Seminar Thesis

Analysis of non-linear estimation algorithms for parameterizing data driven grey-box models

CES Seminar

Aachen, Januar 2017

Thomas Rosen
matriculation number: 292791

Supervisors:
Hassan Harb, M. Sc.
Univ.-Prof. Dr.-Ing. Dirk Müller

This thesis was submitted to:
E.ON Energy Research Center | ERC
Institute for Energy Efficient Buildings and Indoor Climate | EBC
Mathieustraße 10, 52074 Aachen
Abstract

The rapid development and decreasing prices of smart meters has lead to the production of huge amount of data, often called Big Data in this context. The challenge nowadays is to handle and organize this data and transform it into informations. In context of energy management optimization, more data is available due to building automation systems, comprising for solar irradiation and temperature and thermal energy consumption. For characterizing buildings thermal properties, this data can be used to estimate parameters of a grey box model. This model is used in a predictive control e.g. to optimize the energy consumption, the thermal comfort or even to use buildings as a short-term thermal storage which provides flexibility for load shifting. Several challenges exist in parameters’ estimation step of the grey box models. Even the simplest structure comprises non-linear relations between the decision variables. On the one hand this results in a harder to solve optimization problem, on the other hand this highly increases the uncertainty region and therefore most likely the size of error. Additionally, the estimation problem is strongly underdetermined. Because the only inputs in addition to the measured data are total area, area-ratios and radiation-ratios, the grey box model is strongly dependant of the quality and representative distribution of this data. This can lead to problems when the model parameters are estimated within one season and it should forecast some thermal behaviour in another season with totally different temperatures. This work focuses on implementing a tool for analysing grey box models behaviours in the programming language python. The tool allows for modifying all inputs, estimating the parameters, saving, loading and comparing beforehand computed results. In addition the tool enables the user to choose between two optimization solvers for the non-linear parameter estimation: Distributed Evolutionary Algorithm for Python (DEAP) and Spatial Branch and Bound (SBnB). A preliminary evaluation shows that: e.g. the SBnB allows for a better parameters estimation e.g. the impact of the bounds blabla
Contents

Glossary iii
List of Figures v
1 Motivation 1
2 Methodology 2
  2.1 Building modeling ....................................... 2
  2.2 Model Identification ..................................... 4
    2.2.1 General Approach .................................. 4
    2.2.2 Algorithms .......................................... 5
    2.2.3 Implementation ...................................... 7
3 Results 8
4 Conclusion 15
Bibliography 16
A Introduction to Python 18
B User documentation 19
# Glossary

## Symbol and Unit

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>area</td>
<td>$m^2$</td>
</tr>
<tr>
<td>$C$</td>
<td>heat capacity</td>
<td>$W/kg$</td>
</tr>
<tr>
<td>$\dot{Q}$</td>
<td>heat flow</td>
<td>$W$</td>
</tr>
<tr>
<td>$Q$</td>
<td>heat</td>
<td>$Ws$</td>
</tr>
<tr>
<td>$T$</td>
<td>temperature</td>
<td>$K$</td>
</tr>
<tr>
<td>$t$</td>
<td>time</td>
<td>$s$</td>
</tr>
<tr>
<td>$R$</td>
<td>heat resistance</td>
<td>$m^2K/W$</td>
</tr>
<tr>
<td>$K$</td>
<td>heat transfer koefficient</td>
<td>$W/m^2K$</td>
</tr>
</tbody>
</table>

## Indices and Abbreviations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>ambience</td>
</tr>
<tr>
<td>$in$</td>
<td>interior</td>
</tr>
<tr>
<td>$e$</td>
<td>exterior</td>
</tr>
<tr>
<td>$eq$</td>
<td>equilibrium</td>
</tr>
<tr>
<td>$ia$</td>
<td>inner air</td>
</tr>
<tr>
<td>$h$</td>
<td>spatial heating</td>
</tr>
<tr>
<td>$sol$</td>
<td>solar</td>
</tr>
<tr>
<td>$rad$</td>
<td>radiation</td>
</tr>
<tr>
<td>$abs$</td>
<td>absorption</td>
</tr>
<tr>
<td>$conv$</td>
<td>convection</td>
</tr>
<tr>
<td>$int$</td>
<td>internal</td>
</tr>
</tbody>
</table>

continued on next page
Indices and Abbreviations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ext</td>
<td>external</td>
</tr>
</tbody>
</table>
List of Figures

2.1 Grey Box Models [Henrik Madsen, Peder Bacher [2014]] ............................................. 2
2.2 1R1C ................................................................................................................................. 3
2.3 4R2C [Harb u. a. [2016]] ............................................................................................... 3
2.4 Parameter estimation structure ....................................................................................... 4
2.5 Spatial Branch and Bound [Ellen Zhuang [2015]] ............................................................ 6

3.1 1R1C DEAP $n_{gen} = 150$ $n_{pop} = 50$. ................................................................. 9
3.2 4R2C DEAP $n_{gen} = 150$ $n_{pop} = 20$. ................................................................. 9
3.3 4R2C DEAP $n_{gen} = 150$ $n_{pop} = 50$. ................................................................. 10
3.4 4R2C DEAP $n_{gen} = 150$ $n_{pop} = 50$ internal Heat .................................................. 10
3.5 4R2C DEAP $n_{gen} = 150$ $n_{pop} = 50$ modified ............................................................ 12
3.6 4R2C DEAP $n_{gen} = 80$ $n_{pop} = 20$. ................................................................. 12
3.7 1R1C SBnB maxIter=100 no adjustment ................................................................. 13
3.8 1R1C Least Square Curve Fit no adjustment ............................................................... 14
3.9 4R2C Least Square Curve Fit no adjustment ............................................................... 14

B.1 Startscreen ................................................................................................................. 20
B.2 Building Data Screen ................................................................................................. 20
B.3 Simulation Screen ..................................................................................................... 21
B.4 Parameter Estimation Screen ..................................................................................... 21
B.5 Validation Screen ...................................................................................................... 22
1 Motivation

The rapid development and decreasing prices of sensors has lead to the production of huge amount of data, often called Big Data in this context. The challenge nowadays is to handle and organize this data and transform it into informations. For handling, the IT sector drastically improved the use of databases, which today can be found everywhere. In context of energy management optimization more data is available by use of smart meters, databases for solar irradiation and temperature, humanity and many more sensors. For characterizing a buildings thermal properties this data can be used to estimate parameters of a grey box model. This model is used in a predictive control e.g. to optimize the energy consumption, to always stay in some user-defined comfort termpature region or even to use buildings as a thermal storage to counter the challenges posed by the rapid increase of regenerative energy in Germany. But there are severals problems when it comes to estimating parameters of this grey box models. Even the simplest structure is non-linear in parameters. On the one hand this results in a harder to solve optimization problem, on the other hand this highly increases the uncertainty region and therefore most likely the size of error. In addtion the estimation problem is strongly underdetermined. Because the only inputs in addition to the measured data are total area, area-ratios and radiation-ratios, the grey box model is strongly dependant of the quality and representative distribution of this data. This can lead to problems when the model parameters are estimated within one season and it should forecast some thermal behaviour in an other season with totally different temperatures. Previous work has analysed this approach for noisy input data (see Kristensen u. a. [2004].). They are using an extended kalman filter to approximate the covariances and use a maximum likelihood estimator combined with some quasi newton scheme of a tool called CTSM-R (see Niels Rode Kristensen, Henrik Madsen [Dece]) to estimate the parameters. During this work a similar tool for analysing grey box models behaviours in the programming language python (see AppendixA) is developed. The tool is capable of comfortably controlling all inputs, estimating the parameters, saving, loading and comparing beforehand computed results. In addition the tool enables the user to choose between two optimization solvers for the non-linear parameter estimation: Distributed Evolutionary Algorithm for Python (DEAP) and Spatial Branch and Bound (SBnB). This will enable me to analyse the results of the parameter estimation in several ways. In means of error, impact of different simulation parameters, physicality and advantages and disadvantages of solvers.
2 Methodology

2.1 Building modeling

To understand what grey box models are one has to understand two extrema: On the one hand there are blackbox models. Blackbox models are models where one can't see or it doesn't make sense to look in the inner structure from an engineering point of view. The inner structure concentrates on finding pattern or mean values with variances. So here is one disadvantage of blackbox models: Their inner parameters don't have a physical meaning. But one advantage at the same time is these parameters can be calculated fast. Mostly blackbox models are used for data analysis. The most famous modelling approaches for black box modelling are machine learning and regression method. With the famous candidates artificial neural networks (ANN) and auto-regression moving average (ARMA). On the other hand there are whitebox models. White box models are models where many values calculated inside have a physical meaning and can be used to interpret the behaviour. But the drawback of this high physicality is it takes long to calculate or measure all necessary parameters of a model. In between there are grey box models. These models combine some of the physicality of whitebox models with the advantage of reducing the amount of necessary parameters. Most of the strongly simplyfied models are grey box models. In addition grey-box models can be more data driven or design driven (see figure 2.1). In case of this thesis we take a look at data driven grey box models. The model should calculate an average indoor air temperature of a building depending on data like solar irradiation or space heating. Therefore the model structure is reduced to a thermal network of capacities and resistances. The capacities reflects the mass which can "store" the heat inside a model and includes the walls. The resistances represent average heat transfer coefficients between points in the networks. These points can have a mass, so be connected to a capacity or can be massless (see figure 2.3).

![Grey Box Models](figure2.1: Grey Box Models [Henrik Madsen, Peder Bacher [2014]])
2.1 Building modeling

Figure 2.2: 1R1C

Figure 2.2 shows the simplest possible building grey box model structure. All mass of the building is modeled via one capacity ($C_{in}$). The indoor temperature node ($T_{in}$) is connected to the ambience temperature node ($T_a$) with the heat resistance ($K_{in,a}$) which is given by measurements or forecast data.

Figure 2.3: 4R2C [Harb u. a. [2016]]

A more complex structure is shown in figure 2.3. Like the 2R2C structure we have a separation between inner and outer masses (outer walls) with the corresponding capacities $C_{in}$ and $C_e$. In addition
2 Methodology

we have modeled the air inside as a massless point in our network. It models the exchange between inner and outer mass temperatures and has an exchange (through e.g. windows) to the ambience temperature node \((T_a)\). The outer mass temperature \((T_e)\) is in exchange with an equilibrium ambience temperature. In the calculation of this temperature the ambience temperature is corrected by an absorption parameter of the wall times the solar irradiation.

2.2 Model Identification

2.2.1 General Approach

To compute the parameters of the grey box model several methods are used. The general method is to calculate an error of the model depending on input data \(u\) and parameters \(p\) to some measured data \(y\) (see figure 2.4). The tool provides three optimizers: DEAP, SBnB and a basic scipy least-squares fitting. DEAP and SBnB optimize a scalar while the least square fitting combines error calculation and optimization. Therefore the optimizer has “more knowledge” but is less flexible (e.g. to different error types).

![Figure 2.4: Parameter estimation structure](image)
2.2 Model Identification

2.2.2 Algorithms

**Distributed Evolutionary Algorithm for Python**

DEAP is an evolutionary computation framework. It has been used to implement a simple algorithm to compute the parameters of the non-linear optimization problem. An evolutionary algorithm can be split in five steps:

**Algorithm 1 DEAP Algorithm**
1: Initialize a population of \( n \) particles often called individuals on the feasible set
2: while \( \text{while } < \text{ maxItera}tions \text{ do} \)
3: Evaluate the objective function for changed/new particles, calculate the error and save it as fitness
4: Move/Perturb some particles
5: Select close particles and remove the ones with lower fitness
6: Respawn randomly as many particles as removed from the set

So this method just finds a good result when iterating long enough. But it’s not assured to find the global optimum.

**Spatial Branch and Bound**

The Spatial Branch and Bound algorithm is an optimization algorithm for non-linear (even mixed integer) problems. The main idea is to replace the non-linear terms by auxiliary variables to compute a convex relaxation and a local solution in each step. To reach convergence we split the feasible set in a way which minimizes the gap between these two. The algorithm converges when the gap for all global optima candidates is smaller than some \( \epsilon \)-criterion.

The implementation is mainly oriented to Smith u. Pantelides [1999]. First we translate our objective function into a standard form. In this standard form all non-linearities in the optimization argument, here parameters characterizing the grey box model, are replaced by a new auxiliary variables with a non-linear equality constraint separated in an extra function. When initialising the solver the feasible set is stored as a root node of a binary tree. Now the algorithm (see algorithm 2) can be started. First we tighten the bounds. This step ensures the bounds of our auxiliary variables are set depending on the equality constraint. With a local solver we can find a local solution of the exact problem (using the equality constraints) and set this as an upper bound. Now the standard form is used: The objective function (without theses equality constraints) is linear in its arguments. Because we calculate the least squares error the problem becomes quadratic. We can compute convex underestimators for each non-linear term to end up with a convex QP which we solve. The solution is set as a lower bound. Now we can calculate an approximation error of this convex relaxation towards the non-linear auxiliary variables. In case of the grey-box models these are linear fractional
2 Methodology

![Spatial Branch and Bound Diagram][1]

**Figure 2.5:** Spatial Branch and Bound [Ellen Zhuang [2015]]

---

**Algorithm 2** SBnB Algorithm

1: Save the feasible set as root in a binary tree
2: Tighten bounds
3: Local solution → ub[x] → paramList[x]
4: Convex solution → lb[x] → nonLinApproxErr[x]
5: gap[x]=lb[x]-ub[x]
6: for i<maxIter; i++ do
7:   for leaf y with biggest gap do
8:     Split feasible set at variable with biggest nonLinApproxErr and $w_p$
9:     remove y from lists
10:    for all leaves x of subtree y do
11:      Tighten bounds
12:      Local solution → ub[x] → paramList[x]
13:      Convex solution → lb[x] → nonLinApproxErr[x]
14:      gap[x]=lb[x]-ub[x]
15:      Terminate x when gap[x]<\epsilon
16:    for all leaves x do
17:      Discard x when lb[x]>min(ub[j])
18:    if all x discarded or terminated then
19:      break → Algorithm terminated

---

[1]: Image of the diagram.
2.2 Model Identification

terms:
\[ AE_{ijk}^{lf} = \left| w_k - \frac{w_i}{w_j} \right| \]  

(2.1)

We want to split on the variable \( w_i \) or \( w_j \) for which we have the biggest approximation error \( AE_{ijk}^{lf} \). To decide which variable and where to split it we use the calculated parameters \( w_i \) and \( w_j \) of the local solution.

\[ w_p = \arg\min \{0.5 - \frac{w_i - w_{i_{lb}}}{w_{i_{ub}} - w_{i_{lb}}} \cdot 0.5 - \frac{w_j - w_{j_{lb}}}{w_{j_{ub}} - w_{j_{lb}}}\} \]  

(2.2)

Now we can enter the main loop: We find the leaf node in the tree with the biggest gap between lower and upper bound. Also we check for termination by comparing the gap with an \( \epsilon \) which is handled initially to the solver. We split or feasible set at \( w_p \). Therefore we create two new subregions which we store in our tree. We repeat the steps above (tighten the bounds, calculating the local solution and convex solution) for each leaf of the "sub-tree". Now we can discard all leaves where their lower bound is bigger than the smallest upper bound (see 2.5). To not iterate through the tree to reach the leaves in each iteration we store the necessary values to compare them in lists. If all branches are discarded or terminated the algorithm terminates. Else the algorithm runs to a predefined number of maximum iterations.

2.2.3 Implementation

The tool on the top level is split in a typical model-view-controller architecture. This is to separate the computation (model) from handling of the visualizations (view) and a data/communication class in between (controller). The model is split into the grey-box-model class in which the several grey box models are inherited from an abstract base class. To reduce redundant code and reflect the relation between these models the classes form an inheritance tree building a decorator like design pattern [Philipp Hauer]. The model needs a controller object to receive the input data. Then it is registered into a solver object which works as a wrapper for connecting with the interfaces of both optimization algorithms. These are just used to estimate the parameters of the grey box model. In appendix B is a short documentation on how to install, use the tool and how the GUI is structured.
3 Results

The evaluation of the written and introduced tool is done on the basis of a measured data set from a huge building with 37 apartments called Roth Jan Mar. The data is sampled with a timestep of 60 minutes over 40 days. For this evaluation I used data of 365 days with a timestep of one minute. If not mentioned differently least square error is used for parameter estimation and for error analysis which is defined as:

\[
LSQ = \frac{\sum_{i=0}^{N}(T_{ia, measured} - T_{ia})^2}{N}
\]  

(3.1)

Table 3.1 show the used parameter bounds for parameter estimation. Table 3.2 the result parameters associated to the figures. Figure 3.2 shows one problem using this error and no adjustment. In the reason of high data depency and only the error is taken into account problems like totally over or undershooting can be the result. While the parameter estimation error (on the firest half) is reduced to 0.036. The total error is 0.735 so during the forecast period we produce a 20x as big error as in the parameter estimation period. A lot of physicality is lost due to the strict optimization on too less information. The parameters for the inner capacity \(C_{in}\) and heat transfer coefficient to the air \(K_{in, ia}\) are twice as high than the corresponding parameters of the exterior (\(C_{e}\) and \(K_{ia, e}\)). This is in the reason of the measured indoor temperature is rising in the first half. In addition there's is a way to small response behaviour on heat inputs in the data, because the algorithm is based on least squares it focuses on a mean value. The problem can be solved by adjusting longer and over more reprentative data (see figure 3.3). Here the capacities and heat coefficients balance more Best results with no adjustment are achieved in figure 3.4. Here simulated with a higher population which leads to much higher optimization times. The is in the reason of many function evalution because of the tournament/selection phase. In addition I used the internal heat profile. The parameters are balanced (see before) and not hitting their bounds. The total error of 0.046 is quiet good. When using a simplest builing grey box model structure (1R1C see figure 3.1) the total error can be even reduced to 0.02. The parameter estimation uses accept the model structure the same settings. It runs a lot fast because the objective function now is much simpler. Because the model is so simple it can't follow complex dynamic behaviour and is even with a much beter total error worse than the more complex one. The parameters are set to the given bounds in a way the model can represent the rise of the indoor temperature in the first half.

In a next step (see figure 3.5) the calculated parameters (see table 3.2) of figure 3.4 are used to check some physicality on a rough level. For this the space heating is summed up to a portion stored in the
3 Results

Figure 3.1: 1R1C DEAP $n_{gen} = 150$ $n_{pop} = 50$

Figure 3.2: 4R2C DEAP $n_{gen} = 150$ $n_{pop} = 20$
Figure 3.3: 4R2C DEAP $n_{gen} = 150 \ n_{pop} = 50$

Figure 3.4: 4R2C DEAP $n_{gen} = 150 \ n_{pop} = 50$ internal Heat
Table 3.1: 4R2C Parameter Bounds

<table>
<thead>
<tr>
<th>Parametername</th>
<th>$C_{in}$</th>
<th>$K_{in,ia}$</th>
<th>$H_i$</th>
<th>$C_e$</th>
<th>$K_{e,a}$</th>
<th>$K_{ia,e}$</th>
<th>$K_{ia,a}$</th>
<th>$T_{e,init}$</th>
<th>$T_{in,init}$</th>
<th>$H_{int}$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ub</strong></td>
<td>500</td>
<td>100</td>
<td>0.5</td>
<td>600</td>
<td>1.5</td>
<td>25</td>
<td>1.5</td>
<td>18</td>
<td>26.3832</td>
<td>2</td>
</tr>
<tr>
<td><strong>lb</strong></td>
<td>30</td>
<td>0.5</td>
<td>0.01</td>
<td>20</td>
<td>0.05</td>
<td>0.5</td>
<td>0.15</td>
<td>10</td>
<td>19.3832</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3.2: Result Parameters

<table>
<thead>
<tr>
<th>Figure</th>
<th>$C_{in}$</th>
<th>$K_{in,ia}$</th>
<th>$H_i$</th>
<th>$C_e$</th>
<th>$K_{e,a}$</th>
<th>$K_{ia,e}$</th>
<th>$K_{ia,a}$</th>
<th>$T_{e,init}$</th>
<th>$T_{in,init}$</th>
<th>$H_{int}$</th>
<th>$K_{in,a}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2</td>
<td>498.94</td>
<td>44.36</td>
<td>0.11</td>
<td>238.09</td>
<td>0.11</td>
<td>21.17</td>
<td>0.46</td>
<td>16.60</td>
<td>23.67</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3.3</td>
<td>432.82</td>
<td>86.24</td>
<td>0.01</td>
<td>128.12</td>
<td>0.05</td>
<td>5.35</td>
<td>0.42</td>
<td>17.76</td>
<td>22.17</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3.4</td>
<td>343.86</td>
<td>9.63</td>
<td>0.01</td>
<td>304.10</td>
<td>0.05</td>
<td>11.31</td>
<td>0.82</td>
<td>25.39</td>
<td>25.18</td>
<td>1.16</td>
<td>0</td>
</tr>
<tr>
<td>3.6</td>
<td>267.28</td>
<td>30.91</td>
<td>0.01</td>
<td>231.58</td>
<td>0.2</td>
<td>25.0</td>
<td>0.34</td>
<td>17.73</td>
<td>24.25</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3.9</td>
<td>200.42</td>
<td>39.27</td>
<td>0.05</td>
<td>599.96</td>
<td>0.18</td>
<td>1.10</td>
<td>0.36</td>
<td>18.0</td>
<td>21.52</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3.1</td>
<td>500.0</td>
<td></td>
<td>0.024</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3.7</td>
<td>265.03</td>
<td>0.037</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0.55</td>
</tr>
<tr>
<td>3.8</td>
<td>363.37</td>
<td>0.034</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0.53</td>
</tr>
</tbody>
</table>

The indoor air temperature predicted by the model should change at this point drastically fall down when no heating applied spike high when its applied and then fall back to the same values. This test should not be done with adjustment setting active because if the adjustment is done in the region of the spike this totally changes the expected outcome.

The other options to solve the problem of unphysical parameters is by turning the adjustment on (see figure 3.6). By this the optimization algorithm can focus on the variance and not getting the mean right because it’s periodic set to the measured value. The error here becomes small because it’s strongly reduced by the adjustment. When adjusting to long can even be a disadvantage because often this leads to solutions on the border.

When computing 1R1C with Spatial Branch and Bound (see figure 3.7) it leads to the known problem of ill-posedness when $\dim(x) \ll \dim(y)$, where $x$ represents the vector of parameters and $y$ the vector of fitting data. Therefore the relaxed least square problem had infinitely many solutions and the algorithm tends to stuck in a local solution. The problem is that this local solution can now be even or bigger than the local solution of the exact problem. But because the algorithm needs a convex underestimator, I applied some Tikhonov regularization to the convex lower bound. The disadvantage is that the gap between lower and upper bound now can not shrink under $\epsilon$. Therefore the algorithm terminates not any more, but runs until the maximum number of iterations.

Figure 3.8 shows the least square fitting of a 1R1C. The total error of 0.015 is the smallest error and therefore the best fitting according to the error. But using adjustment with this method is more a disadvantage than an advantage, because this setting disturbs the gradient calculation. While the
3 Results

Figure 3.5: 4R2C DEAP $n_{gen} = 150$ $n_{pop} = 50$ modified

Figure 3.6: 4R2C DEAP $n_{gen} = 80$ $n_{pop} = 20$
total error of 0.059 for a non-adjustment fitting for 4R2C (see figure 3.9) is good, the physicality is not as good as with DEAP. The temperature overshoots right after the training time and the dynamic of the model is quiet small. This does neither change with internalHeat nor with adjustment.
3 Results

Figure 3.8: 1R1C Least Square Curve Fit no adjustment

Figure 3.9: 4R2C Least Square Curve Fit no adjustment
During this work a software tool in python has been developed to analyze the behaviour of building grey box models outcome depending on their parameters. There are lots of possible configurations of this tool so optimal solutions can be found for most kind of scenarios these models can be used in. In case of fast and accurate optimization the scipy least square minimizer is the best of the tested algorithms. It enables a solid forecast in the mean-sense. But if the adjustment setting or complex grey box models should be used to focus more on small time intervals and high dynamics DEAP turned out to be a good solver. Even when most work has been invested in the spatial branch and bound algorithm, it is the slowest of the used algorithms. Ensuring to find the global solution of the problem was the key idea behind it. But in the matter of ill-posedness and some regularizations due to this, it can not be proven (see chapter cha:results). To enhance it, one should investigate in parallelization, better regularization and more efficient sub-optimizations routines. There are many important points to notice especially the high depency on the quality of data provided. To short data sets lead to unphysical parameters. The effect is strongly reduced by using the adjustment setting. A general solution to always find good parameters independent of the data provided has not been found yet. It may be a good idea to investigate more in data analysis methods which can be applied prior to the parameter estimation and calculate a good training phase and a "good"/stable timestep.
Bibliography


[Henrik Madsen, Peder Bacher 2014] HENRIK MADSEN, PEDER BACHER: Grey-Box Modeling; An approach to combined physical and statistical model building. (2014)


[Philipp Hauer ] PHILIPP HAVER ; PHILIPP HAVER (Hrsg.): Das Decorator Design Pattern. https://www.philipphauer.de/study/se/design-pattern/decorator.php

A Introduction to Python

In the reason of the fast development of new and cheaper infrastructures and hardware in the IT sector it is a challenge of today to produce software for this infrastructure fast and highly flexible. Python faces this challenge by being very short, highly dynamic and fully object-oriented. In python objects are bound to a name. This enables the programmer to switch between mutable and immutable object types (like tuples and lists) and switch fast between call-by-value and call-by-reference. This method has been used and is essential to understand the code written for Spatial Branch and Bound. So even a float is an object in python with several functions. On the one hand this overhead and the interpreting during the runtime makes python slower. But on the other hand a huge flexibility is the result. The reason why python should be used is the huge amount of well-documented, open source packages which can be downloaded and easy installed using the python package manager. One of these packages is PyQt, an API to the QT library enabling to easily design a GUI with the QT Designer and later bind it to the written python code. Another package used for the implementation is DEAP (Distributed Evolutionary Algorithm for Python).
First thing you need is a installation of python. The tool is python 2 and python 3 compatible so the decision is on you. If you want to use some files in another project you should make your decision on the python version of packages which you need for this project. There are several installer package for python out there. Most used are:

WinPython: A portable python installation with already installed some core packages like numpy, matplotlib and pyqt. This installation includes the IDE Spyder.

Pycharm: A more complex IDE with more features (but also more overhead). It’s using the anaconda package manager which is quiet useful for installing more complex package.

After installation of one of these, you just need three more packages: pandas,deap and scipy. All can be installed using pip but depending on your installation there are different channels. Ask google (or the search engine oof your choice) to find the actual channel for your python version. Then download the tool from: https://git.rwth-aachen.de/thomas.rosen/GreyBoxModelTool/tree/master If installed open a new project and place the root directory in such way it can find the init.py file. Now go to the src folder and run the main_app.py file. A QT window should pop up looking like:

You may notice that just the Building Data Button is enabled. This leads to: In this screen one should select input files. Example input files are provided with the package. The first dialog should open a file with building parameters. The example file contain the word param. Next up is loading the measurement data. Example files for this contain the word measurement. In addition one can load a Heat_intern profile which reflects the heat production by other sources like human bodies on an average delay bases. These files should show the heat produced withing all hours of day and can be multiplied by the apartment number or the area. You can use the Show Buttons to check if your data has been uploaded correctly. In a matter of time the GUI doesn’t feature an exception class. So you won’t receive an arrow when you upload something wrong. But the inputs are checked for consistency and so other buttons will stay disabled when the necessary input is not there. In the figure B.3 one can adjust the basic simulation setup. Which grey-box-model structure shoul be used, a timestep and the optimization algorith here called solver. In addition one can adjust some settings for this solver.

The B.4 gives access to the main part of this thesis. First you need to upload the bounds. Example files for this contain the words param_bounds. The tool provides an adjustment setting which can
be used for parameter estimation and forecasting. The setting files and the training time are loaded within the building parameter file and can be adjusted here. For estimating this setting can be used to concentrate more on the dynamic behaviour than getting a good mean value. But this setting will reduce and therefore fake the error of the parameters fitting. For forecasting this can setting makes most sense when the parameters were calculated with this setting too. So the behaviour does not become weird. This simulates that the model is set to a measured temperature each adjustment period. The training time is the time for which the parameter estimation is calculated. For evaluation it makes sense to split the time interval of your data into a training time and a forecast interval. When you click the Start Parameter Estimation Button it is started. Progress is shown just in DEAP.
Figure B.3: Simulation Screen

Figure B.4: Parameter Estimation Screen
via the console. I didn't find the time and a good method to connect the progress bar.

For evaluation it is necessary to run a forecast with the now or prior calculated parameters. This can be done in the Run Forecast section.

Finally we got the B.5 section. Here you can use prior forecasted data and the forecasted indoor temperature compared to the one stored in the measurement file (which was used for fitting). In addition one can load an other forecast file and plot is as a comparison to the one produced in the Run Forecast section.