} + \dot{m}_{CB} + \dot{m}_{CC} = \dot{m}_{C1} + \dot{m}_{C2} + \dot{m}_{C3} \quad (35)$$

- The heat load is conserved (36):

$$Q_{CA} + Q_{CB} + Q_{CB} = Q_{C1} + Q_{C2} + Q_{C3} \quad (36)$$

- The heat loss from the pipes is conserved (37):

$$Q_A + Q_B = Q_1 + Q_2 \quad (37)$$

But in order to obtain the equivalent network, the creators of the Danish method choose branch B to equal the last part of branch 2 (as can be seen in Figure 3) and the mass flow in branch B to equal the mass flow in branch 2, see (38) - (41)

$$d_B = d_2 \quad (38)$$

$$D_B = D_2 \quad (39)$$

$$\varphi_B = \varphi_2 \quad (40)$$

$$\dot{m}_B = \dot{m}_2 \text{ (both in the heat exchanger and DH pipe)} \quad (41)$$

These choices above regarding branch B could have been made in other ways, but these rather simple choices make it possible to define an equivalent network.

When calculating the mass flow for branch A, we must consider the equivalent network. (42) and (43) is derived from different aspects:

$$\dot{m}_A^{HE} = \dot{m}_1^{HE} \quad \text{(in the heat exchanger)} \quad (42)$$

$$\dot{m}_A = \dot{m}_1 + \dot{m}_2 \quad \text{(in the DH pipe)} \quad (43)$$

The difference between the above s is that the mass flow of the heat exchanger is considered to be the mass flow entering the house of consumer 1, while the mass flow of the DH pipe is considered to supply all consumers on that branch (which in this example is consumer 1 and 2).

Before continuing the calculation of the other variables, we must define the auxiliary variables, see s (44)-(47): (page 996 in (Larsen, Pålsson, Bøhm, & Ravn, 2002))

$$\alpha = {}^{m_2}/_{m_1}, \quad \alpha > 0 \quad (44)$$

$$\beta = 1 + \alpha, \quad \beta > 1 \quad (45)$$

$$\varphi = {}^{C^w}/_{(C^w + C^p)}, \quad 0 < \varphi < 1 \quad (46)$$

$$\psi = {}^{\varphi_2}/_{\varphi_1}, \psi > 0 \quad (47)$$

where α is the ratio between the mass flows in pipe 2 and pipe 1 in Figure 3, C^W is the heat capacity per meter for water and C^p is the heat capacity per meter for the steel pipe.

Now we can calculate the lengths and diameters for the branched pipes, see s (48) - (51): page 1002-1003 (Larsen, Pålsson, Bøhm, & Ravn, 2002)

$$L_B = L_2(1 - \gamma\psi) \quad (48)$$

$$d_A = 2\sqrt{\frac{\beta(1+\alpha\psi)A_1A_2}{(A_2+A_1\alpha^2\psi)\pi}} \quad (49)$$

$$D_A = d_A\sqrt{1 + (\frac{\rho^w c^w}{\rho^p c^p})(\frac{1-\varphi_A}{\varphi_A})} \quad (50)$$

$$\frac{\rho^w c^w}{\rho^p c^p} = 0.001089 \quad (51)$$

where D_A is the new outer diameter of the pipe A in Figure 3, ρ^W is the density for water, ρ^p is the density for steel pipe, c^W is the specific heat capacity for water and c^p is the specific heat capacity for steel pipe.

3.2.2 Serial aggregation

The aggregation structure for the Danish method is primarily based on aggregating the branches into one line/branch structure so all the customer nodes will be positioned close to each other separated by short pipes. To further simplify the network, the nearby nodes with small pipe lengths can be collapsed as seen in Figure 5.

If the method is instead used for longer pipes, then the network can be reduced to a few branches/pipes or just one branch/pipe and the quality of the resulting network will be reduced.

In Figure 5, we can see the original network to the left transform into the resulting equivalent aggregated network to the right by removing a pipe and collapsing the two neighboring nodes. Pipe 2 is divided in two parts as we can see in the figure. Pipe A symbolizes pipe 1 and the first part of pipe 2 and pipe B symbolizes the other part of pipe 2 and pipe 3.

When collapsing the nearby nodes (removing the pipe between the nodes) in a line structure, a few physical properties are conserved, see Table 4.

Table 4 - Quantities conserved by the Danish method

Physical properties	**Conserved**
Pipe length	Yes
Pipe inner diameter	No
Water volume	Yes
Delay	Yes
Mass Flow	Yes
Heat load	Yes
Heat loss from supply pipe	Yes
Heat loss from return pipe	Yes

3.3 Illustration of the aggregation in Python

In this section the two methods will be illustrated directly from Python.

3.3.1 The Danish method

In this method, all branches are removed which will yield a single line with all customers. Thereafter the customers are removed to an amount described by the user. Figure 6 illustrate an example of a simplified network to easily understand the function of the Danish method.

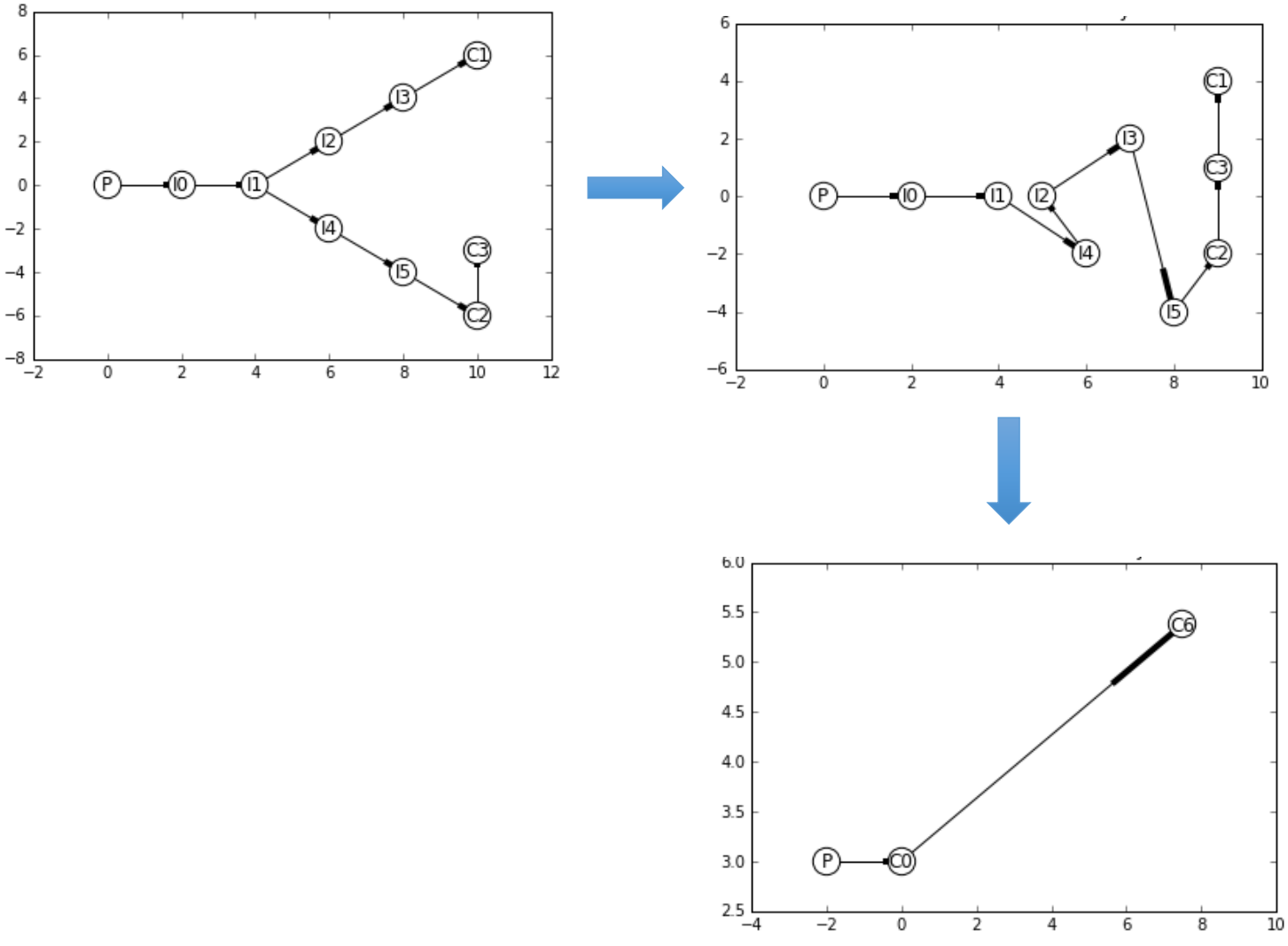

Figure 6 - Aggregation illustration of the Danish method

3.3.2 The German method

Firstly, the complex system is serially lumping the two branches. Thereafter the branch is lumped into a serial. Lastly, the line is lumped into the amount of customers remaining determined by the user (in this case 1). Figure 7 illustrate an example of a simplified network to easily understand the German method. The German method can also handle loops as illustrated in Figure 8.

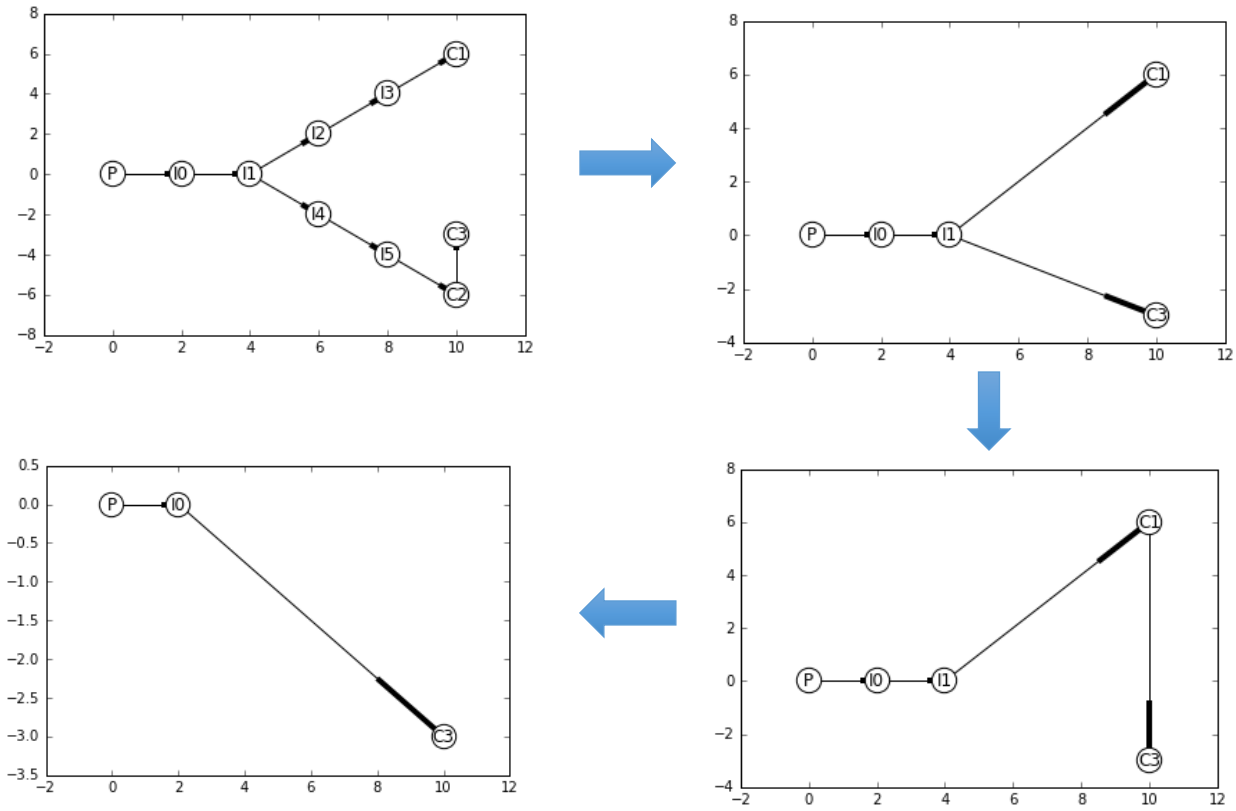

Figure 7 - Aggregation illustration of the German method

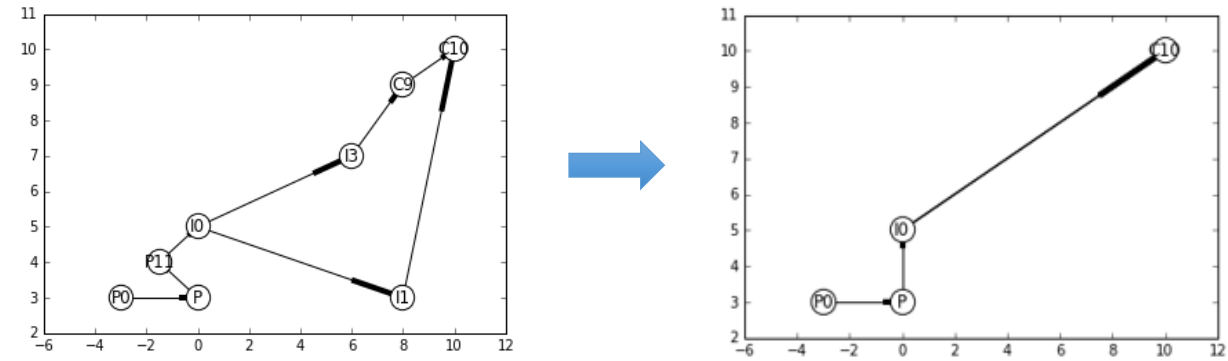

Figure 8 - Handling loops (German)

3.4 Software and programming languages

In this part, the different software programs and languages that have been used in the work will be described.

3.4.1 Modelica

The programming language Modelica is an object-oriented language of large and complex physical systems and is based on differential algebraic. It is suited for multi domain modeling such as: mechatronic systems within automotive, aerospace and robotics applications. Modelica has been designed to allow for codes to be generated automatically during modeling, which makes the modeling effort reduced significantly (Modelica, 2016).

3.4.2 Optimica

Modelica is extended by Optimica, which enables formulation of dynamic optimization problems based on Modelica models. The optimization class specifies properties such as: cost functions and constraints on the system. This extension is supported by the "Optimica compiler", which has been successfully used in different case studies (Åkesson, 2008).

3.4.3 JModelica.org

JModelica.org is developed by Modelon AB and provides a Modelica and Optimica compiler. It is an open source platform and used for optimization, simulation and analysis of complex dynamic systems. JModelica.org also serves as an interface to simulate compiled models from Python. For the optimization class, it uses collocation to transfer problems to nonlinear programming (NLP), and optimizes them using IPOPT (Modelon, JModelica.org User Guide, 2015).

3.4.4 Dymola

Dymola is a program used for modeling and simulating integrated and complex systems. The dynamic behavior of complex systems is possible to simulate among many engineering fields, such as mechanical, electrical and thermal systems. It enables users to build more integrated models that resembles reality more accurate (Modelon, Dymola, 2015). Modelica is the language used in Dymola.

3.4.5 Python

Python is a dynamic and high level programming language. Its syntax makes it possible to express concepts in a shorter manner than other languages such as C++ or Java. Python supports object oriented programming and features a large and comprehensive standard library (Python, 2016).

4) Network Representation

The network representation describes the structure, i.e. the connections between the components: consumer, producer and pipes but not the components themselves. That is why it is flexible. In addition, nothing in the network is “hard coded”, which means that all the variables changes accordingly. For example, if we set up a new network purely based on geometries, heat losses and mass flows are calculated in the model.

4.1 Component models

From Modelon’s DHN library in Dymola, there were a few cases from an Uppsala network (Larsson, o.a., 2014) that we studied. These included a complex producer unit, several customers, accumulators and a splitter (this component sets how much of the flowing water should be divided to each customer). The systems however were not pressure based, which means that the mass flow distributed along the system did not depend on the pressure in different nodes. Therefore, they included a mixer to simply set the mass flow to be distributed according to some fraction. In reality, it is the pressure in the nodes that determines how much of the mass flow enters and leaves the different nodes, therefore we implemented a new network representation with pressure included.

The first model is the simulation model, which is illustrated in Figure 9 (producer). The optimization model (Figure 10) has a very similar design as the simulation model. The difference between them is the producer, i.e. the input to the integrator at the beginning of the system. The simulation model includes a simple temperature reference of 90 degrees Celsius, which is compared to the first supply pipe temperature, whereas in the optimization model the comparator is replaced with two “free variables” called *TsupplyDer* and *PressureDer,* which are “free variables” for the temperature and pressure. These variables are varied in the optimization solver to fulfill all requirements.

The comparator used in the simulation input gives a differential signal to the integrator. If the temperature of the pipe is too low, there will be a positive signal to the integrator, which will then increase. If the temperature of the pipe is too high, there will be a negative signal to the integrator, which will then decrease. This sets the correct temperature according to refTemp to each customer. The free variable component used in the optimization input is simply an input without any value. When the solver tries to find an optimal solution, these are the only variables that the solver can adjust to fulfill the requirements.

Another difference in the models is between the pipes. The optimization pipes include “delay elements”. The reason for this is because that JModelica.org does not include delays in the same way like in Dymola and therefore these have to be included externally for the solver to consider them in the optimization. Figure 11 and Figure 12 illustrates the differences between the simulation and optimization pipes.

The pipe model consists of several parameters that define it. The most important one is the “Number Of Volume Segments”, which is included in the “transportPipe model”. The higher this value is, the more accurate the pipe behavior is described, since its calculations are based a finite element volume. Also, there is a heat loss model (the orange block). This is simply a model that calculates the heat losses based on the geometries of the pipe along with the outdoor temperature. A more detailed illustration of the pipe models is shown in Figure 13 and Figure 14.

The heat unit is used to heat up the water in order to fulfill the customer load demand. The return temperature of the water is 40 °C in this model and this is heated up to approximately 90 °C. Since the pipes have heat losses included, one has to compensate for this when setting the parameters for the heat unit.

4) Network Representation

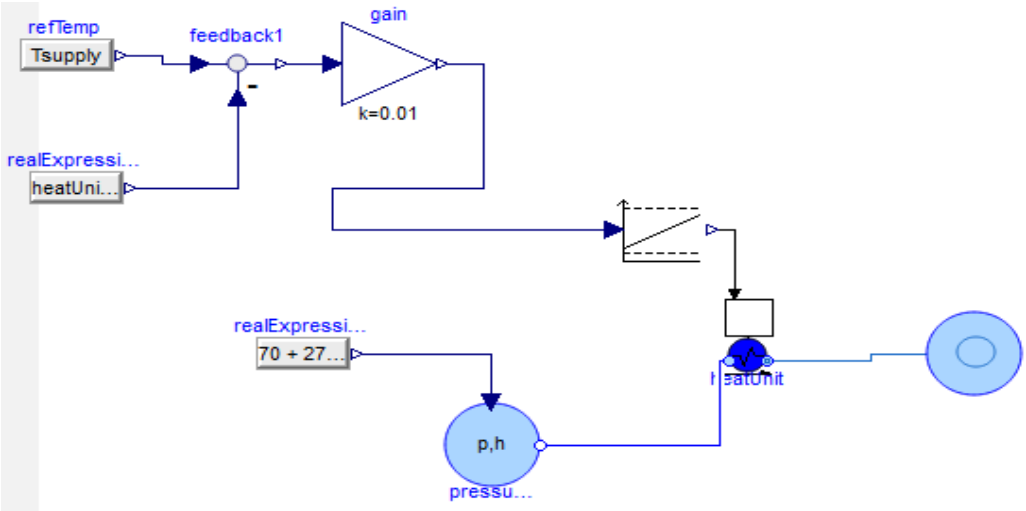

Figure 9 - Simulation model of a producer unit supply temperature control

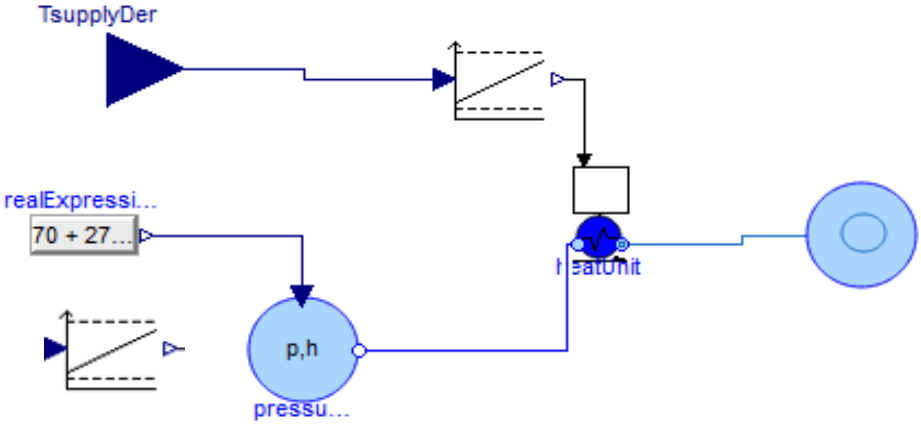

Figure 10 - Optimization model if a producer with two degrees of freedom: supply temperature and pressure

4) Network Representation

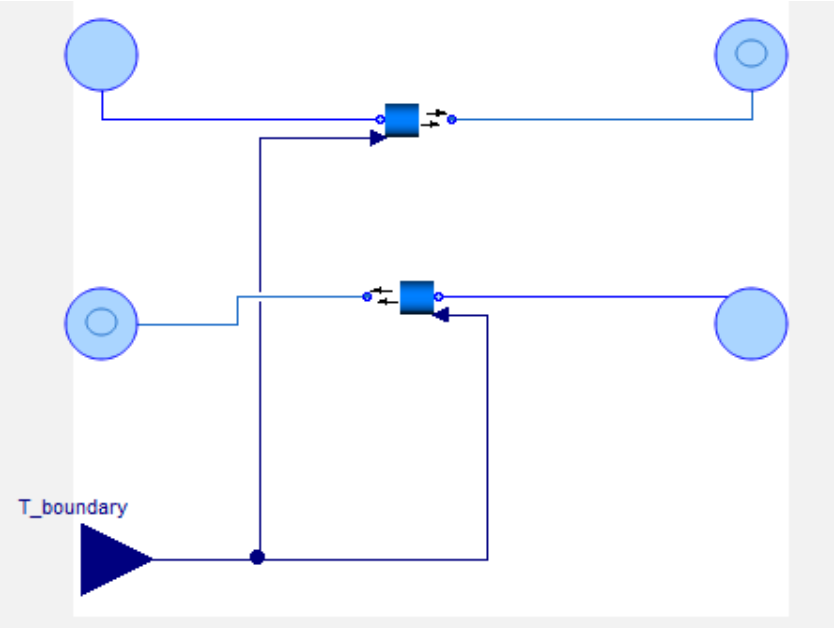

Figure 11 – Simulation twin pipe (supply and return)

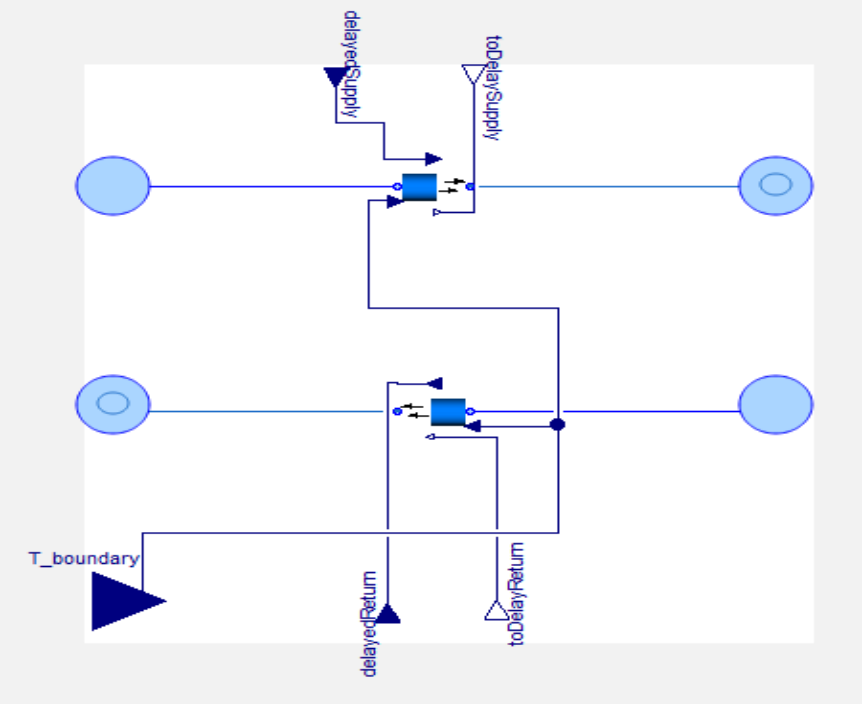

Figure 12 - Optimization twin pipe (supply and return

4) Network Representation

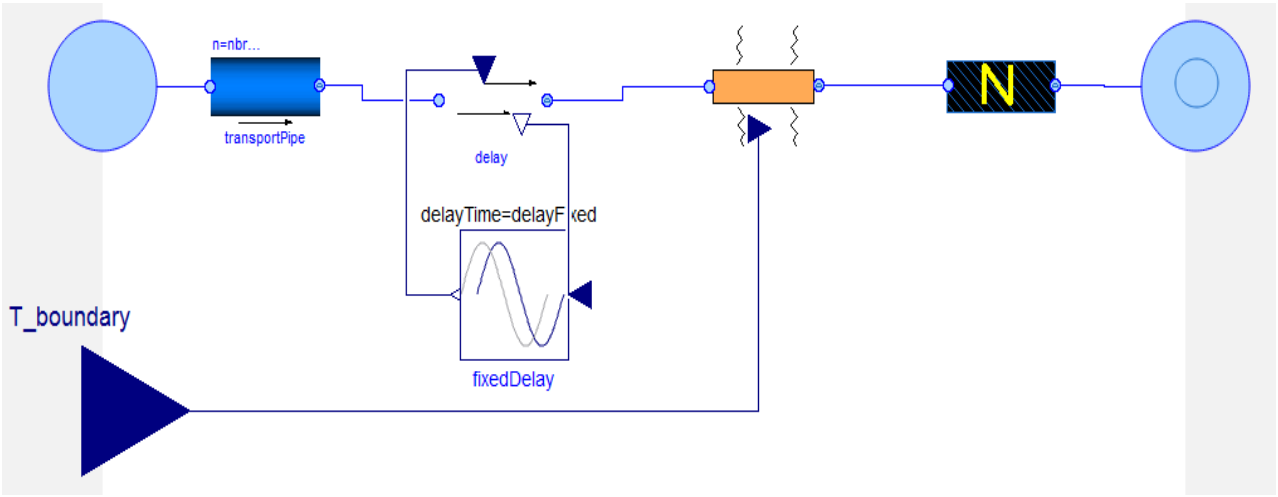

Figure 13 - Detailed simulation single pipe

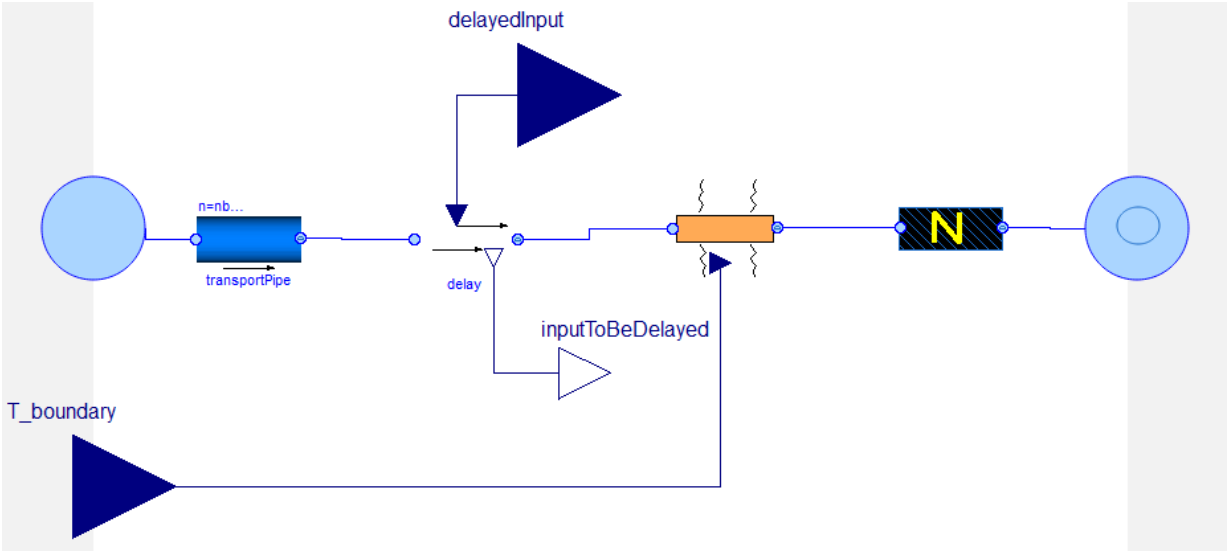

Figure 14 - Detailed optimization single pipe

The ambient temperature is simply the outdoor temperature and is typically illustrated in Figure 15:

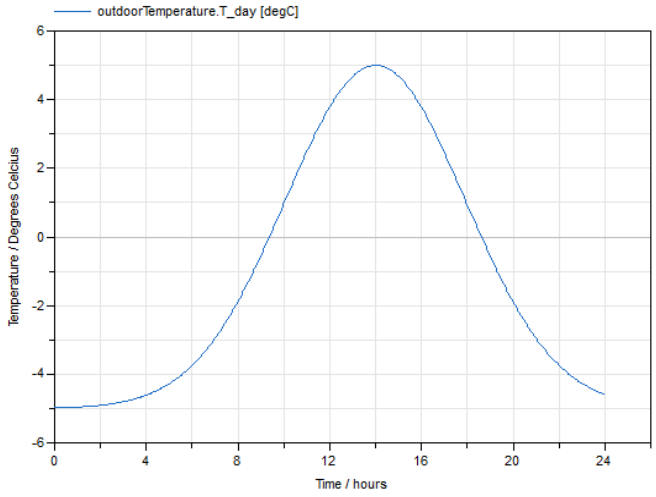

Figure 15 - Ambient temperature based on a winter day

This simply illustrates a typical day during the winter, with the highest temperature around noon and lowest in the early morning and late at night. Each customer could be modeled with a return temperature that varies with the outdoor temperature in reality. However, in these tests a constant value of 40°C is used for simplicity.

The customers are defined to have a fixed load and supply temperature and based on this the system calculates the necessary mass flow. The customer load is based on the doublePeak block. The doublePeak illustrates a typical load demand, with one peak in the morning and one in the evening and is illustrated in Figure 16. This load demand is an own made-up load demand based on a winter day, which we are using for all our customers in our different cases.

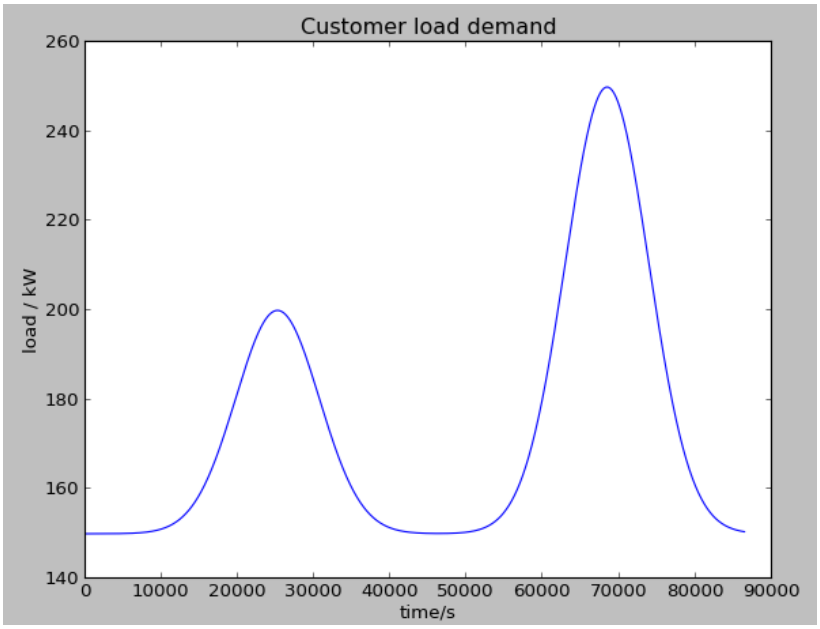

Figure 16 - Customer load prediction for a 24-hour time period (86400 s)

4.2 Python using NetworkX

To represent a real network, the customers and producers should be regarded as nodes and the pipes as edges between these nodes. In Python, there is a free available package called NetworkX. From the official website, it is described as: "NetworkX is a Python language software package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks" (NetworkX, 2016). The package is user-friendly and extremely powerful in terms of manipulating different node structures and complex networks. Firstly, a graph is created:

```
import networkx as nx
G=nx.Graph()
```

This introduces a "blank sheet to work on". Secondly, we add nodes to our graph:

```
G.add_node(1) #add node 1 to the graph
G.add_node(2) #add node 2 to the graph
```

Next, a pipe is added or an "edge" between these nodes:

```
G.add_edge(1,2) #add an edge between node 1 and 2
```

In order to plot this graph, the nodes need to have a position in the xy-plane:

```
G.node[1]['pos'] = (0,0) #set position for node 1
G.node[2]['pos'] = (1,1) #set position for node 2
```

The simple network representation can then be plotted by writing:

```
G.draw() #draw the graph
```

The graph is illustrated in Figure 17.

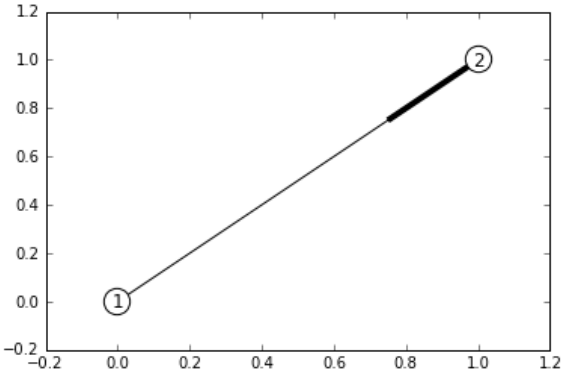

Figure 17 - Simple graph created in NetworkX

NetworkX has several built in functions for the graphs that simplify the programming significantly. For example, the are methods for determining the degree of pipes entering and leaving a node, the neighbors of a node, the edges connected to a node and several more. An edge is simply the line between two nodes, which in our case is referring to the twin pipes (supply and return pipes).

The network representation in this thesis consists of producers, customers, intersections and pipes. The intersections are so called "dummy nodes" which only serves to distribute mass flows to several pipes connecting to it. An example of an imaginary district-heating network can be illustrated in Figure 18.

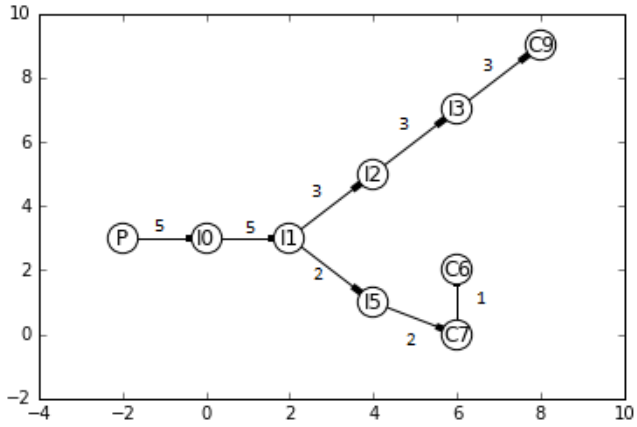

Figure 18 - Complex DHN network

The intersection nodes I0, I1, I2, I3 and I5 do not consume any load whereas the nodes C6, C7 and C9 are customers and thus consumes. The mass flow leaving the node P (the producer) can for example be 5 kg/s. The flow between I0 and I1 is also 5 kg/s, since they do not consume any load. However, in the node I1 there is a branch, which means that the mass flow is divided in the different branches. For example, 2 kg/s can enter node I5 and thus 3 kg/s node I2. The deeper the branching continues in the network the lower the mass flow will be.

4.3 Translate representation to Dymola

As described in the previous section, the network is represented in Python using the package NetworkX. The information of each pipe consists of pure geometry values such as diameters and lengths. To obtain the mass flows and supply temperatures distributed along the system required for a the aggregation around a given operating point, the network has to be simulated. This is done in the software program Dymola or in Jmodelica.org (OCT). Therefore, the network designed in Python has to be "translated" into Modelica code, which is the language used in Dymola. The code for translating the networks was written in Python and generates a Modelica file. The graphical layer in Dymola is not necessary for the optimization procedure, however it enables the user to visually see the designed system to be simulated from Python. Figure 19 and Figure 20 shows the network in Python and its corresponding translated version.

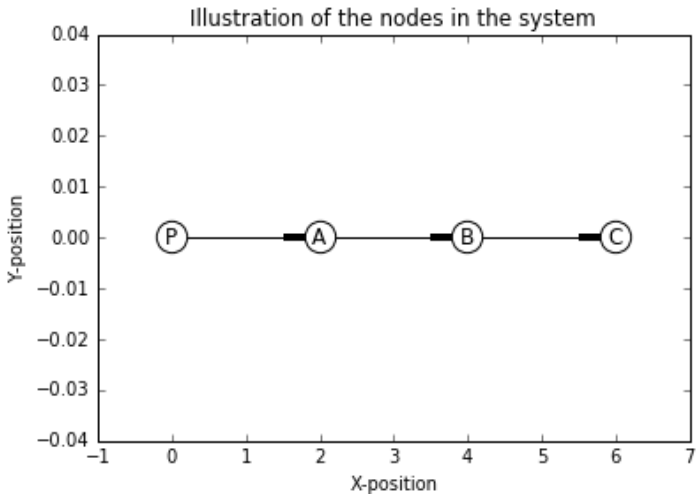

Figure 19 - Simple DHN to be translated to Dymola. NOTE: Each line represents two pipes (supply and return).

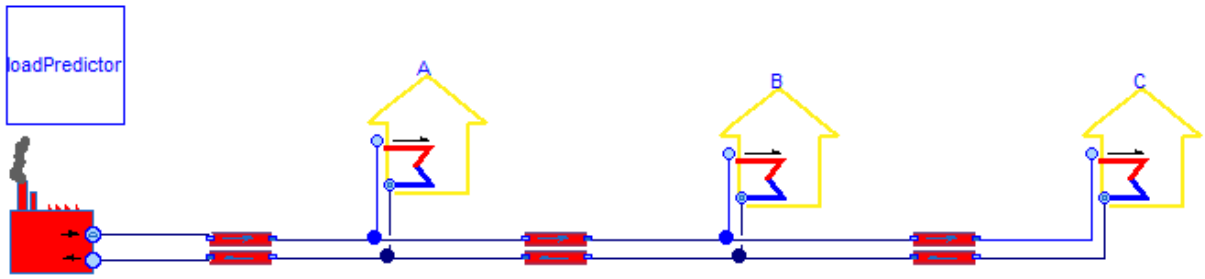

Figure 20 – Translated DHN from Python. NOTE: The customers are connected in parallel, not in serial.

Each line between the nodes is represented as a "double pipe" (one supply and one return), connecting customers and the producer P. Once the translation is done, the model Figure 20 is simulated to obtain nominal values for the different flows in the system.

The loadPredictor component allows the customers to have different kind of load profiles. In this thesis however they are defined to have the doublePeak as shown in Figure 16. Another advantage of this component is that it makes it possible to avoid the following:

- The load connections between the producer and the customers
- The return temperature between the producer and customers
- The outdoor temperature between the producer and the pipes

This component was created and placed in the main graphical window, which contained these 3 parameters (load for each customer, return temperature and outdoor temperature for the pipes). These values were instead “called” from each component to obtain their corresponding values.

However, the reason for this setup is not only for graphical purposes. This method is also required since the customer load has to be manipulated later on, in order to conserve the power load after aggregation. Section “5.3.1 Aggregation and preserving the load” describes this in further details.

Another problem when translating from Python to Dymola is that it is only possible to translate one Python type into one Dymola type/component. Since two different pipe types will be used in Dymola (supply pipe and return pipe), it is not possible to translate the python pipe type into both of these Dymola pipes. It is only possible to translate the python “pipe” into one of them. In order to solve this problem, a new component had to be created, with both a supply and a return pipe, as can be seen in Figure 20. This component was then used when translating the “edge” component from Python. The two versions of this component were created – one for simulation and one for optimization.

5) Process in detail

In this section, the process in section 2.5 Overview of process will be explained in a detailed manner.

5.1 Complex DHN

The first step of the project was to set up a complex district-heating network. Complex in this thesis refers to the real network and that the data of the pipes only include geometries such as diameter, insulation thickness and length. The customers should be defined to have a minimum and maximum supply temperature being delivered to them. Lastly, the producer is defined to only specify the required amount of heat power to be generated, which is the sum of the customer load plus distribution heat losses.

The NetworkX Python package was used to set up the network. Since we have not used the Python language before, the basics of it were initially learned. Basic commands to go through lists, simple dictionary operations and calculations were examined, mainly through the webpage of NetworkX (NetworkX, 2016).

The overall structure of how this was going to be done was based on a previous master thesis work (Larsson H. , 2015). A quite simple structure for the German method in Python was built, however, there were several assumptions in this work. For example, the networks used in the models were not pressure based and therefore the mass flows distributed were predetermined according to a split, which you set earlier. In reality, the pressure decides how the mass flow is distributed.

Another flaw of the implementation was that the system could not handle a network representation containing “loops”, which is quite common for real district heating systems. Also, in our implementation, there is an “exclude function” for key customers not to be aggregated. If for example a branch of two customers contains vital data for the overall dynamic behavior, the user can exclude these customers from being aggregated. This function can be quite useful in reality when there are thousands of customers in the network.

The network representation of the complex system is designed according to section 4.2 Python using NetworkX. This “complex” network is then built upon to resemble a real network purely based on geometries. A typical complex system is illustrated in Figure 21, with several producers, branches, loops and serial connections:

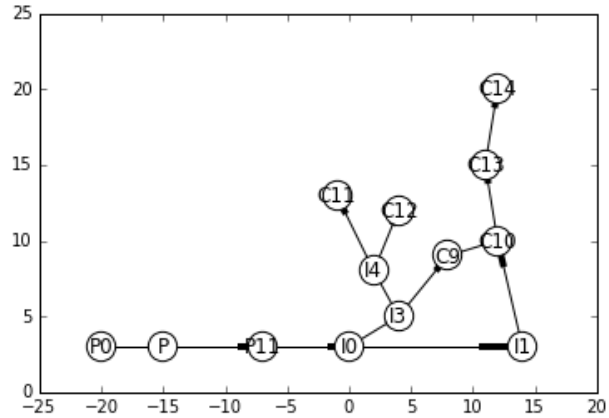

Figure 21 - Typical complex DHN

When a customer is added to the network, an index is created as a signature for each customer. This is used to keep track of all the customers in the network. Another important key factor in setting up the complex network is to define the customer loads as a matrix, according to (52):

$$loadPredictor = A(n,load) \quad (52)$$

The size of this matrix depends on how many customers the complex network is represented by, which is the value *n,* whereas the load consists of a typical load distribution (the doublePeak is used in this thesis, as shown in Figure 16). This matrix will then be used to compensate for load distributions after aggregation is performed. Section "5.3 Simplified DHN" will describe this in further details.

This way of setting up the customer load makes it possible for the user to decide how the load is to be defined for each customer and to make it possible for the criteria of load preservation (37) to be fulfilled, according to fractions in the aggregation part.

5.2 Nominal values from Simulation of Complex DHN

This simulation of the complex DHN is performed after the complex network is designed in Python. When the complex network program is executed, it generates an MO-file (Dymola file), which is compiled and simulated by Dymola.

The design of the network has made it possible for Dymola to only require geometries from the system along with nominal values on all the loads, and based on this information it simulates the dynamic behavior of it. The simulation gives important information such as what mass flows and supply temperature each customer receives, the pressure in different nodes of the network and how much the losses are from the pipes. This simulation yields a new type of complex network, however with all this additional simulated information.

The new district-heating network is still "complex", since it still contains several customers, however the new system contains the dynamic behavior of mass flows and temperatures distributed along the network.

5.3 Simplified DHN

The reason for updating the original complex network with dynamic behavior is because that the aggregation methods used requires constant values on loads, supply and return temperatures and mass flows for calculations. As described in previous sections, The German and the Danish method differ regarding what dynamics are preserved and what kind of structures they are able to handle. The details of these methods are explained in section 3) Methods.

5.3.1 Aggregation and preserving the load

When the complex network is being aggregated, the total load in the original network has to be preserved in its corresponding simplified network, according to (37). The solution to this was done by a number of steps. The signature of each customer (indices) makes it possible to keep track of which customer the aggregation is currently working on.

Imagine a network consisting of 4 customers, where each customer contains the fraction 1 in the beginning (see Figure 22).

5) Process in detail

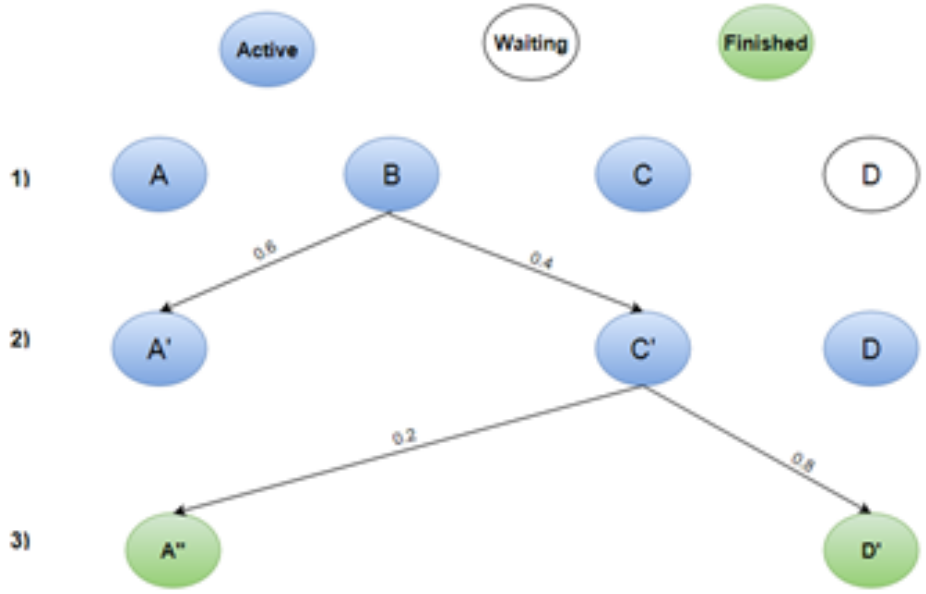

Figure 22 - illustration of aggregation preservation

In the first step, the 3 customers A, B and C are active for aggregation. The customer to be removed is the one in the middle, i.e. customer B. The distribution of loads is based on (30) and (31) and will determine how much of the load that flows to customer A and B. In this example, arbitrary values of 0.6 and 0.4 are distributed. The sum of the fractions has to be one in each step.

These fractions are stored in a type of list called dictionary in Python. For example, in the first step, the dictionary will have the following content:

*Dict1 = {1, taken from customer **A,** 0.6: taken from customer **B**}*
*Dict2 = {1, taken from customer **D,** 0.4: taken from customer **B**}*

The fraction 1 in both dictionaries is taken from customer A and C, since these are not aggregated. In the second step, A', C' and D are active. The customer in the middle to be removed is customer C'. Once again the fractions are chosen arbitrary and stored in the two dictionaries.

In the third step, the aggregation is finished, since the method cannot handle two customers. The dictionaries will now have the following contents:

*Dict1 = {1, taken from customer **A,** 0.6: taken from customer **B,** 0.2: taken from customer **C'**}*
*Dict2 = {1, taken from customer **D,** 0.4: taken from customer **B,** 0.8: taken from customer **C'**}*

Once again, the sum of each column of fractions is always one. Also, the total sum of all fractions is equal to four, which is the number of customers in the original network.

The resulting dictionaries will then be translated into the optimization models and used to compensate for load in the complex network. Without this procedure, the total load in the complex network would otherwise be equal to two, which would break the preservation of load from (37) and (52).

Another important thing when doing aggregation is that the valve openings are updated according to the new mass flow. When 3 customers are aggregated and the middle one is removed, the sum of the mass flow of the previous 3 customers is preserved in the 2 remaining customers. Therefore, the new valve opening has to be increased as a compensation of the increased mass flow.

5.4 Simulation and optimization of simplified DHN

After the aggregation is performed, the original, complex network has been reduced to contain only a few customers and still preserve its main dynamics. The reason for simulating the lumped network again is because that the optimization requires initial guesses, which the simulation provides. Based on this information, the optimization is performed, which gives optimal trajectories for different flows and pressure and still fulfill the customer load.

5.4.1 Optimization with JModelica.org

The main goal regarding optimization is to minimize a given objective function. That function can for example be heatUnit.T_out, which is the supply temperature.

Figure 23 illustrates how the optimization is performed:

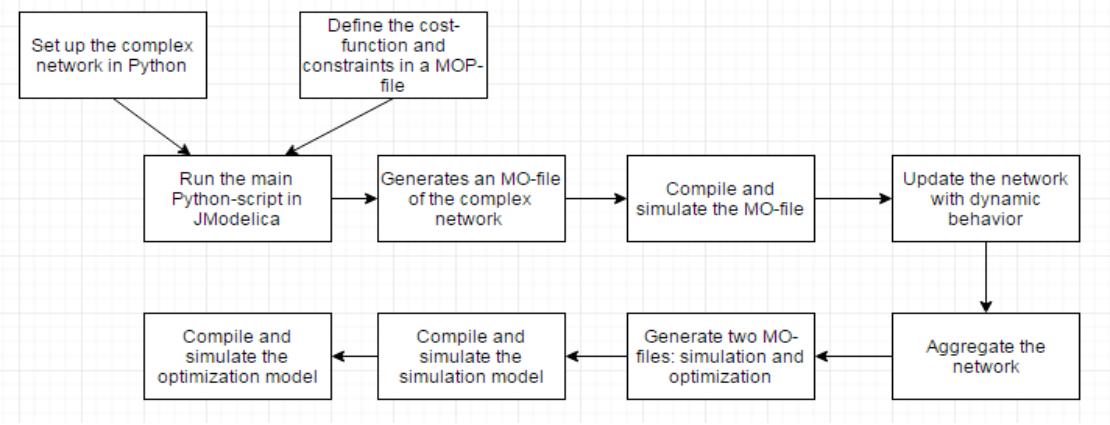

Figure 23 - Optimization with JModelica.org

The first step is to set up the complex network in Python and define the cost function and corresponding constraints in a simple text file. When the main script is executed, all the coming steps in Figure 23 are performed automatically. A simple example of the MOP-file is illustrated in Figure 24.

```
optimization caseSimple_opt_mod4(startTime = 0, finalTime = 24*3600, objectiveIntegrand =1/10000000*(heatUnit1.T_out))
extends OptiJasirCodes.OptCases.SimpleOpt4;
constraint
customer1.T_in >= 70+273;
end caseSimple_opt_mod4;
```

Figure 24 - Simple MOP-file example

The cost function in this case is the supply temperature of the heatUnit (producer). This means that the solver tries to minimize the output of it and still fulfill the customer load. In the constraint section, there is an additional requirement that the supply temperature of the first customer should not be lower than 70 degrees Celsius.

The simulation model is loaded in the main Python script. This means that all the variables and trajectories are stored in Python. First we simulate the model from a negative time to zero, and then from zero to a positive time. The reason for this is to make sure that the system is in a steady-state mode, which sets proper initial values for the system. This gives a more accurate simulation for the “real case”, when the model is simulated from zero to a positive time. Therefor this was included in our modelling.

In the last part, the simulation model gives proper initial guesses for the solver in the optimization case. In this part, several parameters can be chosen such as the number of finite elements, initial trajectories, different linear solvers etc. Based on the trajectories from the simulation model as initial guesses, the solver will try to find an optimal solution.

5.5 Optimal trajectories for customer load

In the end there is a result text file with all the data for the different variables in the system. The goal is to minimize the supply temperature and still fulfill the customer load. The goal is to use the results as inputs to the original, complex network. If the mass flows and temperatures add up to reasonable values for the customer demand, the loop can be seen as successful.

6) Result

In this section, illustrations and answers to the three problems from section 1.1 Problems to examine will be presented. Initially, the concept of a pressure-based system will be demonstrated and after that the full framework will be shown on a real district-heating network in Austria with 16 customers.

6.1 Pressure based system

In previous models regarding district-heating networks there were not any pressure included in the system, instead there was a component, a splitter, which was used to simply divide the mass flow between different nodes according to an arbitrary fraction. In reality, it is the pressure difference that decides this fraction.

In this section the optimization including pressure will be demonstrated (cost-function is set to minimize temperature at customer D). In the first case there is no constraint on the pressure difference between any nodes. As can be seen from Figure 25, the temperature reaches its minimum value, while the mass flow and pressure changes.

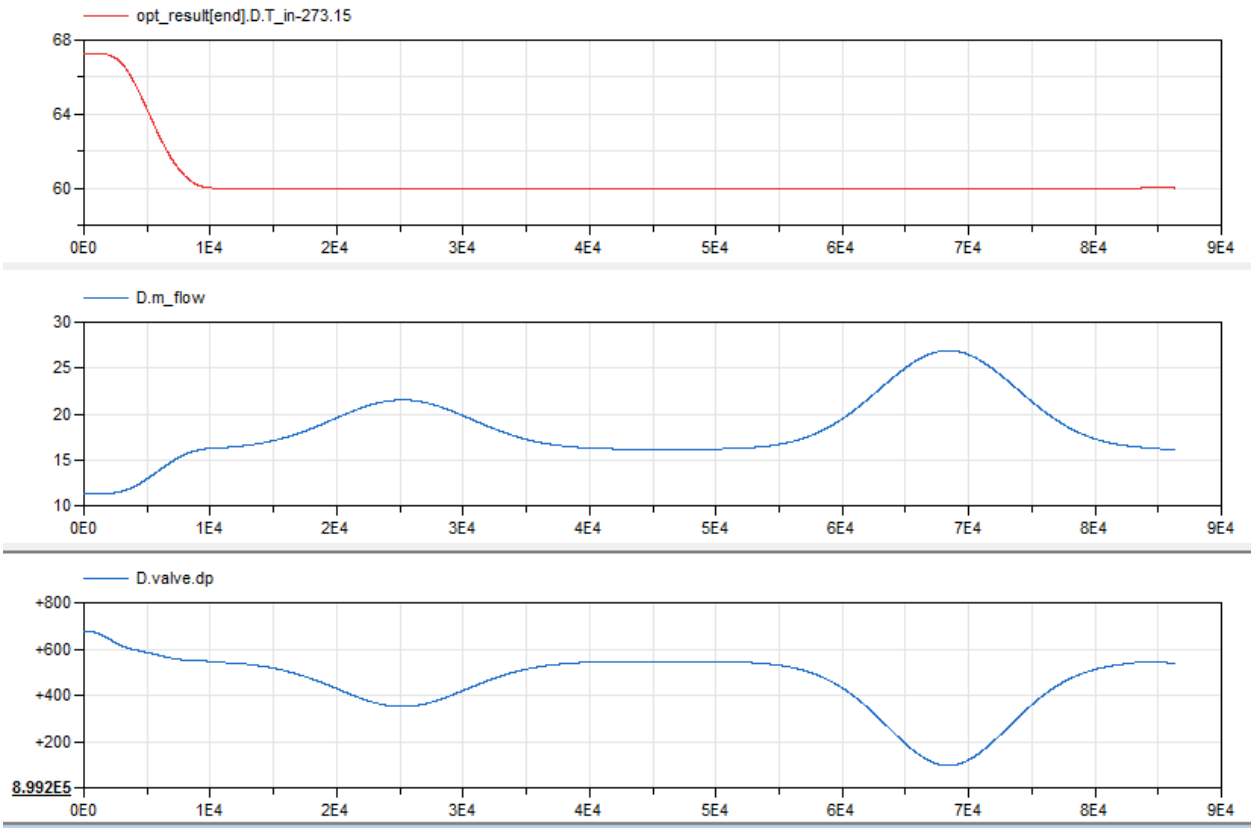

Figure 25 - Optimization without a constraint on pressure. Top: supply temperature at customer D. Middle: mass flow at customer D. Bottom: differential pressure across customer D.

In Figure 26 however, there is a constraint on the pressure difference. In this case we set it to be a minimum of 960 kPa. As we can see, the lowest limit of the pressure difference “D.valve.dp” is exactly this value. This affects the mass flow as well, since the pressure difference sets the mass flow. Therefore, the only thing that can change is the supply temperature for the customer.

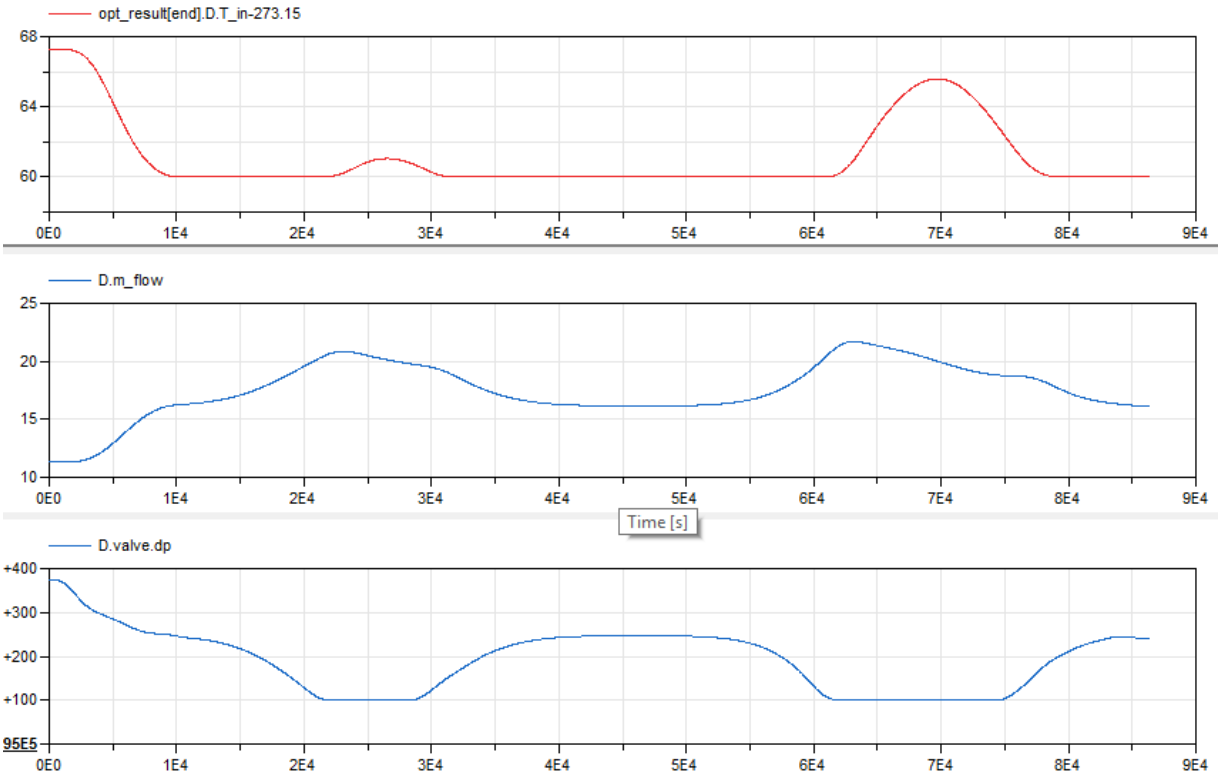

Figure 26 - Optimization including a constraint on pressure

This answers question 1 in section 1.1 Problems to examine – introducing pressure in the framework does indeed give reasonable results.

6.2 Test cases for Graz

In this section, the full framework will be demonstrated for a real case. The network will be a small area in the Austrian city Graz with 16 customers and 4 intersections. To illustrate the effect of errors of the optimization, the network will be aggregated to two and three. Since the network itself is relatively small (about 1km between producer and the customer furthest away), the lengths of the pipes will be increased to illustrate the effect of delays. The real network is illustrated in Figure 27 and its corresponding representation in Figure 28.

Figure 27 - District heating network for a small area in Graz. Picture taken from (TU-Graz, 2015).

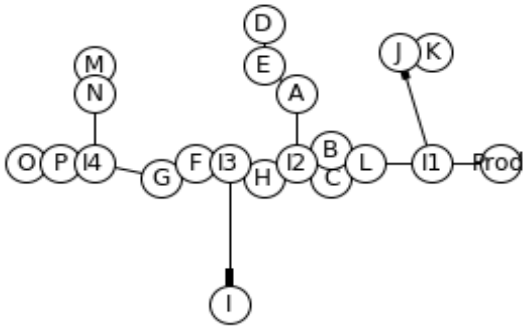

Figure 28 - Corresponding network in Python

There are 16 customers (A-P) connected to the district heating network and the longest distance from the producer to customer (Prod to customer D) is about 1 km, which gives an idea of the size of the network is. There are also intersections included, which are referred to as I1, I2, I3 and I4. However, these are defined to have a load of 0 but still be defined as customers, as described in section 4.2 Python using NetworkX.

6.2.1 Aggregate to 2 customers

In this section, the network will be simplified down to 2 customers. This complex network is firstly simplified using the German method (see 3.1 The German method). Figure 29 illustrates the corresponding simplified network:

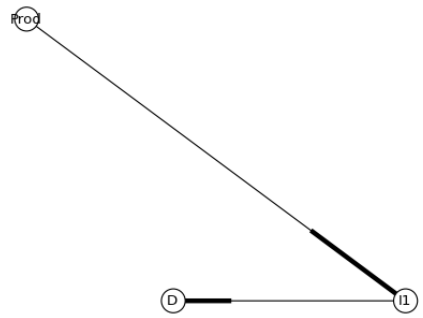

Figure 29 - Lumped network for Graz

As can be seen from Figure 29, the remaining customers left are I1 and D and the reason for this is because I1 is the closest node to the producer and customer D the furthest away, lengthwise. Therefore, these two nodes cannot be removed. The minimum number of customers left after aggregation for the German method is 2. From section 5.3.1 Aggregation and preserving the load, there were illustrations of how the lumping procedure is performed, which further explains this.

Table 5 illustrates the fractions for the remaining two customers after aggregation, as explained in section 5.3.1 Aggregation and preserving the load:

Table 5 - Fractions obtained after aggregation for load preservation

Indices used for lumping	Customer I1	Customer D	Indices used for lumping (continued)	Customer I1	Customer D
1	0.64	0.36	11	0.71	0.29
2	0.77	0.23	12	0.44	0.56
3	0.86	0.14	13	0.93	0.07
4	0.52	0.48	14	0.36	0.64
5	0.00	1.0	15	0.54	0.46
6	0.63	0.37	16	0.52	0.48
7	0.68	0.32	17	1.0	0.00
8	0.28	0.72	18	0.59	0.41
9	0.77	0.23	19	0.81	0.19
10	0.61	0.39	20	0.72	0.28

6) Result

Table 5 shows the fractions obtained after the aggregation. As can be seen, the sum of each fraction row is 1. From index 5, customer I1 gets 0.0 while D gets 1.0 of the load. This is because index 5 is referred to customer D itself, which is not aggregated, which means that it receives its whole load. The same principle applies for index 17.

The size of each fraction for the corresponding customer depends on how far away the middle customer is from the two customers surrounding it. For example, in index 19, we can see that 81% of the load is distributed to customer I1 while only 19% to customer D, which suggests that customer I1 is much closer to the middle customer than customer D. The reason that the customers remaining are I1 and D is because they are the closest and furthest away from the producer.

As a first result, the optimal trajectories for the small area in Graz will be presented. Results are based on minimizing the supply temperature of the producer, to fulfill the minimum supply temperature required by the customers. The customers in Graz require 60°C as a minimum, and the optimal trajectories will therefore be based on this. Figure 30 – Figure 32 illustrates the optimal trajectories in this regard.

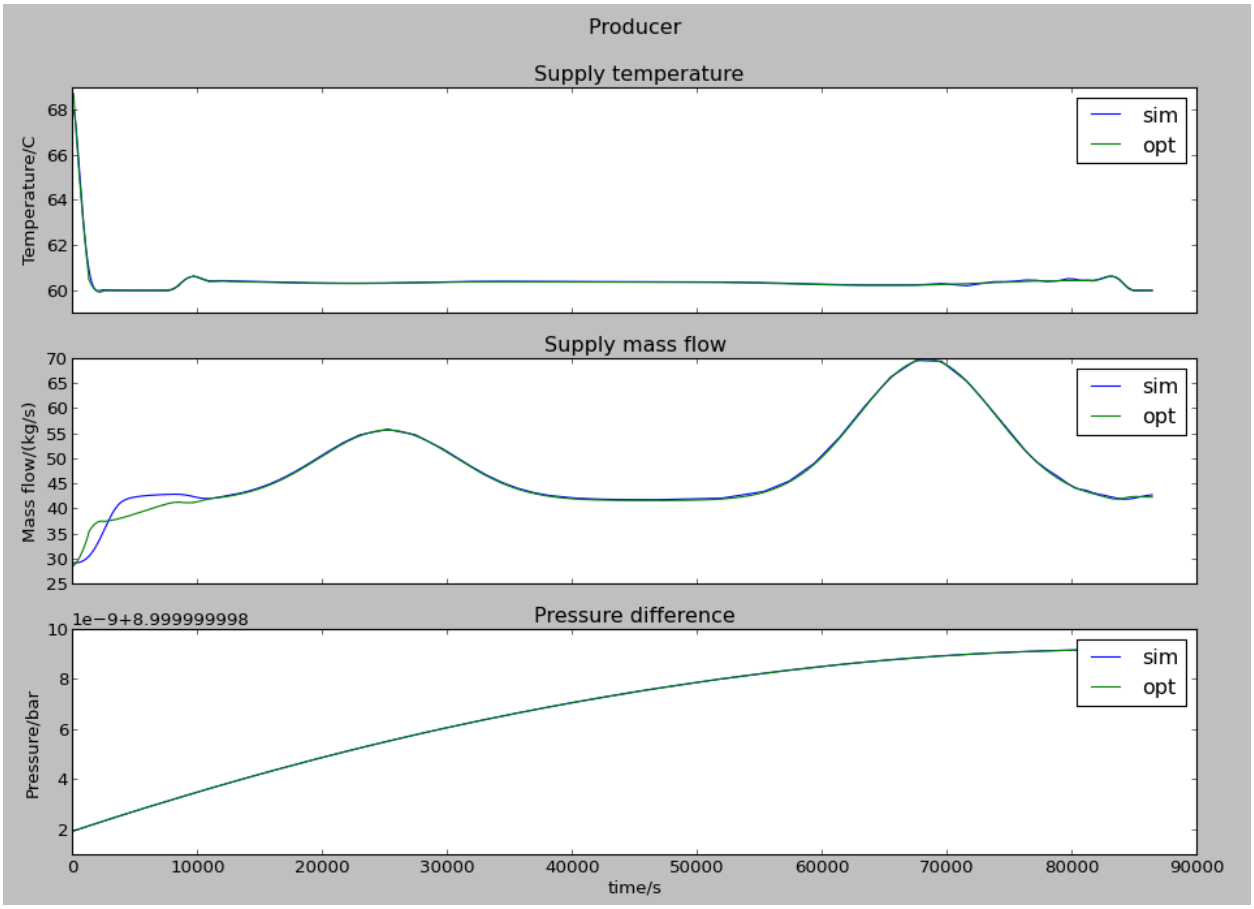

Figure 30 – Optimal producer trajectories for Graz. NOTE: Pressure difference is constant.

The simulation results (sim) are based on the optimal trajectories from the simplified network which are then applied to the original, complex network, whereas the optimization results (opt) are the optimal trajectories directly in the lumped network. Figure 2 illustrates this.

As can be seen from Figure 30 the producer supplies a slightly higher temperature than 60$C°$, even though the minimum temperature required by the customers in Graz is 60°C. The reason for this is due to heat losses in the pipes. Therefore, the solver has to compensate for this and to guarantee that the limit is fulfilled; it sets a slightly higher value. Another thing to notice is that the only thing that varies is the mass flow, which varies according to the customer load demand (Figure 16). This is because the cost function tries to minimize the supply temperature.

In (52), the load predictor equals $\dot{Q}$ and the term $C_w\ (T_{supply}\ - T_{return})$ is constant, thus the mass flow is the only thing that can vary. Also, the pressure is set to a constant value in the whole network (10 and 1 bar at the supply and return nodes of the producer). The trajectories for the remaining customers (D and I1) are illustrated in Figure 31 and Figure 32.

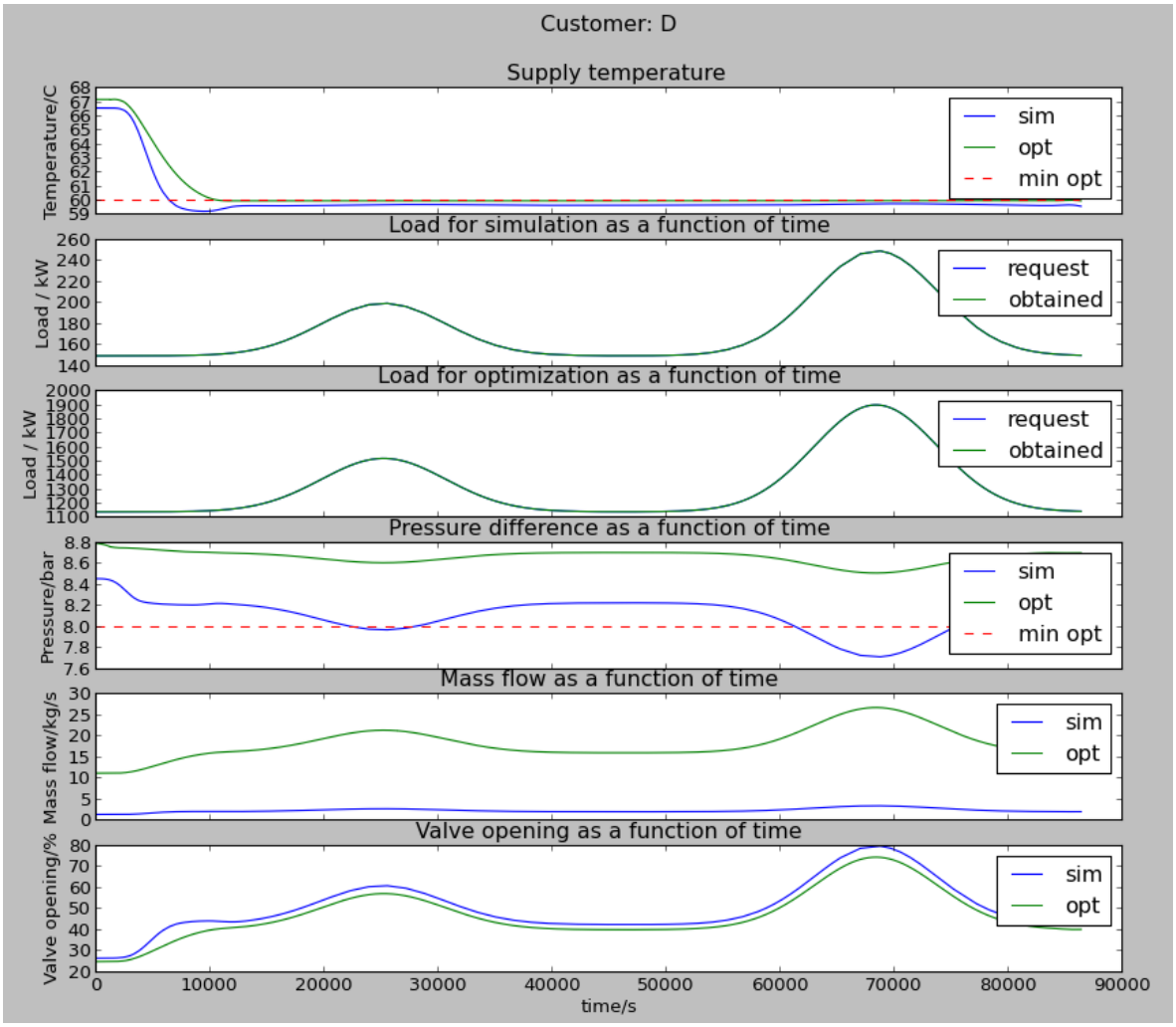

Figure 31 - Optimal trajectories for customer D in Graz. Green: optimized trajectories with aggregated DHN. Blue: trajectories in complex DHN with optimized inputs.

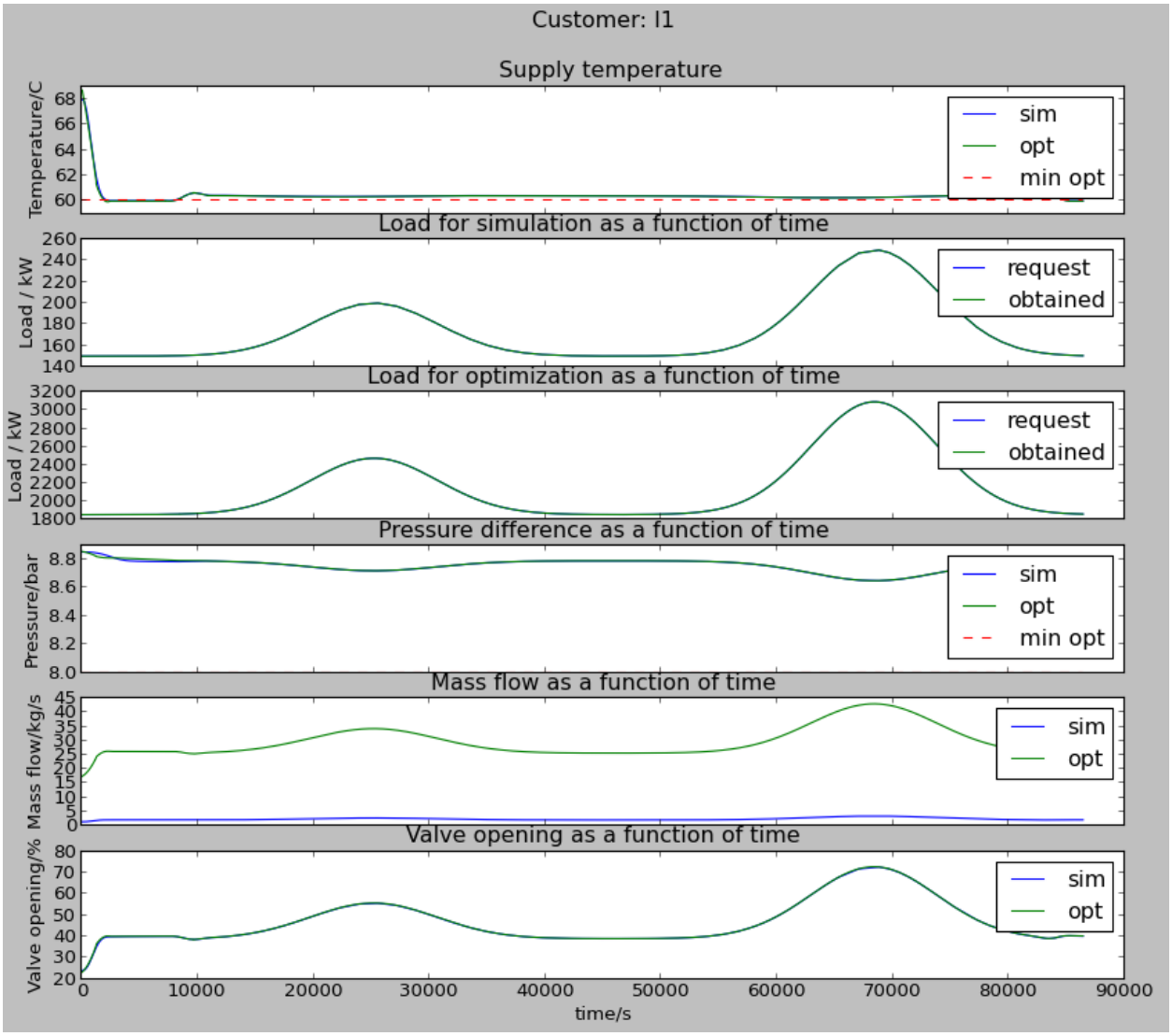

Figure 32 - Optimal trajectories for customer I1 in Graz. NOTE: The "min opt" in pressure difference equals 8.0.

Once again, one have to differ between the "sim" and "opt" results. The sim in this case is based on the large, complex network and customer D has therefore only its original load. The "opt" results are based on the lumped network and customer D has therefore a much higher load (since the total load in the network has to be conserved). This is why the loads and the mass flows in the "sim" results are much lower than the ones in the "opt".

From Figure 31 we can see that the temperature that reaches customer D is about 60 $C°$, which is the minimum it requires. The load requested and the load obtained are identical in this case, which means that the customer receives exactly the amount of load demanded. Another thing to observe since the valve openings are updated is that the load in the optimization is much higher than simulation.

The valve opening decides how much of the mass flow is allowed to flow. As we can see in Figure 31 and Figure 32, as the valve opens, the mass flow increases and vice versa.

The equation for the heat flow in the network is calculated according to:

$$\dot{Q} = \dot{m}\, C_w\, (T_{supply} - T_{return}) \quad (53)$$

A typical value of the specific heat (C_w) for water equal to 4190 $^{J}/_{kg \cdot K}$. In this case, the return temperature of the customers is set to 43°C and supply temperature to 60°C.

From Figure 31 and Figure 32 the sum of the mass flows at half the time horizon (because the values around this time is constant and therefore it is easier to read off the values) is 15.5 + 26.5 kg/s = 42 kg/s and by using (53), the total load is calculated according to:

$$\dot{Q} = 42 * 4190\ (60 - 43) = 2.99\ MW$$

This total load is the sum of each individual customer load demand, which means that the load is indeed preserved in the system, which answers the second question from 1.1 Problems to examine.

6.2.2 Aggregate to 3 customers

In this case, the same procedure will be performed as in the previous case including a pressure constraint. However, the system will be aggregated to 3 customers instead of 2. However, to avoid having too many figures, the profiles for the closest and furthest customer away from the producer, which is I1 and D, will be presented. Figure 33 illustrates the network after lumping:

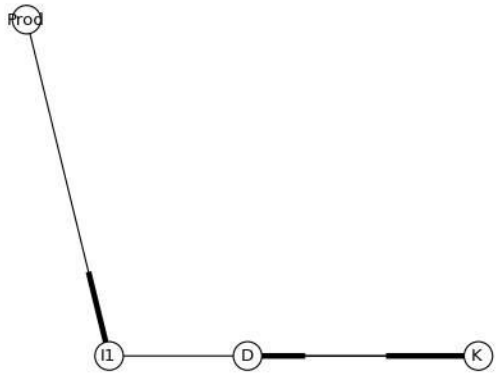

Figure 33 - Aggregated network for three remaining customers

The producer has a pipe connected to customer I1, customer I1 has a pipe connected to customer K and lastly, customer K is connected to customer D.

6) Result

Table 6 illustrates the fractions for the remaining three customers after aggregation, as explained in section 5.3.1 Aggregation and preserving the load.

6) Result

Table 6 - Fractions obtained after aggregation for load preservation

Indices used for lumping	Customer I1	Customer D	Customer K
1	0.64	0.36	0.00
2	0.64	0.36	0.00
3	0.87	0.13	0.00
4	0.52	0.48	0.00
5	0.00	1.0	0.00
6	0.63	0.37	0.00
7	0.68	0.32	0.00
8	0.28	0.72	0.00
9	0.77	0.23	0.00
10	0.00	0.0	1.00
11	0.16	0.0	0.84
12	0.44	0.56	0.00
13	0.93	0.07	0.00
14	0.36	0.64	0.00
15	0.54	0.46	0.00
16	0.52	0.48	0.00
17	1.0	0.0	0.00
18	0.59	0.41	0.00
19	0.81	0.19	0.00
20	0.72	0.28	0.00

Customer K has many zeros in its load fractions and to explain the reason for this we refer to Figure 28. The location of customer K is in the beginning of the network (as a first branch next to the producer). During the aggregation procedure, the method looks for any serial structure (3 customers in a row), which in this case is (I1, J and K). Customer J is in the middle and therefore removed, leaving customer K as a leaf. Thereafter the method leaves this branch (since the branching can only be performed at the end of the tree). In the end, the branch with customer K is the only part left, which is why customer K has not been receiving loads during most of the aggregation.

The corresponding optimal trajectories for this case is illustrated in Figure 34 - Figure 36.

6) Result

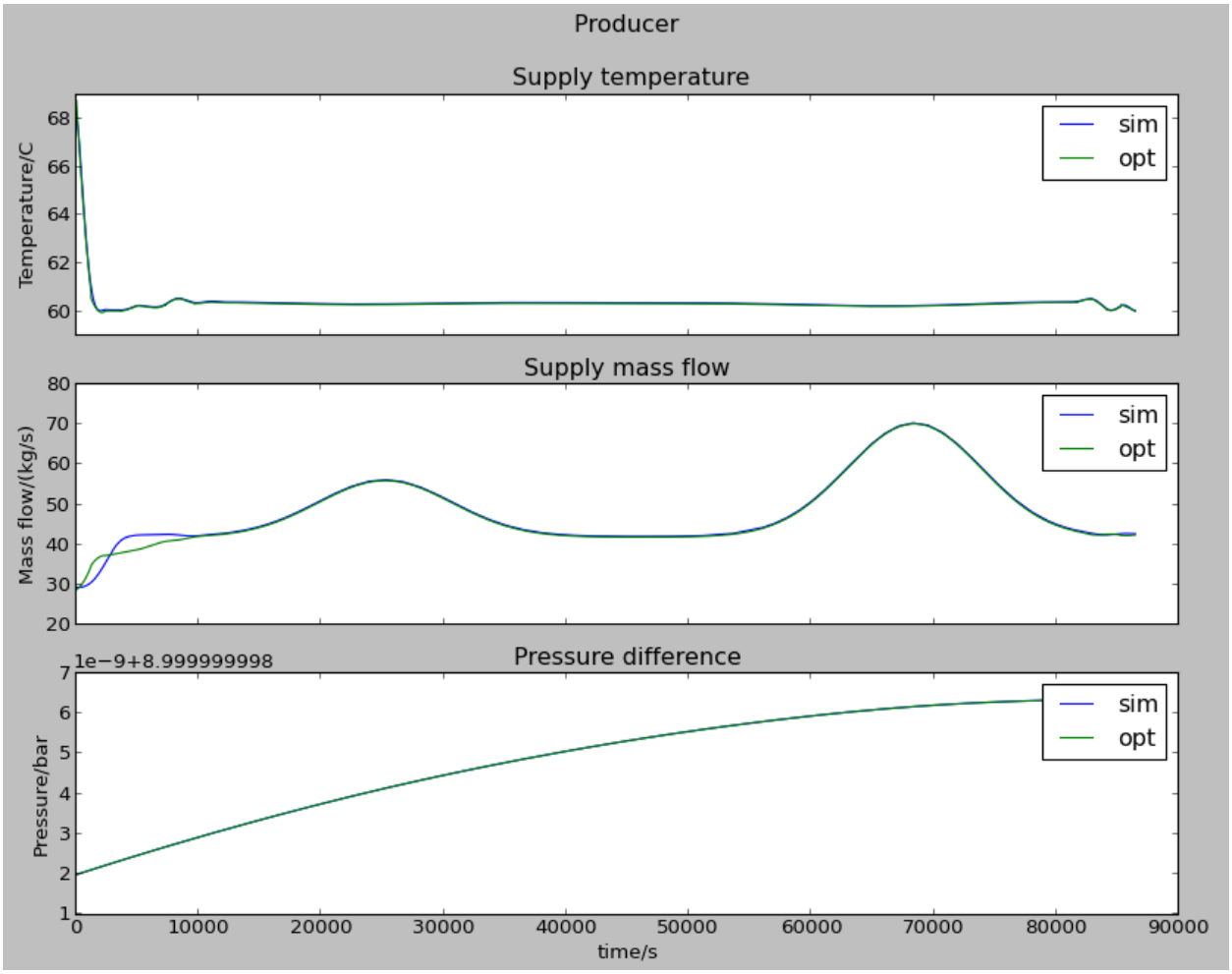

Figure 34 - Optimal trajectories for the producer. NOTE: Pressure difference is constant.

The trajectories for the producer in this case (Figure 34) and in the first case (Figure 30), we see that the profiles for the mass flow are identical. This is because the producer unit supplies for the same amount of total load after aggregation.

6) Result

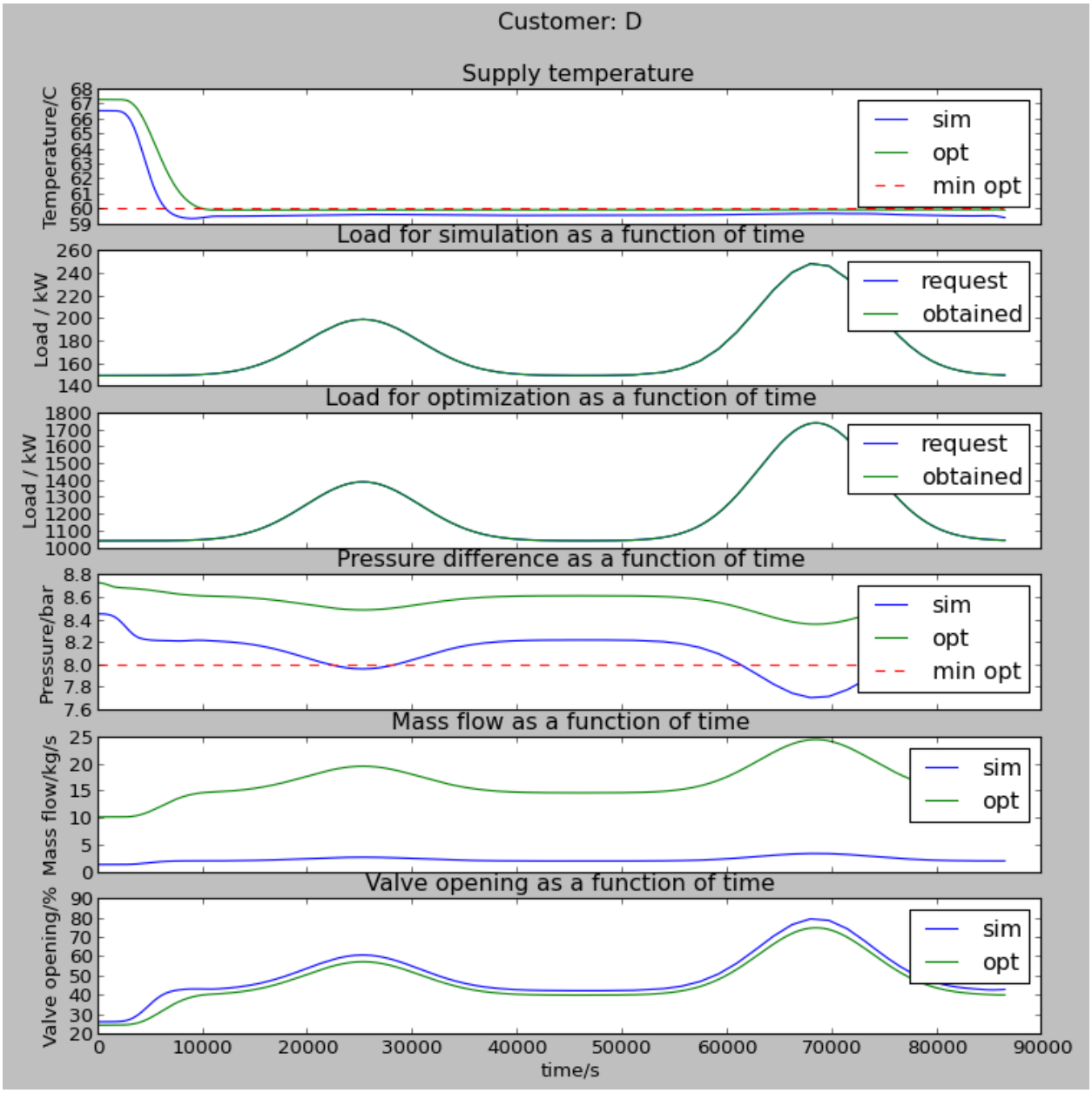

Figure 35 - Optimal trajectories for customer D

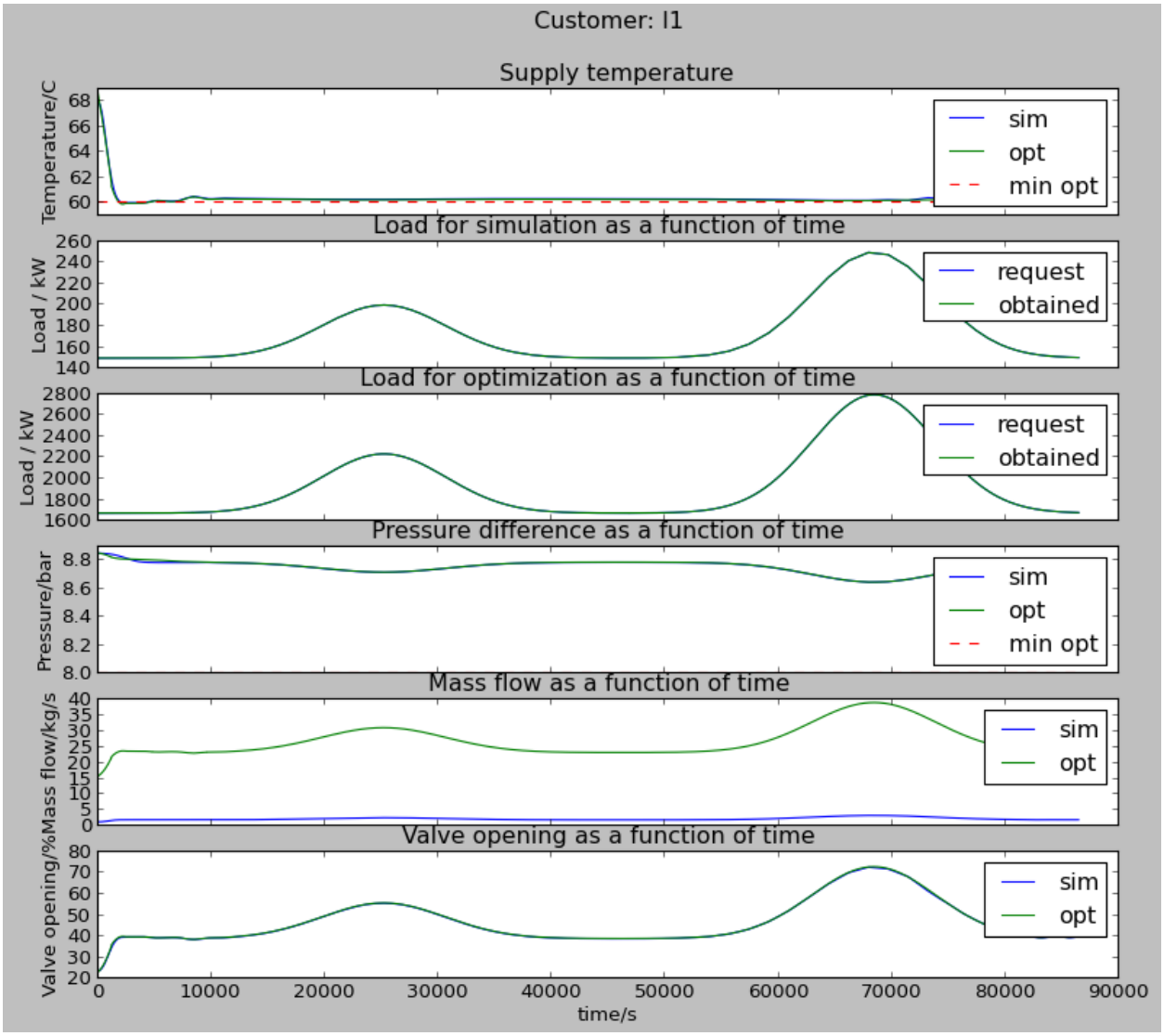

Figure 36 - Optimal trajectories for customer I1. NOTE: The "min opt" in pressure difference equals 8.0.

If we look at the pressure difference for customer I1 (Figure 36) and D (Figure 35), we can see that it decreases from about 8.84 bar to 8.72 bar (at time = 0). This is due to the pressure drop in the network as you go further away from the producer.

By studying the cases in sections 6.2.1 Aggregate to 2 customers and 6.2.2 Aggregate to 3 customers, the second question of the thesis can be answered once again - The customer load is fulfilled after aggregation based on the developed framework, both when the network has 2 and 3 remaining customers.

Regarding question 3 in section 1.1 Problems to examine, the framework is indeed flexible to change components if some criterions are fulfilled. This will be further discussed in the next section.

6.2.3 Increasing lengths (2 customers left)

In this section, the lengths of the pipes in the original network will be increased with a factor 7 to illustrate the effect of delays. Figure 37- Figure 39 illustrates this.

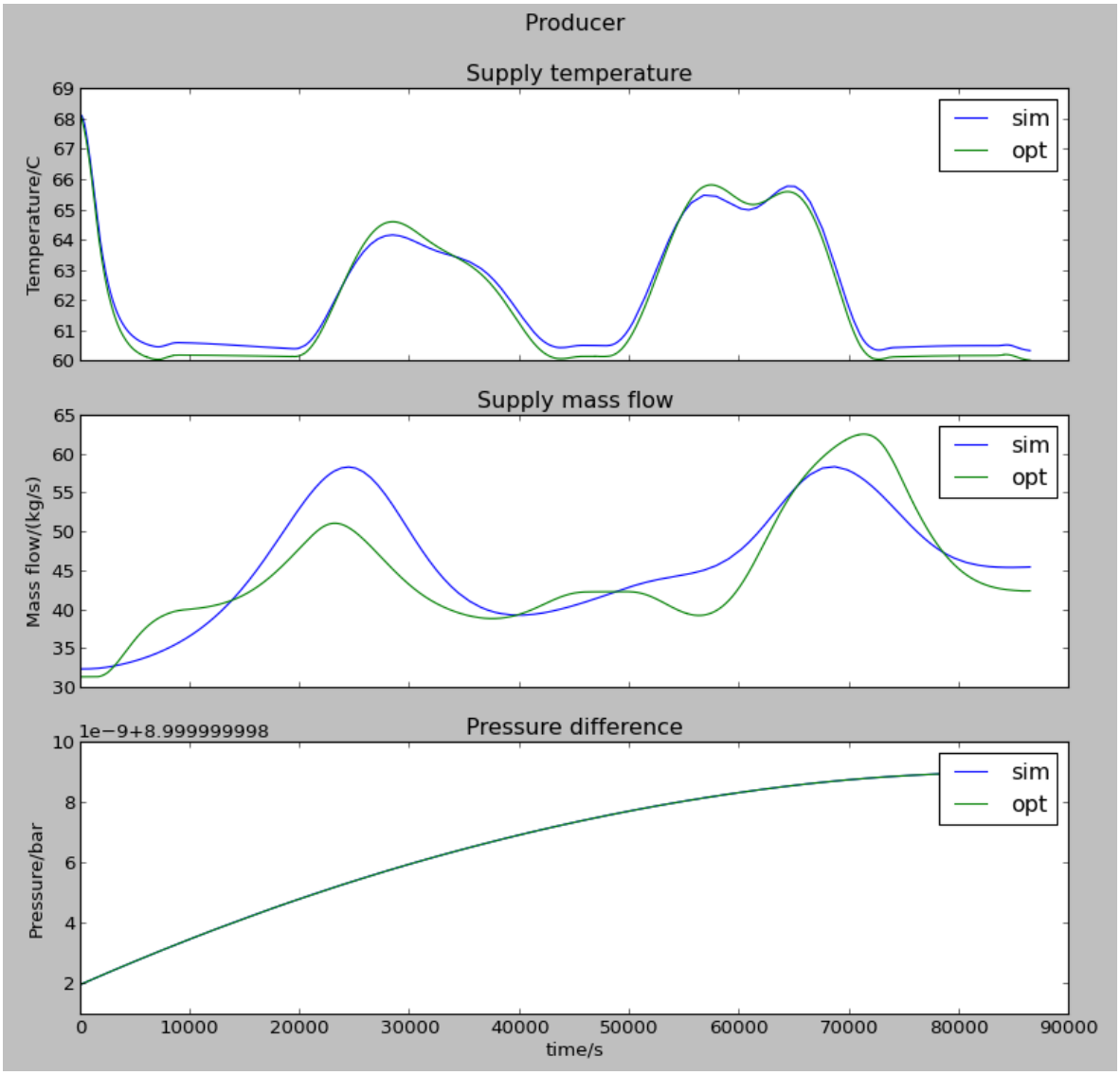

Figure 37 - Optimal trajectories for the producer. NOTE: Pressure difference is constant.

The trajectories for the producer in this section (Figure 37) differ. The supply temperature oscillates and the reason for this is due to the impact of delays. The customers both require a doublePeak profile for their loads (Figure 16) and therefore the producer has to take this into account by supplying two peaks. This happens two times at t=29000s and t=60000s.

6) Result

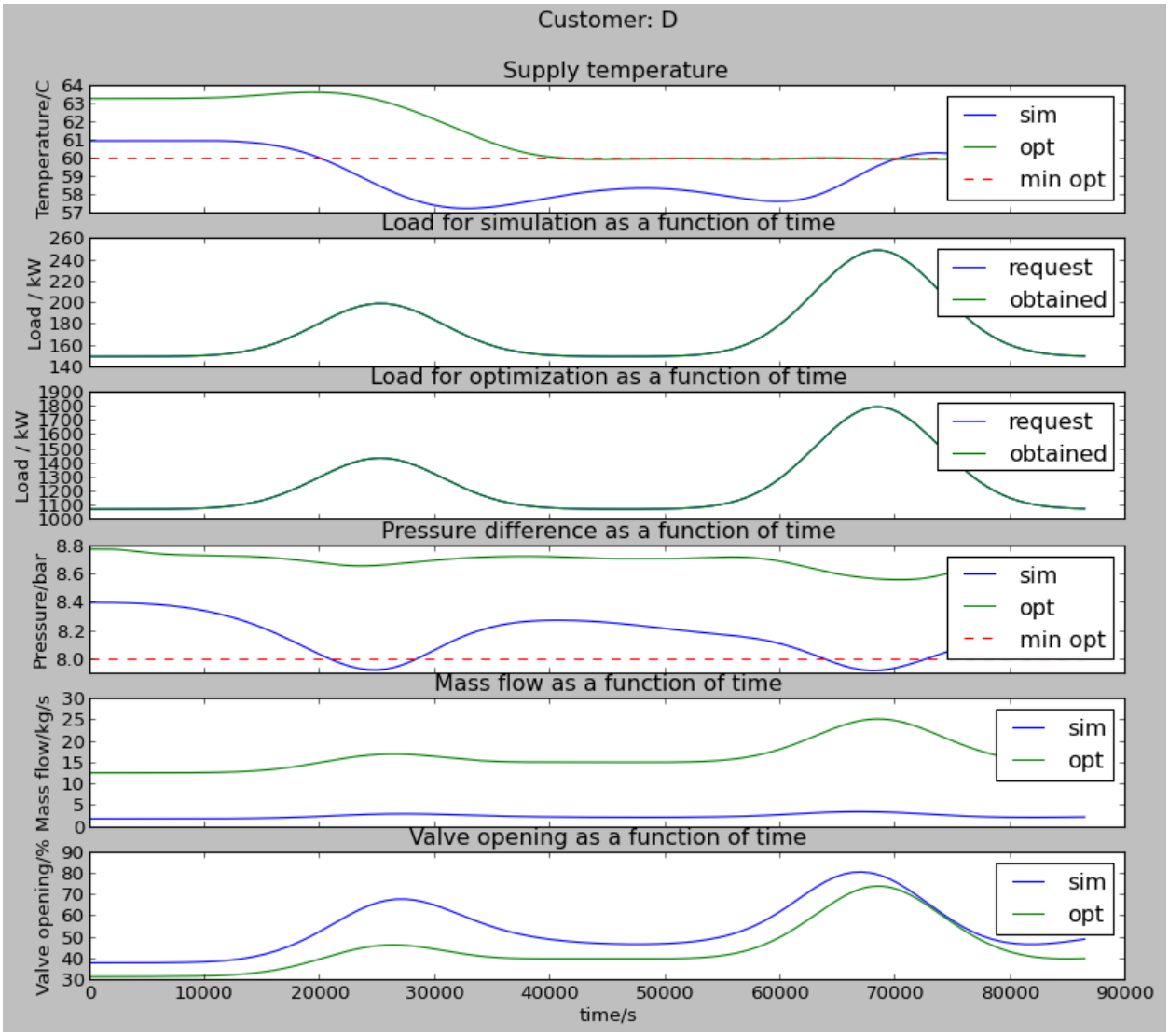

Figure 38 - Optimal trajectories for customer D

The mass flow and temperature at the producer differs from the case without delay:

a) The supply temperature for the producer is not constant which makes sure that the customer operates at its minimum supply temperature.

b) Peaks in the supply temperature for the producer are because it takes into account for transport time to customer D.

6) Result

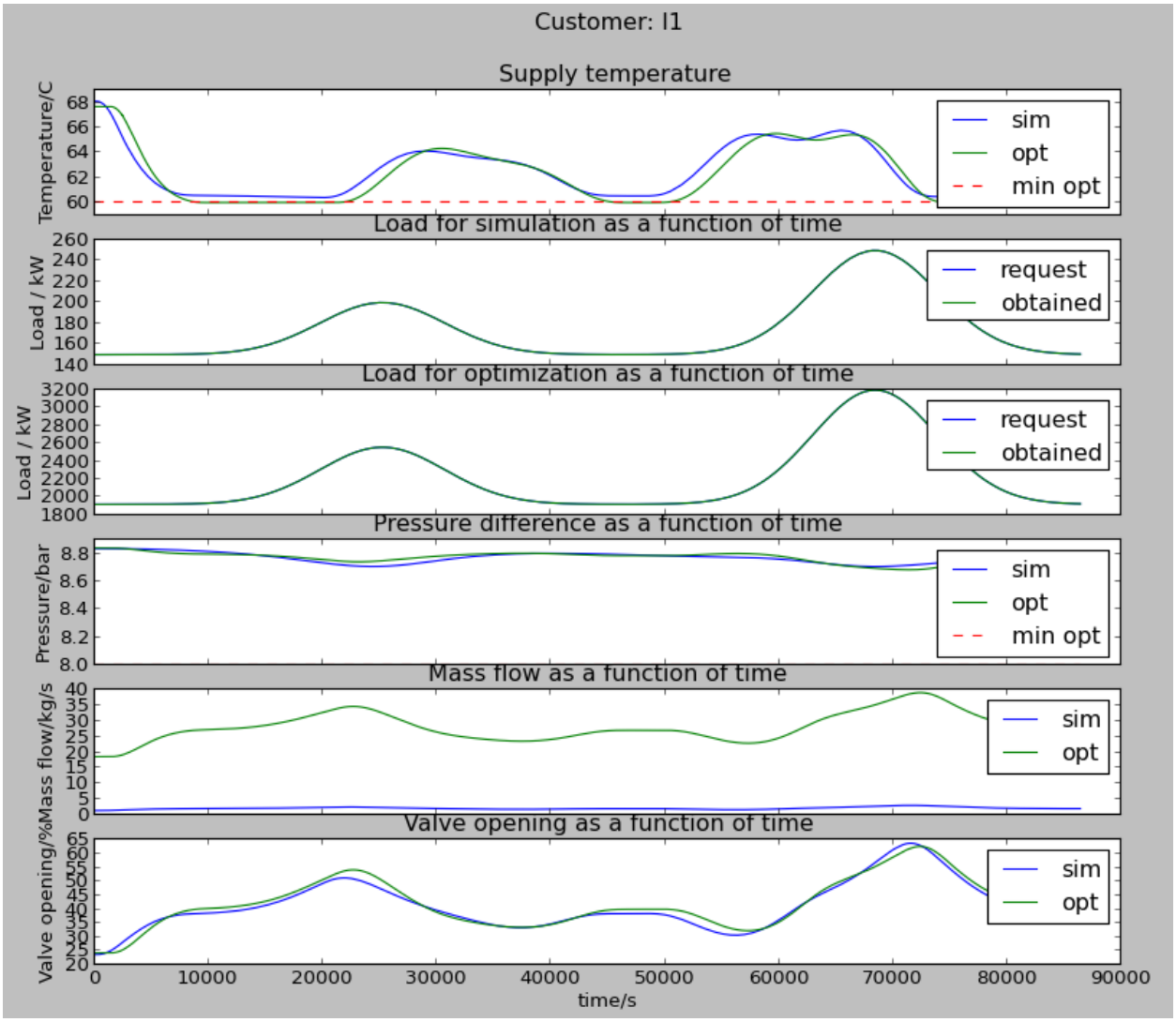

Figure 39 - Optimal trajectories for customer I1. NOTE: The "min opt" in the pressure difference equals 8.0.

We can see from the results for example at the closest customer, I1, that there is almost no delay. Another thing to observe is the supply temperature of the two remaining customers. Customer I1 receives its minimum requirement of supply temperature, whereas customer D is slightly below (oscillating). This is because of the errors introduces when performing aggregation.

7) Discussion

In this chapter, the previous results will be discussed and if it is possible to come to any conclusions based on them. We will also discuss the choice of method, limitations and assumptions and if they are affecting the results. Lastly, we will give some suggestions for future work.

7.1 Pressure included in the system

From the previous thesis (Larsson H. , 2015), there was not any pressure introduced in the network. This is an important factor to make the network more realistic, since it is the pressure that divides the mass flow between nodes and pressure difference is commonly controlled. From section 6.1 Pressure based system, we saw the impact on temperature and flows of introducing constraints on the pressure. This way of handling optimization problems can be very useful since some customers or key nodes need to fulfill a certain value on the pressure difference.

7.2 Finding optimal solutions

When doing optimization, the solver cannot always find an optimal solution. The reason for this can be many, but the most common ones in this framework are that the delay elements are not chosen properly or that the power of the producer is earlier set either too low or too high. For example, if all the customers require a total load of 100 kW and the producer is set to a maximum of 90 kW, it makes it impossible. However, it can also be because of the constraints. If for example there is a constraint saying that the mass flow in one of the pipes should not be higher than a certain value, it makes it impossible to reach the required load.

When we tried increasing the lengths of the pipes in the Graz case it did not work because the pipe and valve components need a start value for the pressure difference at the input and output ports.

Another thing to consider is that when increasing the lengths by for example a factor 10, the distance between the producer and the customer furthest away is almost 10 km. When the supply temperature is 60°C, the return temperature 43°C and the ambient temperature is between -5°C to 5°C according to Figure 15, what happens then with the 10 km distance is that the supply temperature for the customer furthest away will be lower than the required return temperature at 43°C. This makes it impossible to fulfill.

7.3 Flexibility of the framework

The last question of the thesis was to examine if the framework is flexible enough to change components and still find optimal solutions for different cases. This is possible if the network only contains connections between the following three components (see for example Figure 20):

1. Producer (with a load predictor)
2. Pipes
3. Customers

In this thesis, the pipes and producers have been changed. An example of this are the pipe models, which are explained in section 4.1 Component models, where we have used optimization pipes and simulation pipes. The same thing applies for the producer.

This thesis was conducted in parallel with another thesis worker. The thesis of that project was to develop a more advanced and realistic model that can simulate large networks consisting of hundreds of customers within a few minutes. If we were to use the models from that thesis it would give more accurate results, since the whole network would be more realistic. The models used in our project are somehow simplified and are mainly used for optimization, not simulation. Optimization problems should not be too complex and therefore our models are more suitable.

7.4 Errors introduced from aggregation

The aggregation methods are not perfect and they do introduce errors, see (Loewen, 2001) for more details**.** We can see the errors by looking at the simulation and optimization results in Figure 39 for example at the supply temperature. The blue line represents the temperature supplied in the real original network. This is not exactly synced with the green line.

If for example the original network has 100 customers and is aggregated to 3, this will give larger errors than for example aggregating down to 10. Each time a customer is removed or branched, some physical information is lost or assumed and this introduces errors. In this thesis, the framework was used on a real case network, Graz in Austria, which only contains 16 customers. When doing aggregation to 2 or 3 customers, the errors introduced between these are insignificant. We tried going down from 16 to 5 customers, but this requires a high computational time and memory. This was not possible to do on the hardware used in this thesis. But, based on the theory of both aggregation methods, the error should be smaller the more remaining customers you have left.

The error we do get though from the results can be caused of many things. One of these factors is that in the aggregation method we used, we introduced an option to exclude some key customers from being aggregated. For example, if customer K has some important information about pressure or load, this customer should be excluded from lumping since there is a risk of losing vital information. Another important factor is that we do not define how the structure of the network should be after aggregation.

Figure 30, Figure 34 and Figure 37 illustrates the optimal trajectories for the producer. What these three trajectories have in common is that the pressure difference is constant. In the customer models there is a constraint on the pressure difference set to 8 bar, as can be seen as “min opt” from the figures in the result section. Since this value is never reached during optimization, the producer does not have to compensate this regard. However, if for example we were to introduce a constraint on the customer to have a minimum requirement of 8.7 bar instead, this has to be compensated in the producer. This would make the producer increase its pressure difference to make sure that none of the pressure difference in any customer reaches below this value.

8) Conclusion

The main conclusions from this thesis are presented below:

- It is indeed possible to introduce pressure in the framework. This gives a more realistic network modelling.
- The developed framework is possible to be used on real cases. The large network can be aggregated and still satisfy customer load demands with a small error.
- The flexibility of the framework is limited to contain the same kind of parameters in producer, customer and pipes. If this is the case, the framework can be used to change components.
- Setting up proper values on pipes and supply temperatures are vital in order for the solver in the optimization to find solutions. Preferably the geometries and load profiles should be used from real cases and not made up since the values cannot be too unrealistic.

9) Future work

9.1 Danish method

There are two different aggregation methods that has been investigated in this thesis, the Danish method and the German method. The German method is fully implemented and used in the thesis, while the Danish method has only been partially implemented.

The equations for both serial aggregation and branch aggregation for the German method have been written in python, which is why this method is used. But for the Danish method, only the equations for the branch aggregation has been written while the serial aggregation has been neglected, since including both aggregation methods was out of the project objective.

To include the Danish method in the framework, one has to make sure that the network representation is using the “MultiGraphs” from the Python package “NetworkX”, along with the equation for serial aggregation.

9.2 Including more advanced simulation models

In this thesis we used relatively simple models for simulation since the case we focused on was a small area of the Austrian city Graz. This area only contained 16 customers and our models can simulate this size. However, if one were to simulate a much larger area with for example 200 customers, this is not possible with our developed framework. One approach to this is to use more advanced models for simulation and make sure that the parameters in each component is the same as in the developed framework of this thesis. If this is the case, larger networks can be simulated.

9.3 Using the framework for MPC

The strategy of model predictive control is a promising method in dealing with optimizing the district heating network since it allows real-time inputs to control the system. One example is the price based control, which means that the heat pump controller can shift its load to times with low electricity price, based on dynamic price information given by the utility. However, there are some work left that has to be done before this is possible with the developed framework. One example of this is to integrate the discretized part (UCP) of the optimization procedure, as explained in section 2.2 Production planning. Also, one can further develop the translator-codes to make the system more flexible.

10) Illustrations

11) Bibliography

Åkesson, J. (2008). *Optimica - An Extension of Modelica Supporting Dynamic Optimization.* Lund University, Automatic Control. Bielefeld: Modelica Association. Retrieved 05 10, 2016, from http://www.control.lth.se/Publication/ake08mod08.html

Bøhm, B., & Larsen, H. (2004). *Simple models of district heating systems for load and demand side management and operational optimisation .* Retrieved 02 01, 2016, from http://orbit.dtu.dk/en/publications/simple-models-of-district-heating-systems-for-load-and-demand-side-management-and-operational-optimisation(e965d211-ac81-4aff-97d1-9a06c4c9a3d1).html

Bøhm, B., Larsen, H., & Wigbels, M. (2004). *A comparison of aggregated models for simulation and operational optimisation of district heating networks.* ELSEVIER. Retrieved 2016 йил 20-04 from https://www.researchgate.net/publication/222078676_A_comparison_of_aggregated_models_for_simulation_and_operational_optimisation_of_district_heating_networks

Frederiksen, S., & Werner, S. (2013). District Heating and Cooling. Retrieved 02 06, 2016, from https://www.studentlitteratur.se/#9789144085302/District+Heating+and+Cooling

Larsen, H., Pålsson, H., Bøhm, B., & Ravn, H. (2002, may). Aggregated dynamic simulation model of district heating networks. *43*(8). ELSEVIER. Retrieved 04 14, 2016, from http://www.sciencedirect.com/science/article/pii/S0196890401000930

Larsson, H. (2015). *District Heating Network Models for Production Planning.* Lund. Retrieved 04 12, 2016

Larsson, P.-O., Runvik, H., Velut, S., Razavi, S. M., Nilsson, A., Bohlin, M., & Funkquist, J. (2014). *Decision Support for Short-Term Production Planning of District Heating using Non-linear Programming.* Värmeforsk. Retrieved 04 23, 2016, from http://www.varmeforsk.se/rapporter?action=show&id=4486

Loewen, A. (2001). *Entwicklung eines Verfahrens zur Aggregation komplexer Fernwärmenetze.* Dortmund: Fraunhofer. Retrieved 03 01, 2016

Modelica, A. (2016). *MODELICA*. Retrieved 04 14, 2016, from https://www.modelica.org/

Modelon. (2015). *Dymola*. Retrieved 02 02, 2016, from http://www.modelon.com/products/dymola/

Modelon. (2015). *JModelica.org User Guide.* Modelon. Retrieved 04 01, 2016, from http://www.jmodelica.org/api-docs/usersguide/JModelicaUsersGuide-1.17.0.pdf

NetworkX. (2016). *NetworkX*. Retrieved 03 01, 2016, from https://networkx.github.io/

Olsen, P. K., Christiansen, C. H., Hofmeister, M., Svendsen, S., & Thorsen, J.-E. (2014, 05). *Guidelines for Low-Temperature District Heating.* Dansk Fjernvarme. Retrieved 02 03, 2016, from file:///C:/Users/x-gerald.schweiger/Downloads/Guidelines%20for%20LTDH-final_rev1%20(7).pdf

Python. (2016). *Python*. Retrieved 03 01, 2016, from https://www.python.org/doc/essays/blurb/

SETIS. (2013). *4th Generation District Heating Technologies and Systems (4DH).* Retrieved 02 10, 2016, from https://setis.ec.europa.eu/energy-research/sites/default/files/project/docs/Projectdescription4DH.pdf