Optimization of District Heating & Cooling systems

D4.3: Dynamical physical system modelling

Delivery date: M18  Delivery type: Report
Version: V1.0
Dissemination level: Public
Main editor: Peter Lingman (OPN)
Contributors: OPN, IBM, TWT, LTU, LEN

The research leading to these results has received funding from the European Union’s Horizon 2020 Programme under grant agreement n° 649796.
Abstract (public) This deliverable provides physical system modeling components that are part of the OPTi-sim framework. A description of the modeling components and how they are configured is given. Modeling components that are developed consists of DHC-grid models and various building models. The development is done based on requirements established in WP2.

Keywords Modeling, Virtualization, DHC, building models, grey-box, black-box
EXECUTIVE SUMMARY

In this deliverable we explain our efforts in the field of dynamical physical modeling of DHC-networks and buildings. The models obtained here will serve as input to WP5 for control design and WP6 where all models will be integrated and validated. All modelling work done is based and controlled by the requirements and use-cases outlined in WP2.

Large-scale modeling of DHC-networks proved to be rather challenging, as expected, where one of the main challenges is to obtain a realistic simulation performance in terms of simulation time. Motivated by the vast manual labor to configure network models an automatic model simplification and configuration tool has been developed. This tool reads data from the geographical information system at the utility company (LEN) and produces a Modelica® model of the network. By this approach, a good life cycle management of the model is ensured since the model is built upon design data that is continuously updated by the utility companies at e.g. rebuilds or extensions of the DHC-network.

Modelling of buildings was performed using black-box and gray-box approaches. In the black-box approach, data analytics techniques were used on historical consumption data to estimate the consumption of a given building. In the gray-box approach a combination of physics based models and data analytics was used to the thermal dynamics of a building. A key challenge in the case of black-box models was the unavailability of low time resolution consumption data for most buildings served by LEN. Hence buildings were grouped into categories and a black box model was developed per category. In the case of gray-box modelling, limited transience in the data made the task of modelling the dynamic components in the system challenging. However, the proposed methodology was demonstrated using data from one of the pilot buildings.
Table of Contents

1. Introduction ......................................................................................................................................... 6

2. Building and consumer modeling ......................................................................................................... 6

   2.1 ‘Black-Box’ Modelling ................................................................................................................... 7
       2.1.1 Methodology ....................................................................................................................... 7
       2.1.2 Black-box modelling examples .............................................................................................. 9

   2.2 ‘Gray-Box’ Modelling ...................................................................................................................15
       2.2.1 Component models and parameter estimation ....................................................................17
       2.2.2 Integrated framework ..........................................................................................................22
       2.2.3 Gray-box Modelling Example ...............................................................................................23

3. Network modeling ..............................................................................................................................30

   3.1 General approach ........................................................................................................................30
   3.2 Components of the network model .............................................................................................30
   3.3 Pipe and node component ...........................................................................................................31
   3.4 Consumers ..................................................................................................................................32
   3.5 Plants ..........................................................................................................................................34

4. Top-level model components ..............................................................................................................42

   4.1 Model interfaces .........................................................................................................................42
   4.2 Model interoperability .................................................................................................................44

5. Generation of a network model ..........................................................................................................46

   5.1 generating models of DHC network ..............................................................................................46
   5.2 GUI main functions ......................................................................................................................47
   5.3 Figure properties ..........................................................................................................................50
   5.4 GUI Dymola model generation .....................................................................................................53
   5.5 Example of a model generation ...................................................................................................54

6. Conclusions and future work ...................................................................................................................58

References ..................................................................................................................................................59

6. Appendixes .........................................................................................................................................60

   6.1 Model configuration Gui functions ..............................................................................................60
1. INTRODUCTION

The OPTi-framework will enable engineers to design and improve DHC-networks in terms of efficient energy usage and operational economy. As described in the deliverables of WP2 one part of the OPTi-framework is the physical modeling component which enables a virtual representation of the DHC-network including for example distribution network, production units and buildings.

The physical modeling component will serve as input to WP5 for the design of optimal control strategies and system optimization. Physical models will also provide input to WP6 where they will be part of the simulation framework used for use case evaluation and verification.

Integration of modelling components is done using a co-simulation approach meaning that model interoperability is achieved in such a way that each modelling components can be simulated using their own numerical solvers. A brief overview of the co-simulation platform is given in section 4.2.

Using a strategy where the complete system model consists of several modelling components is very beneficial for a number of reasons. Firstly, it will allow model components from several model designers that can choose their preferred tool for modeling, i.e. for each task one can choose the best available simulation framework. Another very important benefit is that distribution of computational load is inherit in the co-simulation approach thereby improving simulation performance. In the OPTi project there are many alternative on how to divide the overall physical model. We have decided natural way is to divide the models into three main units consisting of buildings, network and control of network. This is also the approach taken in the OPTi-project.

In the following sections we will describe the developed modeling components and the results achieved in task 4.3 in WP4.

2. BUILDING AND CONSUMER MODELING

In this section, we describe building and consumer modelling methodologies developed within the OPTi project. We use two different modelling paradigms, referred to as black-box and gray-box respectively. Black-box modelling involves modelling the building using data analytics (machine learning) principles without any consideration of physics. Gray-box modelling, on the other hand, involves a combination of data analytics and physics to represent the building.

In this work, black-box modelling principles are applied to estimate the energy consumption (space heating demand) for a building using historical consumption data using machine learning techniques. Gray-box modelling involves the physical modelling of the building using principles of thermodynamics and heat transfer. The parameters in these models, however, are estimated using appropriate sensory data.

One of the core objectives in the OPTi project is to create a virtual DHC test-bed representing the DHC network in Luleå, Sweden. Models of buildings/consumers are required as an integral part of this test bed to address the various use cases (Cortes, 2015). Certain use cases such as "limitations in the grid" only require that the expected demand of the each consumer in the network is satisfied at all times. Therefore, to address such use cases, it is sufficient to use a model which outputs just the heating demand for each consumer.
Hence a black-box modelling approach suffices. However, certain other use cases such as “peak demand reduction” require an assessment of the impact in indoor temperature, together with the impact on the energy demand. This necessitates a modelling approach for buildings which not only models the heating demand but also the indoor temperature. A gray-box modelling approach is ideal for meeting such a requirement, since it uses physical principles to establish the relationship between indoor temperature and heating demand (and other exogenous variables such as outside temperature), while learning the parameters appearing in the physical model using sensor data for high fidelity.

It should be noted that development of gray-box models requires data such as indoor temperature, supply and return water temperature of water and outside temperature which may not be available for all the buildings in the network. In the OPTi project, we have identified a set of pilot buildings in Luleå which have been instrumented with sensors to record such data. Hence gray-box modelling is performed only for the pilot buildings, and black-box modelling is performed for the other buildings in the network.

In Section 2.1, we first introduce the black-box modelling methodology and demonstrate its application to the various buildings in the Luleå DHC network. The buildings are divided into categories based on the building type, and one black-box model is developed per building of a particular type. Next, we describe the gray-box modelling methodology and demonstrate it on one of the pilot buildings in the Luleå. Finally, we summarize the key findings from these modelling activities, and describe ongoing and future work to create models that are ready for integration with the OPTi co-simulation environment in WP6.

2.1 ‘BLACK-BOX’ MODELLING

We present the black-box modelling methodology in Section 2.1.1, and demonstrate its application to the various buildings in the Luleå DHC network in Section 2.1.2.

2.1.1 METHODOLOGY

The black-box modelling in this case refers to the different machine learning techniques that can be used to represent the energy consumption of buildings as a function of a set of features, and then try to predict the energy consumption at a particular time stamp based on the corresponding feature values for the timestamp. We first introduce the set of feature information corresponding to each time stamp present in the data, which we use to model the energy consumption. The features can be based on the calendar information, or the weather, or any other external meta information associated with the data. In particular, the features that we consider are the hour of day (which can take categorical values from 1 to 24), the ambient temperature (numerical) and the yearly-weekid (which can take categorical values from 1 to 53). The input data is the energy consumption at hourly resolution for different categories of buildings in Luleå, namely, single family domestic building (FV12), residential building (FV13), ground heating (FV14), public buildings (FV15) and trading centre (FV16). Out of these, the details of modelling for FV12 buildings have been already presented in D3.1 and D3.3 and are not included in this report.

Different machine learning techniques have been used to model the energy consumption of the users. We have primarily focussed on three models, the standard regression, the Support Vector Regression (SVR), and the randomforest (RF) methods. The standard regression method (Cortes & Vapnik, 1995) assumes the feature to be a linear (or quadratic or any polynomial of higher degree) combination of the features and estimates the coefficients based on the training data. The Support vector regression (Cortes & Vapnik, 1995) methodology tries to construct a hyperplane or a set of hyperplanes, which can be used for classification or regression of the data points. Intuitively, a good separation is achieved by the hyperplane that has the largest
distance to the nearest training-data point of any class, since a larger margin is supposed to give higher confidence of the classification/regression prediction. On the other hand, the random forest methodology (Ho, 1995) and (Ho, The Random Subspace Method for Constructing Decision Forests, 1998) constructs multiple decision trees to represent the data at the time of training; the output for a test point is some average of the classes (or the predicted values) of the individual trees.

Whatever methodology used, the objective is to use part of the data as training set, which we select randomly using train-test split of 75%-25%, and predict the individual energy consumption of each house, as well as the average energy consumption across all houses, both for hot water and space heating.

Different error measures are used in practice to study the efficacy of a modelling algorithm. The most common are the mean absolute error (MAE), and the root mean square error (RMSE). Suppose \( \mathcal{T} \) denote the test set, and each test point \( i \in \mathcal{T} \) is associated with the actual (observed) value \( y(i) \) and the predicted value \( \hat{y}(i) \). Let \( |\mathcal{T}| \) denote the number of test points in \( \mathcal{T} \). Then, the MAE and RMSE are respectively given as follows:

\[
\text{MAE} = \frac{1}{|\mathcal{T}|} \sum_{i \in \mathcal{T}} |y(i) - \hat{y}(i)|
\]

\[
\text{RMSE} = \frac{1}{|\mathcal{T}|} \sqrt{\sum_{i \in \mathcal{T}} (y(i) - \hat{y}(i))^2}
\]

Normalizing the MAE and RMSE facilitates the comparison between datasets or models with different scales: the range or the mean of the measured data is a common choice of the normalisation factor. Here we denote them respectively as NMAE(R) and NMAE(M). Similarly, the normalized RMSE are denoted by NRMSE(R) (normalized by range) and NRMSE(M) (normalized by mean). If \( y_{\text{max}} \) and \( y_{\text{min}} \) respectively denote the maximum and minimum response value in the test set, then the expressions are given as follows:

\[
\text{NMAE}(M) = \frac{\sum_{i \in \mathcal{T}} |y(i) - \hat{y}(i)|}{\sum_{i \in \mathcal{T}} |y(i)|}
\]

\[
\text{NMAE}(R) = \frac{\sum_{i \in \mathcal{T}} |y(i) - \hat{y}(i)|}{|\mathcal{T}|(y_{\text{max}} - y_{\text{min}})}
\]

\[
\text{NRMSE}(M) = \frac{|\mathcal{T}| \sum_{i \in \mathcal{T}} (y(i) - \hat{y}(i))^2}{\sum_{i \in \mathcal{T}} y(i)}
\]

\[
\text{NRMSE}(R) = \frac{\sum_{i \in \mathcal{T}} (y(i) - \hat{y}(i))^2}{(y_{\text{max}} - y_{\text{min}})}
\]
Out of these measures, NMAE(M) and NRMSE(M) are the ones that are most common in practice for determining the efficacy of machine learning algorithms.

2.1.2 Black-box modelling examples

We next present our experimental results. We first give some details on the data set in Section 2.1.2.1 and show the results in Section 2.1.2.2.

2.1.2.1 Data set description

As stated in Section 2.1.1, the input data is the energy consumption at hourly resolution for different categories of buildings in Luleå, namely, single family domestic building (FV12), residential building (FV13), ground heating (FV14), public buildings (FV15) and trading center (FV16). Out of these, the details of modeling for FV12 buildings have been already presented in D3.1 and D3.3 and are not included in this report.

We list below the details of the data we use in our modeling for the different categories of buildings:

- Residential Building (FV13):
  - Lingonstigen 209: This building consists of 100-110 apartments. The data set for this building contains the total energy consumption as well as the temperature at hourly resolution during the period Dec 2013-March 2016.
  - Lingonstigen 165: This building also consists of 100-110 apartments. As before we have the total energy consumption data as well as the temperature data at hourly resolution; the period however is March 2014 to March 2016.
  - Seminarigetan 16: This building consists of 55-60 apartments. The data set for this building contains the total energy consumption as well as the temperature at hourly resolution during the period May 2015-March 2016.
  - Östra malmgatan 16: This building consists of 60 apartments. The data set for this building contains the total energy consumption as well as the temperature at hourly resolution during the period May 2015-March 2016.

- Ground Heat (FV14)
  - Storgatan 50 & Skomakargatan 19: (resolution of 10 mins; April 2015-March 2016)

- Public Buildings (FV15)
  - For each of the following buildings, our data set consists of the total energy consumption as well as the temperature at hourly resolution during the period April 2015-March 2016
    - City House
    - Kulturens Hus
Our data set for this trading center consists of the total energy consumption as well as the temperature at hourly resolution during the period Nov 2014-June 2015.

We use the regression, SVR and RF routines available in the statistical computing package R. In SVR, a proper value needs to be chosen for "C", the penalty factor. If the value is too large, we have a high penalty for non-separable points and we may store many support vectors and overfit. If it is too small, we may have underfitting. We experimented with different parameters for the support vector regression and finalized on the parameter values, which were giving decent prediction performance. In our run, we set the penalty factor to 1000. The other tuning parameter epsilon is set to 0.1; the kernel is set to radial basis function. For the random forest runs, the number of trees used is 50, while the nodesize (minimum number of data points in each node) is set to 10.
2.1.2.2 RESULTS

In this section, we present our modeling result for each of the building categories. Figure 1 shows the consumption profiles of the four different buildings for FV13 category. For this figure (as well as all the subsequent figures in this subsection), the x-axis for the plots shows the hour index starting from the time point from which we have the data. The y-axis shows the energy consumption value of the building for that hour.

![Figure 1: Hourly consumption profile of different buildings of FV13 category](image)

The profile clearly shows a seasonal variation. As stated before, the time window for the Lingonstigen 209 data is Dec 2013-March 2016. We clearly observe that the consumption is pretty high during the winter months (Dec-Mar), while it goes down in the summers (May-August). Similar seasonal variation is observed in the profile for all the four buildings: the data for Lingonstigen 165 is for the period March 2014-March 2016, whereas for Seminarigetan 16 and Östra malmgatan 16 we have the data for May 2015-March 2016 only. To capture this seasonal behavior in the consumption, we include the \textit{yearly-weekid} as feature.

Table 1 shows the prediction results for the FV13 buildings. As stated before, we train our model using a random 75% of the total data for that building, and test on the remaining 25% of the data. We see that our Random Forest model gives close to 88% modeling accuracy if we take NMAE(M) as our error measure.
Table 1. Total energy consumption prediction for FV13 (entries in %)

<table>
<thead>
<tr>
<th></th>
<th>Regression</th>
<th>SVR</th>
<th>RF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NMAE (Mean)</td>
<td>NMAE (Range)</td>
<td>NMAE (Mean)</td>
</tr>
<tr>
<td>Lingo-nstigen 209</td>
<td>15.97</td>
<td>5.53</td>
<td>20.87</td>
</tr>
<tr>
<td>Lingo-nstigen 165</td>
<td>15.09</td>
<td>5.08</td>
<td>21.91</td>
</tr>
<tr>
<td>Semin-arigeta n 16</td>
<td>14.83</td>
<td>4.18</td>
<td>19.41</td>
</tr>
<tr>
<td>Östra malm-gatan16</td>
<td>15.34</td>
<td>5.52</td>
<td>19.49</td>
</tr>
</tbody>
</table>

We next move to modeling the ground heating consumption. Here we have the power consumption corresponding to ground heating at two locations Storgatan 50 & Skomakargatan 19 at ten minutes resolution during the period April 2015 to March 2016. Figure 2 shows the consumption for the two sites. The plot for Skomakargatan 19 shows the consumption is very less during the summer (June to September). However there seemed to be some wrong values in the outdoor temperature values of the data for Storgatan 50: there were only two readings in the span of the entire year at which the outside temperature showed sub-zero values. Thus for our modeling we used the outdoor temperature values that came with the data for residential houses.

![Figure 2](image-url)
Table 2 shows the prediction results for the FV14 locations, using 75-25% random split of train and test data. We see that our Random Forest model gives close to 81% modeling accuracy if we take NMAE(M) as our error measure.

Table 2: Power consumption prediction for FV14 locations (entries in %)

<table>
<thead>
<tr>
<th>Regression</th>
<th>SVR</th>
<th>RF</th>
</tr>
</thead>
<tbody>
<tr>
<td>NMAE (Mean)</td>
<td>NMAE (Range)</td>
<td>NRMS E (Mean)</td>
</tr>
<tr>
<td>Storgatan 50</td>
<td>29.72</td>
<td>5.7</td>
</tr>
<tr>
<td>Skomakargatan 19</td>
<td>40.25</td>
<td>11.46</td>
</tr>
</tbody>
</table>

We next present the results of modeling the power consumption for the public buildings. This includes the city house, the Solbackens retiring home, Kulturens and Repslagargatan 6. For each of these buildings, we have the power consumption at hourly resolution during the period April 2015 to March 2016. Figure 3 shows that the consumption profile nature for each of these buildings in this category is very similar, though the peak values vary depending on building size: the peak consumption for Kulturens hus goes as high as 1300 KW in the winter season (Figure 3 (b)), while that for Solbackens is nearly 190KW (Figure 3(c)).
Figure 3. Consumption profile of the public buildings (FV15 category)

Table 3 shows the prediction results for the FV15 buildings. Similar to the other categories, we train our model using a random 75% of the total data for that building, and test on the remaining 25% of the data. We see that our Random Forest model gives at least 89% modeling accuracy if we take NMAE(M) as our error measure.

Table 3 Total energy consumption prediction for FV15 (entries in %)

<table>
<thead>
<tr>
<th></th>
<th>Regression</th>
<th>SVR</th>
<th>RF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NMAE (Mean)</td>
<td>NMAE (Range)</td>
<td>NRMS E (Mean)</td>
</tr>
<tr>
<td>Cityhouse</td>
<td>13.15</td>
<td>2.99</td>
<td>19.17</td>
</tr>
<tr>
<td>Kulturenshus</td>
<td>17.42</td>
<td>3.32</td>
<td>23.63</td>
</tr>
<tr>
<td>Solbackens reti. home</td>
<td>17.98</td>
<td>4.49</td>
<td>23.07</td>
</tr>
</tbody>
</table>
We finally present the modeling results the trading center building Logementet, which corresponds to the FV16 category. Here we have the power consumption at hourly resolution during the period November 2014 to June 2015. Figure 4 shows that the consumption profile is high in the winter (Nov-Jan) while it reduces in the latter months.

Table 4 shows the prediction results for the FV16 location Logementet, using 75-25% random split of train and test data, as before. Our Random Forest model gives close to 82% modeling accuracy if we take NMAE(M) as our error measure.

Table 4, Power consumption prediction for FV16 location (entries in %)

<table>
<thead>
<tr>
<th></th>
<th>Regression</th>
<th>SVR</th>
<th>RF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NMAE (Mean)</td>
<td>NMAE (Range)</td>
<td>NRMS E (Mean)</td>
</tr>
<tr>
<td>Logementet</td>
<td>25.58</td>
<td>4.5</td>
<td>34.73</td>
</tr>
</tbody>
</table>

2.2 ‘Gray-Box’ Modelling

We first describe the system that is to be modelled using the gray-box approach. A schematic view of the system is sketched in Figure 5, which consists of both physical components and control components. These components can be categorized as “primary-side” components, “secondary-side” components, or “primary-secondary-side components”.

**Figure 4, Consumption profile of Logementet trading center (FV16 category)**

Table 4, Power consumption prediction for FV16 location (entries in %)
The “primary-side” components are the ones which are completely contained on the primary (network) side. In Figure 5, the only “primary-side” component is the primary valve. The “secondary-side” components are those that completely lie on the secondary (building) side and include pumps, valves, radiator, building, and building internal temperature controller (C1). The “primary-secondary-side” components are those that are partly contained in both the primary and secondary sides. These include the substation and the secondary supply temperature controller (C2).

The recipe used in this work for gray-box modelling of the above mentioned system is as follows. Firstly, component models for all three categories above are developed using physical principles. Next, the unknown parameters appearing in the models are empirically estimated using data. The component models, using the estimated parameter values are then concatenated to provide a model of the system. Each of these steps is described separately in the sub-sections below. It should be noted that Figure 5 represents a configuration with only one secondary water loop. The presented framework is easily amenable to model configurations involving multiple secondary loops. Further, it should also be noted that the component called “building” refers to an effective collection of zones/spaces that are being heated or cooled from the DHC network. In the case of a building with multiple zones and multiple physical radiators, the component “radiator” from the modelling perspective represents an effective heat exchange and controller C1 represents and effective indoor temperature controller for the effective building.

![Figure 5, System to be modelled using the gray-box approach, as shown within the boundary created by dashed lines](image-url)
2.2.1 COMPONENT MODELS AND PARAMETER ESTIMATION

We first derive physical models for each of the physical and control components within the system boundary in Figure 5. The nomenclature used is defined in Table 5.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \dot{m}_P(k) )</td>
<td>Primary water mass flow rate (kg/s) at time instance ( k )</td>
</tr>
<tr>
<td>( T_{P,S}(k) )</td>
<td>Primary water supply temperature (C) at time instance ( k )</td>
</tr>
<tr>
<td>( T_{P,R}(k) )</td>
<td>Primary water return temperature (C) at time instance ( k )</td>
</tr>
<tr>
<td>( \dot{m}_S(k) )</td>
<td>Secondary water mass flow rate (kg/s) at time instance ( k )</td>
</tr>
<tr>
<td>( T_{S,S}(k) )</td>
<td>Secondary water supply temperature (C) at time instance ( k )</td>
</tr>
<tr>
<td>( T_{S,R}(k) )</td>
<td>Secondary water return temperature (C) at time instance ( k )</td>
</tr>
<tr>
<td>( T_Z(k) )</td>
<td>Zone temperature (C) at time instance ( k )</td>
</tr>
<tr>
<td>( T_{SP}(k) )</td>
<td>Zone set-point temperature (C) at time instance ( k )</td>
</tr>
<tr>
<td>( T_{oa}(k) )</td>
<td>Ambient temperature at time instance ( k )</td>
</tr>
<tr>
<td>( T_{S,S,ref}(k) )</td>
<td>Secondary water supply temperature reference (C) at time instance ( k )</td>
</tr>
<tr>
<td>( u_P(k) )</td>
<td>Primary valve control signal (0-100%) at time instance ( k )</td>
</tr>
</tbody>
</table>

2.2.1.1 PRIMARY VALVE

The primary valve is not contained within the system boundary and hence modelling it is outside the scope of the gray-box modelling work presented here. Models of this component were developed within the scope of the network modelling activity presented in Section 3 of this document.

2.2.1.2 RADIATOR

Figure 6 shows a schematic representation of the radiator. At any time instance, \( k \), we denote the rate of heat transfer from the radiator to the building by \( \dot{Q}_{heat}(k) \). Using the first law of thermodynamics (Holman, 1992), we infer that:

\[
\dot{Q}_{heat}(k) = \dot{m}_S(k) c_p (T_{S,S}(k) - T_{S,R}(k))
\]

Equation 7

Here, \( c_p \) denotes the specific heat capacity of water, which is approximately equal to 4.18 kJ/kg-K. Using the concept of log mean temperature difference, the heat transfer rate can also be expressed as:
\[ Q_{\text{heat}}(k) = h_R \frac{T_{S,S}(k) - T_{S,R}(k)}{\ln \left( \frac{(T_{S,S}(k) - T_z(k))}{(T_{S,R}(k) - T_z(k))} \right)} \]

Equation 8

Here, \( h_R \) denotes the effective heat transfer coefficient between the radiator and the building. Equations (1) and (2) can be combined to write:

\[ \dot{m}_S(k) c_p(T_{S,S}(k) - T_{S,R}(k)) = h_R \frac{T_{S,S}(k) - T_{S,R}(k)}{\ln \left( \frac{(T_{S,S}(k) - T_z(k))}{(T_{S,R}(k) - T_z(k))} \right)} \]

Equation 9

Only one unknown parameter, \( h_R \), appears in Equation 8. It can be learnt using linear regression from time-series historical data for \( \dot{m}_S(k), T_{S,S}(k), T_{S,R}(k), \) and \( T_z(k) \).

\[ \frac{dT_z(t)}{dt} = \frac{1}{R_z C_z} (T_\infty(t) - T_Z(t)) + \frac{1}{C_z} \dot{Q}_{\text{int}}(t) + \frac{1}{C_z} \dot{Q}_{\text{heat}}(t) \]

Equation 10

**2.2.1.3 BUILDING**

Figure 7 shows a schematic representation of the building component. We use a resistive-capacitive [6] framework to model the underlying thermodynamics. In such a framework, the building is assumed to have an effective thermal capacitance \( C_z \) and an effective thermal resistance \( R_z \) acting on the heat transfer between the building and the ambient. The first law of thermodynamics [5], applied using this framework results in the following dynamic evolution of the internal temperature at time instance \( t \):

\[ \frac{dT_z(t)}{dt} = \frac{1}{R_z C_z} (T_\infty(t) - T_Z(t)) + \frac{1}{C_z} \dot{Q}_{\text{int}}(t) + \frac{1}{C_z} \dot{Q}_{\text{heat}}(t) \]
In the above equation, \( \dot{Q}_{\text{int}}(t) \) represents the rate of heat transfer to the building through internal sources such as occupants and heat generating appliances. In discrete time, using a time step of size \( \Delta T \) seconds, Equation 10 can be written as:

\[
\frac{T_z(k+1) - T_z(k)}{\Delta T} = \frac{1}{R_z C_z} (T_\infty(k) - T_z(k)) + \frac{1}{c_z} \dot{Q}_{\text{int}}(k) + \frac{1}{c_z} \dot{Q}_{\text{heat}}(k)
\]

Equation 11

The quantity \( \dot{Q}_{\text{heat}}(k) \) appearing in Equation 11 can be computed from either of the Equation 7 or 8. The unknown parameters in (5) are \( R_z \), \( C_z \) and \( \dot{Q}_{\text{int}} \) that can be learnt using Equation 11 through time-series regression from historical data.

**Figure 7, Schematic of the building component**

### 2.2.1.4 BUILDING INTERNAL TEMPERATURE CONTROLLER

In this work, we model the internal temperature controller (controller C1 in Figure 5) and the mass flow rate control device (pump/valve) as one component. This is schematically shown in Figure 8. The controller responds to a signal which is the error between the actual indoor temperature (\( T_z \)) and the indoor temperature set-point (\( T_{SP} \)). The actual response is the secondary water mass flow rate (\( \dot{m}_S \)). For example, at any time instance, if \( T_z < T_{SP} \), it results in a larger secondary mass flow rate.

The controller action is a function of the control law implemented. For example, for a proportional (P) controller, the governing relationship is:

\[
\dot{m}_S(k) = K_z (T_{SP}(k) - T_z(k)) + c_z
\]

Equation 12

Here, \( K_z \) is the proportionality constant and \( c_z \) is the constant/DC offset in the controller. If these constants are not known, they can be learnt using regression analysis from historical data for \( \dot{m}_S(k) \), \( T_{SP}(k) \), and \( T_z(k) \). For other types of control actions, such as Proportional Integral (PI) or Proportional Integral Derivative (PID), equation (6) should be adjusted accordingly.
2.2.1.5 Substation

The substation is modelled as a heat exchanger, as schematically shown in Figure 9. It should be noted that the scope of the modelling here is limited to capturing the heat exchange associated with space heating of the building. Hence, modelling the hot water demand of the building is not considered.

Using the first law of thermodynamics [5], and assuming lossless heat transfer between the primary and secondary streams, we infer that:

\[ \dot{m}_S(k) c_p (T_{S,S}(k) - T_{S,R}(k)) = \dot{m}_P(k) c_p (T_{P,S}(k) - T_{P,R}(k)) \]

Equation 13

Using the concept of log mean temperature difference, the heat transfer rate can also be expressed as:

\[ \dot{m}_S(k) c_p \left( T_{S,S}(k) - T_{S,R}(k) \right) = h_s \frac{T_{P,S}(k) - T_{S,S}(k) - T_{P,R}(k) + T_{S,R}(k)}{\ln \left( \frac{T_{P,S}(k) - T_{S,S}(k)}{T_{P,R}(k) - T_{S,R}(k)} \right)} \]

Equation 14

Here, \( h_s \) denotes the effective heat transfer coefficient between the primary and secondary water streams. It can be learnt by the application of Equation 14 using linear regression from time-series historical data for any five of the six variables: \( \dot{m}_S(k), \dot{m}_P(k), T_{S,S}(k), T_{S,R}(k), T_{P,S}(k), \) and \( T_{P,R}(k) \). This is because, knowing only five of these quantities, the sixth can be obtained using Equation 13.
2.2.1.6 SECONDARY SUPPLY TEMPERATURE CONTROLLER

The secondary supply temperature controller (controller C2 in Figure 5) adjusts the opening \( u_p \) of the primary valve in order to drive the secondary supply temperature \( T_{SS} \) to a desired set-point \( T_{SS,ref} \), as schematically shown in Figure 10. For example, at any time instance, if \( T_{SS} < T_{SS,ref} \), the primary valve is opened more by the controller, resulting in a larger primary mass flow rate, and hence more primary energy becoming available.

It is a common practice (Gustafsson, 2008) to set the set-point temperature as a linear or piece-wise linear function of the ambient temperature \( T_\infty \). A linear relationship would be of the form:

\[
T_{SS,ref}(k) = K_{s,1} T_\infty(k) + c_{s,1}
\]

Equation 15

The controller action is a function of the control law implemented. For example, for a proportional (P) controller, the governing relationship is:

\[
u_p(k) = K_{s,2} \left( T_{SS,ref}(k) - T_{SS}(k) \right) + c_{s,2}\]

Equation 16

The relationship between \( T_{SS,ref} \) and \( T_\infty \) is colloquially referred to as the heating curve of the substation. In the case of a linear heating curve, given by Equation 15, the coefficients \( K_{s,1} \) and \( c_{s,1} \) are called heating curve constants. These coefficients can be assumed to be known. However, they can also be learnt using linear regression applied to Equation 15 using historical data for \( T_{SS,ref} \) and \( T_\infty \). In Equation 16, \( K_{s,2} \) is the proportionality constant and \( c_{s,2} \) is the constant/DC offset in the controller. If these constants are not known, they can be learnt using regression analysis from historical data for \( u_p(k), T_{SS,ref}(k), \) and \( T_{SS}(k) \). For other types of control actions, such as Proportional Integral (PI) or Proportional Integral Derivative (PID), Equation 16 should be adjusted accordingly.
2.2.2 INTEGRATED FRAMEWORK

The overall integrated gray-box modelling framework is shown in Figure 11, which combines the component level models presented in Section 2.2.1. The framework involves two steps. In the first step, unknown parameters in the framework are estimated based on historical data. In the second step, the estimated parameters are used as inputs to the set of parameterized component models derived above. The inputs and outputs to the framework are also shown in Figure 11. The recipe for integration, as summarized in the figure, would be applied in Deliverable 6.2, when integrating gray-box models with the OPTi co-simulation framework.

Figure 11, Overview of the integrated gray-box building modelling framework

\[ T_{S,S}(k+1) - T_S(k) = \frac{1}{R_s C_s} (T_{in}(k) - T_S(k)) + \frac{1}{C_s} \hat{Q}_{int}(k) + \frac{1}{C_s} \hat{Q}_{heat}(k) \]

\[ \hat{Q}_{heat}(k) = \dot{m}_S c_p (T_{S,S}(k) - T_{S,R}(k)) = h_R \ln\left(\frac{(T_{S,S}(k) - T_{S,R}(k))}{(T_{S,R}(k) - T_S(k))}\right) \]

\[ \dot{m}_S(k) c_p (T_{S,S}(k) - T_{S,R}(k)) = \dot{m}_P(k) c_p (T_{P,S}(k) - T_{P,R}(k)) \]

\[ T_{S,R}(k) = K_p T_{S,P}(k) - T_S(k) + c_z \]

\[ T_{S,S,ref}(k) = K_{s,1} T_{in}(k) + c_{s,1} \]

\[ u_P(k) = K_{s,2} (T_{S,S,ref}(k) - T_{S,S}(k)) + c_{s,2} \]

\[ \text{Learned unknown parameters/variables} \]

- \( R_s \): Thermal resistance for heat transfer between building and ambient
- \( C_s \): Thermal capacitance of building
- \( \hat{Q}_{int} \): Heat generated by internal sources (occupants, appliances, etc.)
- \( h_R \): Heat transfer coefficient between radiator and building air
- \( h_c \): Substation heat transfer coefficient
- \( K_p \): Zone temperature controller feedback gain
- \( c_z \): Zone temperature controller constant
- \( K_{s,1} \): Secondary supply set-point feedforward gain
- \( c_{s,1} \): Secondary supply set-point constant
- \( K_{s,2} \): Secondary supply controller feedback gain
- \( c_{s,2} \): Secondary supply controller constant

Historical sensor data
2.2.3 Gray-box Modelling Example

We demonstrate the gray-box framework described above on a real building from one of the pilot sites (Luleå, Sweden). Data available from the building is first used to learn the parameters of the model, component-wise, as described in Section 2.2.1. Next, these parameters are used in the integrated framework as shown in Section 2.2.2.

2.2.3.1 Building Description

The pilot site under consideration is located in Luleå, Sweden, and consists of eight different buildings, partly located on the streets Hallonstigen and Smultronstigen (Figure 12). These buildings are owned by Lulebo AB and are served by a single district heating substation. They consist of a total of 126 apartments, with each building having two or three floors with four to nine apartments in each entrance. These buildings were built in 1973 based on the 1967 Swedish building code.

In these buildings, indoor temperature sensors are installed in 72 of the 126 apartments. The indoor temperature sensors are wireless and communicate through radio. We also have measurements for hot water supply and return temperatures, flow rates and outside temperature.

![Figure 12, Location of the pilot building used for evaluation of the gray-box framework](image)

2.2.3.2 Data

For the example building, time-series historical data is available for the following variables from June 2015. These variables are: (i) primary water mass flow rate, (ii) primary water supply temperature, (iii) primary water return temperature, (iv) secondary water supply temperature, (v) secondary water return temperature, (vi) average indoor temperature across apartments (using a representative set of 72 apartments as mentioned in Section 2.2.3.1), and (vii) ambient temperature. In addition to these parameters, the coefficients corresponding to the "heating curve" and the control gains for the controller C2 (c.f. equations (9) and (10)) are also known and provided by the building energy manager. The time-series
data was extracted from the central database implemented in the OPTi project (Eikermann, 2016) and was available at a time resolution of 1 minute. Although data was available for a time window of approximately 10 months, for demonstration of the modelling methodology, we used data from 19 February 2016 to 24 February 2016.

### 2.2.3.3 Development of Component Models

Figure 13, Steps involved in the development of component models summarizes the steps involved in the development component models that were presented in Section 2.1.2. Each of these steps involves the computation or estimation of appropriate parameters/variables associated with the component models.

We now demonstrate the application of each of these steps on the example building under consideration.

**Step 1:**

Since 5 out of 6 variables in (7) are known with only \( \dot{m}_p(k) \), unknown we easily compute it using (7). It is plotted in Figure 14: Plot of mass flow rates. Primary and tap water flows are from data while the secondary flow rate is calculated from step 1., along with the other flow rates in the system, i.e. the primary side mass flow rate and the tap water mass flow rate.

**Figure 13, Steps involved in the development of component models**
Step 2:

The building manager has mentioned that the indoor temperature controllers in the apartments use Proportional Integral Derivative (PID) logic. Therefore, we assume that the "effective" action on the secondary side mass flow rate to control the average indoor temperature is also PID expressed by the following relationship (in continuous time):

\[ \dot{m}_S(t) = K_P e(t) + K_I \int_0^t e(t) \, dt + K_d \frac{de(t)}{dt} + m_0 \]

Equation 17

where, \( e(t) = T_{SP}(t) - T_Z(t) \)

Equation 18

We attempt to learn the P, I and D gains, \( K_P, K_I, \) and \( K_d, \) respectively, of this effective controller and the constant offset \( m_0 \) using data. The results obtained are as follows:

\[ K_P = -0.3596, \ K_I = -0.000463, \ K_d = -71.5292, \ m_0 = 10.917 \]

Hence, the P, I and D gains turn out to be negative, which is anomalous. Figure 15 shows a plot of the calculated (in step 1) vs. estimated (using the above coefficient values) secondary mass flow rates. It can be seen that the fit between the calculated and estimated values is also poor. The negative control gains and the poor fit can be attributed to the fact that the data used to learn the gains shows negligible transience because the set-point temperature of the indoor temperature in the apartments is maintained constant. Since the controller in Equation 17 is a dynamic controller (includes the derivative of indoor temperature), it requires sufficiently high level of transience in the indoor temperature for accurate estimation of the control gains. To address this issue, controlled experiments are planned in the test building in the future, where transience in indoor temperature will be sought by varying the set-point temperature over time. The above-mentioned anomalies are expected to be resolved using data from such experiments. These new results will be reported in Deliverable 6.2.
Figure 15, Plot of calculated vs estimated secondary mass flow rates

It is important to note that the coefficients are coming out to be negative. This should not happen. Furthermore, the fit to the data is also not very good. We believe that the accuracy of the model can improved significantly if we use more transient data. The data we are getting now is from the whole system operating in steady state.

**Step 3:**

Since the heating curve coefficients are directly available from the building energy manager, these need not be learnt. To check whether the provided coefficients are correct, we compare the secondary supply temperature reference \( T_{S,S,ref} \) with the actual secondary supply temperature \( T_{S,S} \). The result is shown in Figure 16. We note that the match between these plots is satisfactory except around midnight. Upon further investigation, the building energy manager confirmed that this is because \( T_{S,S,ref} \) is increased by \( 1^\circ C \) around midnight over and above the value based on the heating curve. Accounting for this correction, we conclude that \( T_{S,S,ref} \) is close to \( T_{S,S} \), confirming that the heating curve coefficients provided are correct.
For this step also, we were provided with the controller gains for the PID controller (C2) controlling the primary valve opening based on the difference between the actual and reference secondary supply temperatures. Hence, these gains need to be learnt. We used these gains to calculate the values of $u_p$, as shown in Figure 17. It is important to note that the values shown are relative to the starting value, when the controller is switched ON, since the controller has a non-zero integral gain. We do not have an opportunity to verify the accuracy of the computation of $u_p$, since ground truth values for $u_p$ are not available. In the future, experiments are planned to measure $u_p$, so as to perform the above-mentioned accuracy verification. These results shall be reported in Deliverable 6.2.
Step 5:
Using data for the test building, we find that the argument of the logarithm in Equation 14 turns out to be negative. This is because for some time instances, \( T_{P,R} < T_{S,R} \). This could be result of losses in the substation, which are not accounted for, when applying the first law of thermodynamics to obtain Equation 13. Hence Equation 14 cannot be used to model the substation for the test building. To address this issue, we consider a simplified variant Equation 19 of Equation 14, which uses a linear temperature difference instead of log temperature difference.

\[
\dot{m}_S(k)c_p \left( T_{S,S}(k) - T_{S,R}(k) \right) = h_S' \left( \frac{T_{P,S} + T_{P,S}}{2} - \frac{T_{S,S} + T_{S,R}}{2} \right)
\]

Equation 19

The heat transfer coefficient, \( h_S' \) appearing in Equation 19, was estimated from linear regression. To verify the accuracy of the estimate, we compare the measured value of the secondary return temperature with the value computed using Equation 19. We find that the two plots almost coincide, Figure 18.

As a future improvement to the substation model, we plan to include considerations such as heat losses to the surroundings.

Step 6:
The radiator heat transfer coefficient, \( h_R \) can be computed directly from Equation 5, for each time instance, since all the other variables in that equation are known. However, a plot of the computed values of \( h_R \), versus the secondary mass flow rate (\( \dot{m}_S \)) shows a strong linear behavior (Figure 19). This leads us to the conclusion that the radiator heat transfer coefficient, \( h_R \) is not constant, and its value at any time instance \( k \), can be computed from the following expression:

\[
h_R(k) = \alpha_R \dot{m}_S(k)
\]

Equation 20
The value of $\alpha_R$ can be obtained by fitting a line between $h_R$ and $\dot{m}_S$ in Figure 19. We estimated it as 0.2434 kJ/kg-K.

![Figure 19: Comparison of radiator heat transfer coefficient (computed value) and secondary mass flow rate](image)

**Step 7:**

We perform time-series regression on Equation 11 to estimate the values of $R$ and $C$. We assume that the internal heat generation, $\dot{Q}_{int}$, is negligible. The computed values of $R$ and $C$ are $1.2903 \times 10^{-4} \text{ K}/\text{kW}$ and $7.182 \times 10^{9} \text{ kJ}/\text{K}$ respectively. The predicted indoor temperature using these values captures the overall trend of temperature variation quite well (Figure 20). For instance, temperature during the day increases while it falls during the night. However, given that Equation 11 is a dynamic equation, the fit can be improved significantly by using indoor temperature data with more transience. To address this issue, controlled experiments are planned in the test building in the future, where transience in indoor temperature will be sought by varying the set-point temperature over time, based on which new results will be reported in Deliverable 6.2.

![Figure 20 Comparison of actual and estimated indoor temperature values](image)
3. NETWORK MODELING

The approach taken for DHC-network modeling is presented and each model component is explained. An example of generating a physical model using the automatic model configuration tool is also given.

3.1 GENERAL APPROACH

Physical modeling of large scale process can be made in many different tools and at many different levels of complexity. Based on the requirements developed in WP2 the level of details needed and the main functionality of the models is set. In the choice of the modelling tool one of the most important features is tool interoperability in order to be able to choose best available tool for different tasks (Lingman, 2013). In this project tool interoperability is achieved by using a co-simulation platform and utilizing standardized functional mock-up units (FMU/FMI). For the network modelling Modelica was chosen as modeling language motivated by:

- Modelica is fully object oriented which is a requirement for large scale modelling
- Modelica is equation oriented, i.e. very suitable for physical modeling
- Dymola supports FMU standard rather good
- Dymola has efficient numerical handling of large equation systems
- In-line with other ongoing activities e.g. (IAE EBS Annex 60, 2016)

Given the nature of a DHC network it was rather early discovered that it will not be feasible to efficiently manually configure a model. For example, the pilot network in the city of Luleå, Sweden consists of roughly 10000 substations (also called customers) and 50000 pipe sections of varying type. Furthermore, the network consists of sensors, producers, and actuators, all normally very geographically distributed.

In this deliverable required modeling components, automatic model configuration, and model simplification and adaptivity is developed.

3.2 COMPONENTS OF THE NETWORK MODEL

The following three main components are implemented in Modelica and explained in this section:

- District heating pipe for water distribution.
- Substation at the consumer to extract energy from the DHC-grid.
- Production units that produce warm DHC water.
- Pressure increasing units (normally an arrangement of valves and pumps, also called pumping stations)
3.3 PIPE AND NODE COMPONENT

Pipes and nodes are bi-directional and in each direction forward/supply and return flow is represented. Pipes have the following characteristics and parameters volume, temperature, temperature delay, flow resistant, heat loss and height difference.

Pressure in port_a and port_b, see Figure 21, will differ depending on the dynamic pressure due to flow and static pressure due to height difference. For short pipes the pressure difference can be ignored for computational reasons.

Heat loss is calculated due to a set constant, temperature of the water in the pipe and surrounding temperature that is set to a fix temperature. The heat loss factor is calculated according to (13941:2009+A1:2010, 2010). For temperature delay a transmission line model is implemented to assure a propagation of temperature.

Nodes are components placed between each of the modelling components and nodes summarizes the flows and they can also represent either static or dynamic behavior in terms of pressure, see Figure 22.

Figure 21, Pipes and ports (square blue objects).
3.4 **Consumers**

Consumers or substations are represented by three different types.

1. Ordinary consumers or black-box which represents most of the substations in the network and which only have a certain power demand characteristic.
2. Pilot consumer or gray-box where the valve opening and return temperature are simulated simultaneously from building models in the co-simulation platform.
3. Hot water valve consumers are used in special version of the model where the hot water valves (also tap water valves) are of special interest. This consumer is named HW (hot water).
Figure 24, Pilot, gray-box.

Figure 25, Consumer HW.
3.5 PLANTS

Plants include producers such as boilers, pumps and valves. Each location in the Luleå grid is unique hence requiring unique models. The models are based on drawings and information from LEN. In some situations models are simplified, e.g. valve that are normally not used are removed in order to improve performance. The same general producer model is used for all producers. It consist mainly of a heater and a shutdown valve. There are also sensors for temperature and flow.

Figure 26, Producer of warm water.

KVV (combined heat and power plant is KraftVärmeVerk in Swedish hence the abbreviation KVV for CHP) and HVC2 are located at the same site. KVV has both return and forward pumps. HVC2 has only return pumps. There are also a number of valves that has to be opened and closed to direct flow as required by the situation.
Figure 27, KVV and HVC2 including expansion vessel.
Figure 28, KVV the heat exchange from Luleå Kraft AB (LUKAB) boilers running on oil and gas (steel mill process gas).

Figure 29, HVC2 two boilers one oil and one mixed oil and gas.
Figure 30, HVC1 Pumping station and auxiliary boilers.

Figure 31, HVC1, three oil boilers and two electrical boilers.
Figure 32, Pumping station TSP2 and Auxiliary boiler HVC4.

Figure 33, HVC4 Auxiliary boiler oil and wooden powder.
Figure 34, HVC5 Pumping station and auxiliary boilers.

Figure 35, HVC5 three oil boilers.
Figure 36, TSP1 Pump station.
Figure 37, TSP3 Pump station.
4. TOP-LEVEL MODEL COMPONENTS

In this chapter the high level interfaces between different models will be explained together with the approach taken to achieve model interoperability using co-simulation.

4.1 MODEL INTERFACES

In Figure 38 a typical substation is shown including a heat exchanger that transfers heat from the DHC grid to the building. In the modelling approach taken for OPTi-sim, separation between building model and grid is done at the substation. In principle this approach is motivated by model modularity and simulation performance. By separating the DHC-grid model from the consumer models independence is created that will assure a plug and play feature. A variety of consumer models can easily be docked on to the grid model depending on the needs. Simulation performance is improved by the possibility to distribute computational load using, for example, a co-simulation platform.

Two different interfaces are developed, one for gray-box buildings and one for black-box buildings. The main difference between the two interfaces is that in the gray-box model the primary valve is controlled directly by the TC_SS controller which is placed in the gray-box building model whereas in the black-box case the primary valve is controlled in the DHC-grid model based on a heating request from the black-box model.

Figure 38, Instrumentation of typical substation in Luleå (heating). Primary heating valve is controlled by a temperature controller (TC) controlling the secondary supply temperature (TI-SS). FI stands for flow indicator and TI stands for temperature indicator. Normally the circulation pump on the secondary side is running with constant speed but there is also some buildings that use pressure and/or flow control. Controlled circulation pumps can be found in buildings larger than one and two family houses.

The interface for one pilot plant is shown in Table 6. There are 4 pilot substations in the Luleå pilot and each have the same interface setup.
Table 6, Interface for one of the pilot building models (in this case street heating). Direction declares whether the signal is sent from the DHC-network to the gray-box building (input) or vice versa (output).

<table>
<thead>
<tr>
<th>Name</th>
<th>Object</th>
<th>Parameter</th>
<th>Direction</th>
<th>Unit</th>
<th>min</th>
<th>max</th>
<th>Default</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>C_652808_opening</td>
<td>C_652808</td>
<td>opening</td>
<td>input</td>
<td>%</td>
<td>0</td>
<td>100</td>
<td>50</td>
<td>Primary valve control, Storgatan 50 ground heating</td>
</tr>
<tr>
<td>C_652808_T_r</td>
<td>C_652808</td>
<td>T_r</td>
<td>input</td>
<td>°C</td>
<td>0</td>
<td>100</td>
<td>60</td>
<td>Primary return temperature, Storgatan 50, ground heating</td>
</tr>
<tr>
<td>C_652808_T_s</td>
<td>C_652808</td>
<td>T_s</td>
<td>output</td>
<td>°C</td>
<td>0</td>
<td>150</td>
<td>C_652808.T_s</td>
<td>Primary supply temperature, Storgatan 50 ground heating</td>
</tr>
<tr>
<td>C_652808_m_flow_act</td>
<td>C_652808</td>
<td>m_flow_act</td>
<td>output</td>
<td>kg/s</td>
<td>0</td>
<td>10</td>
<td>C_652808.m_flow_act</td>
<td>Primary mass flow, Storgatan 50 ground heating</td>
</tr>
</tbody>
</table>

The interface for the black-box models is shown in Table 7. Each of the buildings in the Luleå DHC-network has a categorization number (FV) that is used in order to generate a realistically varying energy usage (consumption), Section 2.

Table 7, Interface between DHC-network model and black-box building models. The FV-value is used to generate a consumption power.

<table>
<thead>
<tr>
<th>Name</th>
<th>Object</th>
<th>Parameter</th>
<th>Direction</th>
<th>Unit</th>
<th>min</th>
<th>max</th>
<th>Default</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>FV11</td>
<td>input</td>
<td>%</td>
<td>0</td>
<td>100</td>
<td>50</td>
<td>Consumer profile mining and manufacturing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FV12</td>
<td>input</td>
<td>%</td>
<td>0</td>
<td>100</td>
<td>50</td>
<td>Consumer profile one and two family houses</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FV13</td>
<td>input</td>
<td>%</td>
<td>0</td>
<td>100</td>
<td>50</td>
<td>Consumer profile residential buildings</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FV14</td>
<td>input</td>
<td>%</td>
<td>0</td>
<td>100</td>
<td>50</td>
<td>Consumer profile ground heating</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FV15</td>
<td>input</td>
<td>%</td>
<td>0</td>
<td>100</td>
<td>50</td>
<td>Consumer profile official building</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FV16</td>
<td>input</td>
<td>%</td>
<td>0</td>
<td>100</td>
<td>50</td>
<td>Consumer profile other business and service</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For the Majorca pilot the situations is slightly different since modelling focus is on building level and no interface to the distribution network needs to be considered.
4.2 MODEL INTEROPERABILITY

OPTi-Sim is based on the TWT Co-simulation Framework (Pfeil, Burger, Kübler, & Fäßler, 2013), which manages signal exchange between multiple subsystem-simulations running in different tools, possibly located on multiple hosts. The FMI model description standard (FMI, 2014) is used to describe the simulations and their signals. A schematic of the principle setup of a co-simulation is shown in Figure 39. The Co-sim Master allows for controlling and monitoring of the co-simulation and provides a graphical user interface (GUI). The Co-Sim Router communicates with all simulations via their connectors and forwards sent messages to specified recipients. The Co-sim Connectors connect simulations to the framework.

The co-simulation framework provides different methods for the integration of simulations (for details, see deliverables D2.3 (Ritter, D2.3 System Architecture Specification (Ver. 1), 2015) and D2.5 (Ritter, D2.5 System architecture specification (Ver. 2), 2015)). So far, OPTi-Sim comprises Matlab and Dymola simulations. Dymola simulations are integrated as functional mock-up units (FMU). The integration of a FMU simulator just requires a ‘.fmu’ file and an initialization file (‘FMU_init.xml’) which needs to be adjusted to the FMU simulator. Running ‘StartFmuConnector.bat’ connects the FMU simulator to the co-simulation framework.

For the integration of Matlab simulators, a python tool has been implemented. As input, two .csv files have to be provided: ‘input.csv’ contains comma separated variables that are inputs to the simulator; ‘output.csv’ contains the output variable names as comma separated values in the first line, and the initial values as comma separated values in the second line. Executing the python tool generates a Matlab simulator ‘simulator.m’, which automatically connects to the co-simulation framework when started in Matlab.

![Figure 39, Schematic of the Co-simulation framework.](image)
Figure 40, Graphical user interface of the TWT Co-simulation Framework. With the buttons marked in red, simulators can be connected in batch mode and synchronized and a co-simulation can be started, paused, stopped and reset.

Once all simulators have been connected, they appear with state ‘INIT’ in the Co-Sim GUI (see Figure 40). Now the simulators can be synchronized and a co-simulation can be started from the GUI.
5. Generation of a network model

This section describes how physical models for the DHC-network are generated and tailored for specific purposes, i.e. use cases as described in WP2 deliverables.

5.1 Generating models of DHC network

Building large scale network models it is not feasible to do manually and therefore an automatic model configuration tool has been developed. Each model that is built is designed for one or several tasks and with the variety of use cases and control tools in the Opti project many different type of models are needed. Therefore a tools has been designed to handle both model configuration from GIS-data (Geographical Information System) and tailoring models depending on the requirements and needs. In the tailoring process another important feature is to simplify networks structurally to assure efficient simulation performance (at least real-time).

In order to simplify user interaction when generating a model for a specific task, a graphical user interface (GUI) is developed. The user can also plot and verify that each model manipulation is realistic by plotting each grid model in the GUI. Data is continuously stored in a database to ensure fast and efficient handling of the relatively large data sets that define a DHC network.

The original data that is stored into the database is extracted from the geographical information system (GIS) at the production company (Luleå energy). It is possible to have several different representations of the GIS data and here we have used three different level depending on size and location.

- Small: 85 consumers (smaller network for one part of Örnäset)
- Medium: 1513 consumers (whole Örnäset area)
- Large: 9533 consumers (whole Luleå)

The original GIS data is stored in the database with .raw as extension and after each optimization step the new data is stored in new tables with the extension .opt. At each moment the user can choose the type of data to plot, the original data or the data after optimization, the GUI is shown in Figure 41.
Figure 41, GUI for generating network models.

Next the main functions of the GUI are explained next.

5.2 GUI MAIN FUNCTIONS

1. **Import data** Reads GIS data (temporarily stored in Excel) and write to data base with extension (.raw).
   Data consists of nodes, pipes, consumers, producers, pumps, sensors, producers and pilot buildings.

2. **Start optimizing** Create files in data base with extension (.opt) and performs data check:
   - Checking for empty nodes.
   - Checking for nodes with only one connection.
   - Checking for consumers without matching nodes.
   - Checking for producers without matching nodes.
   - Checking for nodes with no pipes connected.
   - Checking for pumps without matching nodes.
   - Checking for pipes connected to non-existent nodes.
3. **Save and get data** Save optimized data to database and read the saved data from database.

4. **Drop DB tables** Check database tables and clean them for restart of process.

5. **Optimizing data tools**
   a. **KV_HW** If checked consumers that have a certain type of tap-water valve are pin-pointed and not allowed to be, for example, merged with other building. In this particular case the tap-water valves have been identified as very large.
   
   b. **Pipes serial** Find nodes with only 2 pipes and then concatenate pipes in series.

   ![Figure 42, Before and after applying the pipes serials function.](image)

   c. **Consumer pipe** Move service-line (connection pipe between street and substation) pipes into the consumer.
   
   d. **Consumer node** Once the consumers are moved to the street they are merged with eventual other consumers on the same node. Merging is constrained by for example type of building etc.
   
   e. **Pipes parallel** Check and remove parallel pipes. May occur in raw data (faulty data) and usually necessary after implementing function Merge branch in order to solve eventual circle reference problem.
   
   f. **Clean data** This function removes nodes and pipes with no connection. May appear after some optimization steps.
6. **Merging** There are two types of merging algorithms implemented in the GUI. The first type (called Branch) will try to cluster and move consumers towards a larger pipe diameter and the second type (called consumers) merges neighboring buildings with same consumption type (FV-class).

a. **Branch** In the GUI the user can select a preferred minimum pipe inner diameter (D\_Min) from a set of available pipes in the current data. The merging function will then cluster and move consumers towards this minimum pipe diameter and after a performed merging very few pipes larger than D\_Min exists. All Consumers from a branch are merged into one, and the branch is removed. Taking into account the circular reference to the main network.

b. **Consumers** This function merge consumers that are near each other (neighbors) and that have the same consumption type (FV-code). Other restrictions for merging are also applied (e.g. not merging pilot buildings).

c. **Nodes** This function merge nodes that are close together and thus remove the pipe between if:
   1. The pipes length is < l\_min
   2. The distance between the nodes is < l\_min

Node volumes are calculated based on the pipe diameter and l\_min. When two nodes are merged also volumes are added.
• Nodes that connect to consumers can be merged according to the consumer's nominal flow. But the consumers must have the same code and KV_HW=0.
• Nodes that connect to the producers may take a default volume 5m$^3$.

7. **Plot and delete data:** The user can select whether to plot optimized or raw data. In the generated plot color-coding and text is used to explain an object. It is also possible to select a data cursor and point with the mouse on an object in the plot and get information about the object.

### 5.3 Figure properties

In the GUI it is possible at any point to generate a plot of the system (x and y coordinates are in [m]). In the plot it is possible to zoom in and out and also highlight information for various objects. The main purpose of the plot function is to verify manually raw data and each optimization step.

In the figure, consumers appear as squares with a circle in the middle and pipe that connect to respective node. Different color of squares and sizes means different consumer types (FV-code).

*Figure 45, Color coding in the plot window.*
Figure 46, Different consumer types according to FV number.

Consumer with a yellow node means it is a node with one consumer, red node means that the node has multiple consumer.

Figure 47, highlighting objects by clicking in the GUI gives additional information.

Information of a specific consumer can be seen by selecting the Data Cursor icon and then left click on any consumer. The information that is shown consists of coordinates, no. of consumers, node id that is connected to the consumer, address, power, code type, pilot, and heat data.

To indicate where Pilot buildings and pressure sensors are located text and an arrow is used in the plot. It will also show D if the node is of dynamic type, see Figure 48.
Figure 48, Indication of pilot buildings, dynamic nodes and differential pressure sensors in the plot figure.

Producers and pumps are also shown in the plot where pumps are indicated by a circular object and producers are indicated by a rhomb.

Figure 49, Pumps and producers in the plot.

A yellow small circles represent nodes and the pipe is the line between two nodes. The pipe inner diameter is also reflected in the thickness of the drawn line in order to improve the information to the user of the model configuration tool.
5.4 GUI DYMOLA MODEL GENERATION

After applying simplification algorithms the raw data is transformed and ready to be configured to a model. The first step in this process will introduce some dynamical nodes into the system and the second function will convert the topology in the data base into Modelica code.

Every mathematical model is a simplification of reality. In models, some real-life dynamics are left out and replaced by static relationships. What and how things are left out normally depend on the purpose and usage of the model (here the requirements from WP2 are used). For example, we do not account for compressibility of water, pressure propagation delay or flexibility of pipes in the models we have derived in the OPTi project. The consequence of this approach is that the model will lack some dynamics compared to the real counterpart. In the world of simulation this causes large systems of nonlinear equations for pressure and flow that make the simulation performance deteriorate.

To make it easier for the numerical solver we can add pressure dynamics in well-chosen points in the fluid system. This way we can split the nonlinear systems of equations into smaller parts. Since we only add dynamics of the same magnitude as the real system dynamics we do not lose accuracy – in fact the model will behave more like real fluid systems. Where the most efficient place to introduce pressure dynamics are, to split the equations as much as possible, is a graph theoretical problem that is solved by an external algorithm. The user decides how many dynamic nodes to place out in the system and then presses the Dynamic nodes button to optimally distribute and place the dynamic nodes in to the grid model. Once this is done the user can set a suitable model name and press the Create FMU button at which a Modelica code is automatically generated. This Modelica code is ready for simulation and in the Opti project we export the model into a FMU suitable for co-simulation. The export function is manually and easily done on Dymola.

In the GUI every user interaction is stored into a text file together with a timestamp. This enables a reproduction of the exact same model if necessary and also allows for easy comparison of different applied optimization steps.
5.5 **EXAMPLE OF A MODEL GENERATION**

In the following steps it is demonstrated how a DHC-network model can be generated using the graphical user interface.

Original raw data of Luleå DHC network. In the tool every consumer or node can be investigated by pointing at the object in the graph (as shown for one consumer at Arcusvägen). The raw data includes:

- Consumers: 9533
- Pipes: 2x22376 (supply and return)
- Nodes: 22368

![Original raw GIS-data.](image)

**First step** in the optimization is to concatenate pipes in series, or rather nodes with only two connections are moved. This results in:

- Consumers: 9533
- Pipes: 2x17898 (reduction of 4478 pipes, 20% reduction)
- Nodes: 17890

![Log-file generated from the GUI. Enables a clear tracking on how a specific model has been made.](image)
Second step of the optimization is to move the service pipe that provides water from the street to the consumer substation directly into the consumer substation (additional volume and pressure loss is moved into the substation). This results in:
Consumers: 9533
Pipes: 2\times 9165 (Removed 2 \times 8733 pipes, 48.7932 \% reduction)
Nodes: 9157

Third step is to merge all consumers that are connected to the same node and have the same kind of customer characteristics (FVC-class). This results in:
Consumers: 7977 (1556 consumers removed, 16.3\% reduction)
Pipes: 9165
Nodes: 9157
Fourth step is to remove all parallel pipes. These pipes arise from previous operations or simply from faulty raw-data. Usually also parallel pipes arise from the next steps where merging of consumers is done. In this particular run now reduction was achieved.

Fifth step is to clean data removed eventually lose nodes etc. that occur from faulty data or from upcoming merging strategies. In this run no reduction was achieved.

Sixth step is to start merging consumers. In the first part a minimum pipe diameter (D_min) is selected. This means that the algorithm tries to move all consumers directly to the pipe that has a diameter of D_min or larger. Results for D_min=0.2m:
Consumers: 494 (reduction of 93.8%)
Pipes: 3149 (reduction of 65.6%)
Nodes: 3144 (reduction of 66%)

Seventh step After the branch merging step is completed all the other steps are repeated (serial pipes, consumer pipes and consumer node and cleaning) resulting in:
Consumers: 493
Pipes: 331 (reduction of 89.4%)
Nodes: 326 (reduction of 89.5%)
Eight step consumers that are neighbors and have the same user categorization number and are not a measurement point or a pilot building are merge. Results:
Consumers: 354 (28.2% reduction)
Pipes: 256 (22.7%)
Nodes: 256 (21.5% reduction)

Ninth step a minimum pipe length (Lmin) can be selected and all pipes that are shorter than Lmin will be concatenated if possible. Selecting Lmin to 10m results in:
Consumers: 354 (0% reduction)
Pipes: 234 (8.6% reduction)
Nodes: 234 (8.6% reduction)
6. CONCLUSIONS AND FUTURE WORK

In this section we summarize our conclusions and future work starting with the building modelling work followed by the DHC-network model development.

For the building modeling we used the two different modeling paradigms, black-box and gray-box, to represent the energy consumption of different buildings at different time stamps. Black box modeling involves the use of data analytics (machine learning) principles without any consideration of physics. Gray box modeling, on the other hand, involves a combination of data analytics and physics to represent the building.

For the black box modeling, we used popular machine learning approaches like the standard regression, the support vector regression and the random forest method to model the energy consumption. The random forest methodology gave the best results in terms of prediction accuracy, the accuracy varying between 80-90% depending on the buildings of the different categories.

Our black box modeling approach can also be used to predict the consumption of all the buildings in the network, even if we have the historical data for only a few of those buildings. To model the consumption of all the pilot buildings in the Lulea network, we use the Pnum value of a building. Each pilot building of the network is associated with a Pnum value, which is a quantity proportional to the amount of energy consumed by that building. We train a single model based on the entire data (combining different building data) for a particular category: the consumed energy for a particular time step is scaled by the Pnom value of that building for which we have the corresponding training data point. To predict the consumption for another building and a time stamp, we use the fitted model to determine scaled consumption, which we multiply with the Pnum value of the test building to get the actual consumption. As an ongoing work, we plan to validate how the energy prediction using this scheme matches with the actual observations.

For the gray box modelling, we established component models using physical principles and learnt the parameters in these models using static and time-series regression techniques. We applied the proposed modelling framework on a real-world, test building to demonstrate its application.

During the course of gray box modelling, using the data provided, we found that due to the lack of sufficient transience in the indoor temperature, modelling of the dynamic components (indoor temperature controller and building) were either anomalous or not very accurate. We have provided this feedback to the pilot partners to conduct future controlled experiments with transience in the indoor temperature induced by changing the set-point temperatures. We also identified that the modelling of the substation may be improved by including models of heat losses. The improved models, as a result of the above efforts would be presented in Deliverable 6.2. Furthermore, only one pilot building was used as a demonstration test case for the gray box model. We plan to apply the modelling methodology to the other pilot buildings in both the trial sites (Lulea and Mallorca), and these models would then be integrated in the OPTi co-simulation framework. These details are beyond the scope of this document and will be reported in Deliverable 6.2.

Regarding DHC-network modelling it is concluded that in order to achieve numerically well performing models of large and distributed energy systems a huge amount of manual work is involved. To overcome this problem we early decided to develop and tool that would minimize manual work efforts by automating the model generation process. This is clearly a step in the right direction and a contribution beyond the state-of-
art of developing large and dynamic DHC-network models. It is also concluded that the automatic model
generation feature will enable an efficient life-cycle management of the model since it makes use of network
design data (GIS data) that normally is updated over time.

Using a divide-and-conquer approach is necessary where models are divided into separate processes and
later put together in a co-simulation platform. This approach is not only useful in order to achieve tool
interoperability but, as mentioned, to achieve reasonable simulation performance.

In WP6 the DHC-network model will be integrated with the other modeling components and verification and
validation will be done.

REFERENCES

heating. Svensk standard.
Cortes, P. (2015). D2.1 State of the art, scenarios, requirments and KPI. OPTi Consortium, European
standard.org
Systems Conference (pp. 1-8). IEEE.
Analysis and Recognition, QC., 278-282.
Pattern Analysis and Machine Intelligencs 20(8), 832-844.
All-Electric Aircraft Systems. AIRTEC - Supply on the Wings, Frankfurt/Main.
6. APPENDIXES

In this section we give complementary information.

6.1 MODEL CONFIGURATION GUI FUNCTIONS

**function** import_data(GUI, table_ext, filename)

Read data from excel file and import them to the MySQL tables with extension (.raw) and these tables are node_opt.raw, consumer_opt.raw, pipe_opt.raw, plant_opt.raw

**Parameters**

GUI: if 1 that means run the function from GUI. If 0 that means run the function from matlab editor.

**table_ext**: this is file's extension which will write data in and the default is (raw).

**filename**: the name of an excel file which the function reads data from.

**function** import_io_data(GUI, table_ext, filename)

Read data from Excel file and import them to the MySQL table with extension (.raw) and this table is io.raw.

**Parameters**

GUI: If 1 means run the function from GUI. If 0 means run the function from matlab editor.

**table_ext**: file's extension which will write data in and the default is (raw).

**filename**: the name of an excel file which the function reads data from.

**function** result=check_data(conn, table_ext, clean)

Check for dead-end and solitaire nodes, for consumers without matching nodes, for plants without matching nodes, for pipes connected to non-existent nodes. Returns 1 if there is no error and 0 if there is an error.

**Parameters**

**conn**: name of data base driver.

**table_ext**: this is file's extension that will write data in and the default is (raw).

**clean**: if 1 means remove the id's that have the error from data base , if 0 means not removing ids from data only checking.

**Output**: 1 no error, 0 error

**function** optimize_data_start(GUI, from_ext, to_ext)

To start the optimization process tables in MySQL are copied from the tables with extension (raw) to tables with extension (opt).

**Parameters**

GUI: if 1 means run the function from GUI. If 0 means run the function from matlab editor.

**from_ext**: tables’ extension which will take a copy from, default is (raw).

**to_ext**: tables’ extension which want to write the optimization data in, default is (opt).
function copy_tables_opt(from_ext, to_ext)
Create backup tables with extension (sav) and create new tables in MySQL through using function (copy_table).

Parameters
GUI: if 1 means run the function from GUI. If 0 means run the function from matlab editor.
from_ext: tables’ extension which will take a copy from, default is (opt).
to_ext: tables’ extension which want to save the backup data in, default is (sav).

function Pipe_Dmin_identify
This function reads the values of pipes Dmin and show them in a list in the GUI interface. The user can choose a certain Dmin which it will use in the calculation of branch merging function.

function reduction_str = optimize_data_pipes_seriell(GUI,to_ext)
Concatenating pipes in series.

Parameters
GUI: if 1 means run the function from GUI. If 0 means run the function from matlab editor.
to_ext: tables’ extension which want to write the optimization data in, default is (opt).

Output reduction_str: a string contains the reduction value and percentage of pipes after optimization which will write on the GUI_log file.

function reduction_str= optimize_data_consumer_double(GUI,hw_flag,to_ext)
Connecting consumers in one node if they have the same node.

Parameters
GUI: if 1 means run the function from GUI. If 0 means run the function from matlab editor.
hw_flag = 1 this means buildings with large tap water valves are not allowed to merge.
hw_flag = 0 this means buildings with large tap water valves are allowed to merge.
to_ext: tables’ extension which want to write the optimization data in, default is (opt).

Output reduction_str: a string contains the consumers’ reduction value and percentage of after optimization, which will write on the GUI_log file.

function reduction_str = optimize_data_consumer_pipe (GUI,to_ext)
Moving connecting pipes into the consumers.

Parameters
GUI: if 1 means run the function from GUI. If 0 means run the function from matlab editor.
to_ext: tables’ extension which want to write the optimization data in, default is (opt).
Output reduction_str: a string contains the reduction value and percentage of pipes that connected to the consumers after optimization, which will write on the GUI_log file.

```matlab
function reduction_str = optimize_data_pipes_parallell(GUI,to_ext)
    It is optimizing nodes with parallel two pipes.

Parameters
GUI: if 1 means run the function from GUI. If 0 means run the function from matlab editor.
to_ext: tables’ extension which want to write the optimization data in, default is (opt).
```

Output reduction_str: a string contains the reduction value and percentage of pipes after optimization, which will write on the GUI_log file.

```matlab
function reduction_str = optimize_data_clean (GUI,to_ext)
    Remove nodes and pipes with no connection.

Parameters
GUI: if 1 means run the function from GUI. If 0 means run the function from matlab editor.
to_ext: tables’ extension which want to write the optimization data in, default is (opt).
```

Output reduction_str: a string contains the reduction values and percentage of pipes, nodes and consumers after optimization, which will write on the GUI_log file.

```matlab
function reduction_str = optimize_data_merge_branch(GUI,hw_flag,Dmin_def,to_ext)
    It is merging consumers depending on the value of pipes’ Dmin.

Parameters
GUI: if 1 means run the function from GUI. If 0 means run the function from matlab editor.
hw_flag = 1 this means buildings with large tap water valves are not allowed to merge.
hw_flag = 0 this means buildings with large tap water valves are allowed to merge.
Dmin_def: a certain pipes Dmin.
to_ext: tables’ extension which want to write the optimization data in, default is (opt).
```

Output reduction_str: a string contains the reduction values and percentage of pipes, nodes and consumers after merging, which will write on the GUI_log file.

```matlab
function reduction_str = Optimize_data_merge_FVC(GUI,hw_flag,to_ext)
    It is merging consumers depending on the value of consumer’s code.

Parameters
GUI: if 1 means run the function from GUI. If 0 means run the function from matlab editor.
hw_flag = 1 this means buildings with large tap water valves are not allowed to merge.
hw_flag = 0 this means buildings with large tap water valves are allowed to merge.
to_ext: tables’ extension which want to write the optimization data in, default is (opt).
```
Output reduction_str: a string contains the reduction values and percentage of pipes, nodes and consumers after merging, which should be written on the GUI_log file.

```
function reduction_str = optimize_data_merge_nodes(GUI, hw_flag, lmin_def, to_ext)
```

Merging nodes volumes that are close together and remove the pipe between them.

**Parameters**
- **GUI**: if 1 means run the function from GUI. If 0 means run the function from matlab editor.
- **hw_flag = 1**: this means buildings with large tap water valves are not allowed to merge.
- **hw_flag = 0**: this means buildings with large tap water valves are allowed to merge.
- **lmin_def**: a certain lmin can be selected from list.
- **to_ext**: tables’ extension which want to write the optimization data in, default is (opt).

Output reduction_str: a string contains the reduction values and percentage of pipes, nodes and consumers after merging, which will write on the GUI_log file.

```
function result = Run_python(node_no, GUI, pyt_file, result_file)
```

Python program which analyzes the position of dynamic nodes.

**Parameters**
- **node_no**: default number is 5 but it can be (1, 5, and 10) from list.
- **GUI**: if 1 means run the function from GUI. If 0 means run the function from matlab editor.
- **pyt_file**: the name of file that will be run in python program.
- **result_file**: this file contains the nodes with dynamic values = 1.

Output result 1 = command error, result 0 = no error.

```
function path_mo = create_model_FMU(model_name, table_ext)
```

Take data from DB (opt tables) and then creates Modelica model.

**Parameters**
- **model_name**: the name of the model with extension (.mo). Default name is OpTiNet_test.
- **table_ext**: opt tables.

Output path_mo = the path of the (model_name.mo).

**Opti_GUI_Plot**

Plot data from (.raw tables) or (.opt tables) in the figure. These data are pipes, nodes, consumers, pumps, producers with the some properties which are so necessary to show.

```
function gui_show_log(GUI, file_name, event, parameter, data)
```

**Parameters**
- **GUI**: if 1 means run the function from GUI. If 0 means run the function from matlab editor.
file_name: name of the file that will use to register every step in the GUI. file's name is the date of running the GUI ex: (GUI_log16-Aug-2016.txt) and the path of the file is:
D:\SVN\OpTi\Projektmapp\kod\Matlab\GUI\GUI_log this file can open in notepad program.

event: each step in GUI has a specific number which will used to select which have to write in the file.

Parameter: values should be written in the file ex: lmin, Dmin.

data: it is a structure:

data_typ: small, medium, large

data_con: consumer no.

data_node: nodes number

data_pip: pipes no.

data_str: reduction value for each optimization or merging Event, parameters and data are the information will be written in the file.

---

function val = checksql_DB(to_ext)

This function is used to check if tables are created or not in SQL.

Parameters

table_ext: extention of the table ex. That needs to check (.raw, .opt, .mbd, .svt).

Output val: 1 means tables are not created yet. 0 means tables are created.

function txt = myupdatefcn(~,event_obj,DCM_Data,DCM_Node)

Show information through Data cursor for each consumer and node.

Parameters

~: currently not used (empty).

event_obj: Object containing event data structure to get the position (x, y) of the object.

DCM_Data: structure containing consumer data.

DCM_Node: structure containing node data.

Output txt: the text that shoing in the data cursor.

function dropall_tables(table_ext)
This function is used to drop tables from MYSQL.

**Parameters**


**function** `button_status(value)`

This function is used to change the status of buttons enable or disable in GUI. It helps to disable all buttons when the user press on a certain button to do one process at the time and when the process finish, all buttons become enable again.

**Parameters**

- `value`: 1 enable, 0 disable.

http://www.ltu.se/cms_fs/1.101555!/file/LivingLabsMethodologyBook_web.pdf
https://flic.io/