Toolbox for development and validation of grey-box building models for forecasting and control

De Coninck, Roel; Magnusson, Fredrik; Åkesson, Johan; Helsen, Lieve

Published in:
Journal of Building Performance Simulation, Taylor & Francis

DOI:
10.1080/19401493.2015.1046933

2016

Document Version:
Peer reviewed version (aka post-print)

Toolbox for development and validation of grey-box building models for forecasting and control

Roel De Coninck[a,b,c], Fredrik Magnusson[d], Johan Åkesson[e], Lieve Helsen[b,c]
[a]3E nv, 1000 Brussels, Belgium,
[b]KU Leuven, Department of Mechanical Engineering, 3001 Heverlee, Belgium,
[c]EnergyVille, 3600 Waterschei, Belgium,
[d]Department of Automatic Control, Lund University, SE-221 00 Lund, Sweden,
[e]Modelon AB, Ideon Science Park, SE-223 70 Lund, Sweden

Abstract

As automatic sensing and Information and Communication Technology (ICT) get cheaper, building monitoring data becomes easier to obtain. The availability of data leads to new opportunities in the context of energy efficiency in buildings.

This paper describes the development and validation of a data-driven grey-box modelling toolbox for buildings. The Python toolbox is based on a Modelica library with thermal building and Heating, Ventilation and Air-Conditioning (HVAC) models and the optimisation framework in JModelica.org. The toolchain facilitates and automates the different steps in the system identification procedure, like data handling, model selection, parameter estimation and validation.

To validate the methodology, different grey-box models are identified for a single-family dwelling with detailed monitoring data from two experiments. Validated models for forecasting and control can be identified. However, in one experiment the model performance is reduced, likely due to a poor information content in the identification dataset.

Keywords: grey-box models, parameter estimation, collocation method, validation, Modelica

Introduction

The continuous progress in ICT has led to the availability of small and low-cost sensors, low-power wireless data transfer protocols, cheap and accessible data storage and powerful servers. Applied to the building sector, these technologies can be used to collect large amounts of building monitoring data at relatively low costs. The abundance of data gives rise to new opportunities and applications in existing buildings like fault detection, energy efficiency analysis and model-based building operation. A first step in many of these applications is the creation of a building energy system model.

Models can be classified according to the white-box, grey-box and black-box paradigm. (Bohlin, 1995; Madsen and Holst, 1995; Kristensen et al., 2004; Henze and Neumann, 2011). Although the boundaries between these categories are blurry and often overlapping, this paradigm is useful for understanding the modelling procedure. White-box modelling bases the model solely on prior physical knowledge of the building. Most building simulation software falls under this category, like TRNSYS, EnergyPlus and many others (Crawley et al., 2008). Black-box modelling bases the model solely on response data (monitoring of the building) and a universal model set, including e.g. AR and ARMAX. Although physical insight is not required for making a black-box model, a model structure has to be chosen and this often involves making assumptions about the system, for example with regard to linearity. Grey-box identification methods and tools cater for the situation where prior knowledge of the object is not comprehensive enough for satisfactory white-box modelling and, in addition, purely empirical black-box methods do not suffice because the involved physical processes are too complex. Grey- and black-box models are also called inverse models.

The difference between white- and grey-box modelling is not in the complexity of the model. A single-state model can be a white-box model if all parameters can be fixed based on physical knowledge only. However, when one or more parameters in a white-box model are estimated based on a fitting of the model to measurement data, the model becomes grey, no matter its complexity. Therefore, the distinction between

white and grey cannot be made by only looking at the model structure: one has to know how the model parameters have been identified.

All three model types can be either deterministic or stochastic. A deterministic model cannot explain the differences between the model output and the true variations of the states (observations). Madsen and Holst (1995) therefore introduced a Wiener process in the system equations to cope with the simplifications of the model and uncertainties in inputs and monitoring. The obtained model is a stochastic state-space model.

For existing buildings with available monitoring data, the grey-box approach is considered to combine the best of two worlds: physical insight and model structure from the white-box paradigm and parameter estimation and statistical framework from the black-box paradigm. This paper describes an approach to grey-box modelling for buildings and the development of a toolbox combining Modelica and Python. The resulting framework will be referred to as *the toolbox* in the remainder of this paper and will be validated on a single-family dwelling. The toolbox is not publicly available, but can be obtained with an open-source license for research purposes by contacting the authors of this paper.

The toolbox has been developed with two purposes in mind. A first application is model predictive control (MPC). In this context, the grey-box model serves as the control model in a feed-back loop with the building. According to Henze (2013), the process of model identification accounts for 70 % of the effort for implementing an MPC controller. Automating this process can therefore reduce the total cost of MPC in buildings. To validate such a control model, the k-step prediction performance is used. A second application is load forecasting for real buildings. The forecast horizon is typically one day or one week. In this case, a different metric to validate the model is required: the simulation performance. This is the model deviation from a measured output in an open-loop simulation when measured disturbances are applied.

It is clear that a model showing a good simulation performance will also have a good k-step prediction performance. The opposite is not true. Therefore, we will use the simulation performance as quality criterion for the model validation.

This paper is split in two parts. The first describes the methodology and development of the toolbox. The second describes the validation results for a well monitored experimental single-family dwelling near Munich, Germany. The validation is carried out for two different experiments on the same building.

Part I
Methodology

Overview

A high-level overview of the toolbox is shown in Figure 1. The toolbox is composed of four major components:

1. the Modelica library *FastBuildings* with thermal zone models, HVAC components and building models;
2. different *.mop files* specifying the model components and which parameters to estimate;
3. *JModelica.org* as a middle layer for compilation of the *.mop files* as well as formulation and solution of the optimisation problem;
4. Python module *greybox.py* delivering the user interface and top-level functionality.

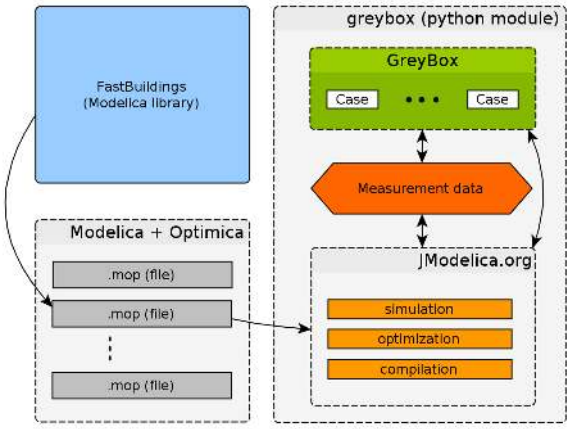

Figure 1: Overview of the grey-box buildings toolbox.

Modelica library *FastBuildings*

Modelica is an equation-based modelling language for cyber-physical systems (Elmqvist, 1997). The object-oriented philosophy stimulates model reuse, resulting in many available libraries, often open-source and free. Modelica is gaining importance in the building simulation community (Wetter, 2011; Wetter and Van Treeck, 2013). The choice for Modelica for the construction

of the models is based on three major arguments (Wetter, 2009). Firstly, Modelica allows for linear, nonlinear and hybrid model formulations and therefore it does not limit the model structure as such. Secondly, Modelica is equation-based, thus allowing efficient Newton-type solvers to be used as an alternative to for example genetic algorithms. Thirdly, Modelica has a *connector* concept to support component-based modelling.

The *FastBuildings* library targets low-order building modelling. The library has sub-packages for thermal zone models (including windows), HVAC, user behaviour, inputs, buildings and examples. Single and multi-zone building models can be created easily by instantiating one of the predefined templates in the `Building` sub-package and redeclaring the desired submodels, like the thermal zone, HVAC or window model. The following design principles are applied throughout the library.

- The thermal connectors are `HeatPorts` from the `Modelica.Thermal` package, which is part of the *Modelica Standard Library* (MSL).
- Thermal resistors and capacitances are not used from the MSL. Simplified versions with less auxiliary variables are implemented. They have exactly the same interface and connectors to ensure compatibility with the MSL.
- A strict naming convention is used for consistency and to enable the *greybox.py* toolbox to automate certain tasks.
- The library heavily relies on the `extends` construct in order to avoid code duplication. This is specifically useful for the thermal zone models that have increasing complexity as a function of their order.
- An `inner/outer` component `simFasBui` passes all inputs like weather data, occupancy etc. from the top level to all sublevels.
- The models for thermal zones, HVAC and user behaviour have exactly the same interface as their equivalents in the *IDEAS* library. *IDEAS*, developed by KU Leuven and 3E, is an open-source library for modelling and simulation of buildings and integrated districts (Baetens et al., 2012). Therefore, it is very easy to replace one or more detailed components from an *IDEAS*-based model by a low-order equivalent from the *FastBuildings* library.

Currently, the thermal zone models available in the *FastBuildings* library are based on a resistor-capacitance (RC) network analogy which is often used for the modelling of thermal processes. This is however not required: any model that specifies a relationship between the heat flows and temperatures at the interface of a thermal zone can be implemented. Different examples of models in the library are schematically presented in Figure 5 in Part II.

The *FastBuildings* library largely contains the domain specific knowledge that is fundamental in grey-box modelling. Different thermal zone models often encountered in literature are present in the library and it is very easy to add more models (Davies, 2004; Bacher and Madsen, 2011; Sourbron et al., 2013; Reynders et al., 2014). The *FastBuildings* library is very dynamic in the sense that it is being extended with extra building models the more it is applied to different cases. How these models are chosen in a forward selection approach is explained in Section *Toolbox functionality and work flow*. The *FastBuildings* library is distributed with the *Modelica license 2* and can be found in the *openIDEAS* source code repository on Github (KU Leuven and 3E, 2014).

The JModelica.org platform

The toolbox relies heavily on the JModelica.org platform, which is an open-source tool for simulation and optimisation of dynamic systems described by Modelica code (Åkesson et al., 2010). For simulation purposes, JModelica.org uses the Functional Mockup Interface (Blochwitz et al., 2011). For optimisation purposes, JModelica.org offers various algorithms and also supports the Modelica language extension Optimica (Åkesson, 2008). Optimica allows for high-level formulation of dynamic optimisation problems of the type presented in Section I.

Every model structure for which the parameters have to be estimated is characterised by a different *.mop* file. These files are very similar to ordinary Modelica (*.mo*) files, but they can also contain Optimica code. Each *.mop* file has the same structure and has to define two models: one model for simulation, called `Sim`, and one for parameter estimation, called `Parest`. By default, the models are based on the *FastBuildings* Modelica library, which has been developed in conjunction with this toolbox. However, this is not required for the toolbox to work, as long as some naming conventions are followed. Any parameter present

in the model can be estimated, including initial values of the states.

The toolbox estimates the unknown model parameters using JModelica.org's algorithm based on direct collocation. Collocation is used to discretise time, which reduces the optimisation problem to a nonlinear program (NLP), as presented in Section *Solution method* and described in more detail in Magnusson and Åkesson (2012), where in particular optimal control is also treated. JModelica.org utilises third-party NLP solvers, which require first- and second-order derivatives of all expressions in the NLP with respect to all decision variables. CasADi is used to obtain these by algorithmic differentiation (Andersson et al., 2012). In this paper we use the NLP solver IPOPT with the sparse linear solver MA27 from HSL (Wächter and Biegler, 2006; HSL, 2013).

Problem formulation

Identification of the unknown model parameters is formulated as a dynamic optimisation problem of the general form

$$\begin{aligned}
&\text{minimise} && \int_{t_0}^{t_f} e(t)^T Q e(t)\,\mathrm{d}t, && \text{(1a)}\\
&\text{with respect to} && x(t), w(t), u(t), p,\\
&\text{subject to} && F(t, \dot{x}(t), x(t), w(t), u(t), p) = 0, && \text{(1b)}\\
& && x(t_0) = x_0, && \text{(1c)}\\
& && \forall t \in [t_0, t_f].
\end{aligned}$$

The system dynamics are modelled by a differential-algebraic equation (DAE) system (1b), where t is the time, $x(t)$ is the state, $w(t)$ is the vector-valued algebraic variable, $u(t)$ is the vector-valued system input, which includes both control variables and disturbances, and p is the vector of parameters to be estimated.

Algebraic variables often occur in Modelica models. A typical example is the conversion of measured electricity consumption in a radiative and a convective fraction. The resulting radiative and convective heat fluxes are contained in $w(t)$.

The DAE system may be implicit, non-linear, time-variant, and high-index. It is the result of the compilation of the *FastBuildings* model. In the case of high-index systems, index reduction is automatically performed by the JModelica.org compiler.

Since a gradient-based method is applied to solve the dynamic optimisation problem, F needs to be twice continuously differentiable with respect to all of its arguments (except the first one). This disables the use of hybrid models. Initial conditions are given by specifying the initial state, as given by (1c), where t_0 is the start time. The initial state is usually unknown, in which case some, or all, elements of x_0 can also be introduced as elements of the vector p.

The objective (1a) of the optimisation is to minimise the integrated quadratic deviation e of the model output from the corresponding measurement data. The model output y is typically some of the states, but could also be some of the algebraic variables (and also inputs, as discussed below). The matrix Q, which typically is diagonal, is used to weigh the different outputs. The measurement data is assumed to be a function of time, denoted by y_M. Since measurements are typically discrete in time, they are simply interpolated linearly to form y_M. The output deviation e is then given by

$$e(t) := y(t) - y_M(t). \tag{2}$$

The inputs can be treated in two different ways. The first is to assume that the inputs are known exactly by their measurement data and treat them as fixed values instead of decision variables. The second way is to have an error-in-variables approach where the inputs are kept as decision variables and treat them as model output, that is, include them in the vector y and penalise their deviation from the corresponding measurement data. The second way is useful for coping with uncertainties in measurement data.

Solution method

The approach taken to solve the optimisation problem (1) is based on low-order direct collocation as presented by Biegler (2010). The idea is to divide the time horizon into a number of elements, n_e, of fixed (but possibly distinct) lengths h_i and approximate the time-variant system variables $\dot{x}, x, w$ and u by a polynomial of time within each element, called a collocation polynomial. These polynomials are determined by enforcing the dynamic constraints at a certain number of points, n_c, within each element. These points are called collocation points and $t_{i,k}$ is used to denote collocation point number $k \in [1..n_c]$, where $[1..n_c]$ denotes the integer interval between 1 and n_c, in element number $i \in [1..n_e]$.

The system variables' values at these points, denoted by

$$(\dot{x}_{i,k}, x_{i,k}, w_{i,k}, u_{i,k}, e_{i,k}) := (\dot{x}(t_{i,k}), x(t_{i,k}), w(t_{i,k}), u(t_{i,k}), e(t_{i,k})),$$

are then interpolated based on Lagrange interpolation polynomials to form the collocation polynomials. There are different schemes for choosing the placement of collocation points with different numerical properties. In this paper we only consider Radau collocation.

All collocation methods correspond to special cases of implicit Runge-Kutta methods and thus inherit desirable stability properties making them suitable for stiff systems.

This approximation reduces (1), which is of infinite dimension, into a finite-dimensional nonlinear program (NLP) of the form

$$\begin{aligned}
\text{min.} \quad & \sum_{i=1}^{n_e} \left(h_i \sum_{k=1}^{n_c} \omega_k e_{i,k}^T Q e_{i,k} \right), && \text{(3a)} \\
\text{w.r.t.} \quad & \dot{x}_{i,k}, x_{i,l}, w_{i,k}, u_{i,k}, p, \\
\text{s.t.} \quad & F(t_{i,k}, \dot{x}_{i,k}, x_{i,k}, w_{i,k}, u_{i,k}, p) = 0, && \text{(3b)} \\
& x_{1,0} = x_0, && \text{(3c)} \\
& x_{n,n_c} = x_{n+1,0}, \quad \forall n \in [1..n_e - 1], && \text{(3d)} \\
& \dot{x}_{i,k} = \frac{1}{h_i} \sum_{j=0}^{n_c} \alpha_{j,k} \cdot x_{i,j}, && \text{(3e)} \\
& \forall i \in [1..n_e], \quad \forall k \in [1..n_c], \quad \forall l \in [0..n_c].
\end{aligned}$$

The NLP objective (3a) is an approximation of the original objective (1a) based on Gauss-Radau quadrature, where the measurement error $e_{i,k}$ in each collocation point is summed and weighted by the corresponding element length h_i and quadrature weight ω_k, which depends on the choice of collocation points. Note that the decision variables are not only the unknown parameters p, but also the discretised system variables $\dot{x}_{i,k}, x_{i,l}, w_{i,k}$, and $u_{i,k}$ (unless it has been eliminated). The constraint (1b) from the continuous-time model dynamics is transformed into the discrete-time constraint (3b) by enforcing it only in each of the collocation points.

Since the states need to be continuous (but not differentiable) with respect to time, the new continuity constraint (3d) needs to be introduced. Because we use Radau collocation, where no collocation point exists at the start of each element, this also requires the introduction of the new variables $x_{i,0}$, which represent the value of the state at the start of element i. With the introduction of $x_{1,0}$, the initial condition (1c) is transcribed into (3c).

Finally, we introduce the constraints (3e) to capture the dependency between x and $\dot{x}$, which is implicit in (1). The state derivative $\dot{x}_{i,k}$ in a collocation point is approximated by a finite difference of the collocation point values of the state in that element. The finite difference weights $\alpha_{l,k}$ are related to the butcher tableau of the Runge-Kutta method that corresponds to the collocation method.

All that remains is to solve the NLP (3) in order to obtain an approximate solution to the original problem (1). We do this numerically using IPOPT, as described in Section *The JModelica.org platform*.

Toolbox functionality and work flow

The user interacts with the toolbox through the *greybox.py* Python module. This module defines two classes `GreyBox` and `Case`, as shown in Figure 1. The idea is to instantiate the `GreyBox` class once for the system identification of a given building. The `GreyBox` object will contain many different instances of the `Case` class. Every `Case` is an attempt (successful or not) to obtain a model for the given building. The `Case` therefore keeps track of the model structure, identification data, initial guess, solver settings and results of a single parameter estimation attempt. The functionality of the toolbox is packed in methods of the `GreyBox` class and can be grouped into different domains, according to the foreseen workflow. This is shown in Figure 2. This workflow is discussed in the following paragraphs.

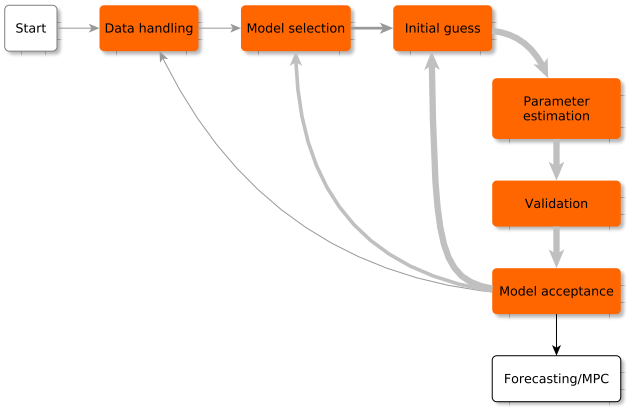

Figure 2: Workflow and high-level functionality in the toolbox.

The methods under **data handling** are used to load the data files, resample the data if desired, create data

slices of given lengths (for example one week, but can be any period) and show a plot of any data slice. Typically, one data slice is the training set, and the other slices can be used for cross-validation. Resampling the data is an important step because the toolbox automatically chooses the collocation points to coincide with the measurements. Thus, the (size of) the numerical problem (3) is strongly dependent on the chosen sampling time, and it is often a good strategy to start with a large sampling time and refine it in a later stage.

When the data has been pre-processed, a model structure has to be specified in the **model selection** step. This is accomplished by specifying the path to a *.mop* file. There are two models in the *.mop* file: a Modelica model for simulation, called `Sim`, and a Modelica + Optimica model for parameter estimation, called `Parest`. The main difference is that in the model `Parest`, the value of the Optimica attribute `free` is set to `true` for each parameter to be estimated. The compilation of both models happens automatically by invoking the corresponding JModelica.org functionality. Model information (state vector, parameter vector and required inputs) and solver settings are also obtained in this step.

Before the parameter estimation can be attempted, an **initial guess** has to be specified for each element in the parameter vector. These can be set by default, by inheritance, by Latin hypercube sampling or manually.

When the default initial guesses are used, an appropriate value is chosen for each parameter, based on its name. For example, the naming convention in *FastBuildings* forces all parameter names for thermal resistances to start with 'R' (like R_{Wal}), for thermal capacities with 'C' (like C_{Zon}), for fractions with 'fra' (like fra_{Rad}), etc. Based on the first letter(s) of a parameter to be estimated, a default initial value will be set.

An alternative for obtaining the initial guess is to start from the optimised parameter vector of a previous case, the *parent* case. This is especially useful when a new *.mop* file is selected that has similarities with a previously processed *.mop* file. Due to the naming conventions in *FastBuildings*, the corresponding parameters will have the same name. Therefore, the best initial guess for a similar parameter in the new model will be the optimal value from the parent. For new parameters, the default initial guess method described above is used.

The last automated option to obtain initial guesses is based on Latin hypercube sampling. Due to the non-convexity of the problem, there can potentially exist many local minima. To investigate the parameter search space more systematically and increase the chances of finding a global minimum, a Latin hypercube sampling method has been implemented. This method will take a single initial guess as well as lower and upper bounds for each parameter and derive a univariate beta distribution from these three values. The distribution can be symmetric or asymmetric, as shown in Figure 3.

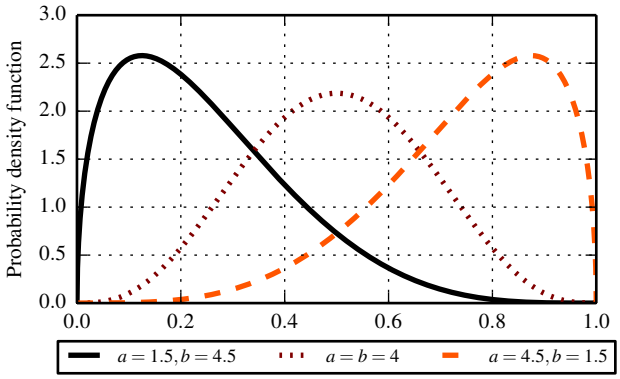

Figure 3: Symmetric and asymmetric beta distributions. The parameters a and b characterise the probability density function.

The Latin hypercube sampling will then derive n stratified samples from each distribution and combine them randomly to obtain n different initial guesses. Each of these guesses will be copied to a new `case` to keep track of the results.

When a `case` has an initial guess for the parameter vector, the **parameter estimation** can be started. However, the NLP (3) requires good initial guesses for each of the decision variables (including all collocated states and algebraic variables). This is handled by simulating first with the `Sim` model and the initial guess of the parameter vector. The resulting simulation trajectories are used as initial guesses for the decision variables in (3). Numerical scaling factors for each system variable are also computed as the infinity norm of the corresponding trajectory.

The solution time and the number of iterations can vary a lot depending on the initial guess and the ability of the model to represent the measurement data. Through the IPOPT interface, the toolbox allows the specification of a maximum solution time and/or a maximum number of iterations after which it will interrupt the optimisation.

The estimation adds the optimised parameter vector to the `case`, as well as the IPOPT solver statistics.

The **validation** of the results is always based on a post-simulation with the `Sim` model and the optimised values of the parameter vector. This can be done on the training data (auto-validation) or on any other dataset (cross-validation). There are both visual and quantitative validation methods. The visual methods contain, for example, time series plots of the resulting trajectories and corresponding residuals, scatter plots of the residuals with monitoring data and a plot of the autocorrelation function of the residuals. This also implies a check on the weights of the matrix Q from (1a) in case the error-in-variables method is used. When a full Latin hypercube sample has been estimated, a visual check of the different local optima is implemented. This can be used to judge whether the sample was large enough to suppose that the global optimum has been found. The quantitative methods are based on a computation of the root-mean-square error (RMSE) for each trajectory in the vector e from (2). As the RMSE is computed based on post-simulation with adaptive step length, discretisation errors in the collocation method are accounted for in the model validation process.

A computation of the confidence interval for each of the estimated parameters is implemented. This gives an indication of the accuracy of the estimation and the parameter's influence on the model's input-output behaviour. The standard deviation of the estimated parameters $\hat{p}$ is computed according to Englezos and Kalogerakis (2000). The standard deviation for parameter i is the square root of the diagonal element on (i,i) in the covariance matrix $\mathrm{cov}(\hat{p})$ of the estimated parameters, which is given by

$$\mathrm{cov}(\hat{p}) = \hat{\sigma}^2 (J^T J)^{-1},$$

where $\hat{\sigma}^2$ is the estimated variance of the output deviation e. J is composed based on the results of a simulation with the estimated parameters and sensitivities computations activated. J contains the sensitivities of the model outputs with respect to the estimated parameters as shown in Eq. (4).

$$J = \begin{bmatrix} \frac{\partial y_1(t)}{\partial p_1} & \frac{\partial y_1(t)}{\partial p_2} & \cdots & \frac{\partial y_1(t)}{\partial p_{n_p}} \\ \frac{\partial y_2(t)}{\partial p_1} & \frac{\partial y_2(t)}{\partial p_2} & \cdots & \frac{\partial y_2(t)}{\partial p_{n_p}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_{n_y}(t)}{\partial p_1} & \frac{\partial y_{n_y}(t)}{\partial p_2} & \cdots & \frac{\partial y_{n_y}(t)}{\partial p_{n_p}} \end{bmatrix} \tag{4}$$

In this equation, $y_1 \ldots y_{n_y}$ are the n_y model outputs and $p_1 \ldots p_{n_p}$ are the n_p free parameters. This method can only be applied when the model output is equal to one or more states.

The final step in the system identification is **model acceptance**. Model acceptance is needed on two levels: for a single model, and between different models. For a single model, generally a Latin hypercube sampling is executed and the resulting global optimum is accepted if it is a *valid* solution. Valid means that:

- the parameters do not lie on the specified minimum or maximum bounds;
- the parameter values are physically reasonable;
- the confidence intervals are within reasonable bounds.

These criteria are not totally objective and often require an expert's check on the model. If the global optimum is not valid, the local optima are analysed and may contain a valid model. If no valid model is found within the sample, there are different options. A new Latin hypercube sample can be generated with different distributions and/or a larger sample size. Sometimes it can also help to change numerical settings for the solver or to resample the data differently. If none of these solutions leads to a valid model, the selected model structure cannot be matched to the considered identification dataset and a different model structure has to be chosen.

A *forward selection* approach is preferred for inter-model acceptance. This approach starts with a very simple model, generally a first order single zone model with a low number of free parameters. Then, model order and complexity are increased until (i) the models cannot be validated or (ii) the RMSE in cross-validation cannot be improved anymore. The selection of the candidate models in this procedure is not systematic. The procedure can be carried out manually or automatically. The manual solution is a trial and error procedure, often accompanied by tailor-made models based on the results of previous identification attempts. For the automatic solution, the modeller makes a selection of models that is passed to the toolbox. The toolbox will then sort the models according to the number of parameters and start with the identification of the least complex one. Validation tests are specified by the user and the toolbox will automatically select the best valid model. Which models are passed to the toolbox is again a user-specified choice based on expertise and available meta-information. It is of course possible to pass every model of the *FastBuildings* library, but this will often result in an unacceptable CPU-time. Even if the forward selection procedure is

not as strict as the one presented by Bacher and Madsen (Bacher and Madsen, 2011), it has several practical advantages. Firstly, the initial guesses and distributions for the parameters can be inferred from previously identified models. This is enabled by a strict naming convention in the *FastBuildings* library and by the fact that similar model components like walls, infiltration, solar gains etc. are repeated in more complex models. Secondly, starting from the first identified model, there is always a reference performance (RMSE) with which the new results can be compared. This approach avoids overfitting of the model, as will be demonstrated in Part II.

Data requirements

The developed grey-box modelling approach is intended for existing buildings. The aim is not to develop a detailed emulator model, but to develop a simplified low-order model that works well in an MPC context or for predicting loads in buildings. For practical use in existing buildings, we can not count on the existence of detailed emulator models, hence model order reduction approaches are excluded. Therefore, the aim is to develop a methodology that can cope with very little meta-information and a limited amount of measurement data.

With regard to the meta-information, the requirements depend on the complexity of the model. For very simple models, typically single-zone and without HVAC components, there is no need for any a-priori knowledge of the building. Only the location of the building has to be known if weather data is to be obtained from a generic weather service. Other building properties, like building size, orientation, window area, envelope properties etc. are not required. Nevertheless, this information can be beneficial for fixing initial guesses and for validation of estimated parameters.

The more meta-information we want to use, the more manual interventions are needed in the identification procedure. Therefore, this information is optional. By default, initial guesses are hard coded based on naming conventions. This means for example that all resistances in a model are attributed the same initial value, unless a resistance with the same name has been estimated in the parent `case` of the current model. In this situation, the initial guess is the optimised value from the parent `case`. Experience shows that the combination of these initial guesses with the Latin hypercube sampling is sufficient to find good parameter estimates also without using a-priori knowledge. Also for the very first model in the forward selection approach, which is supposed to be very simple, this works reliably.

The toolbox uses upper and lower bounds for the parameter estimation. The main reason for these bounds is to reduce the feasible region and thus the search space. Most bounds reflect basic physical laws, for example by imposing that resistances, capacities, *gA* values and fractions have to be positive. For fractions, an upper bound of one can be imposed, but it can also be relaxed. This can be used for example to obtain internal gains as a fraction of measured electricity consumption. Thanks to body heat gains, this estimated fraction is allowed to be larger than one. Most parameters do not have an upper bound because it is impossible to specify them without using meta-information. If a parameter estimation would result in unrealistic high values, this is to be detected by either too high confidence intervals, an expert check on the values or bad cross-validation. The use of bounds for the starting temperatures of the states is illustrated in Part II.

Sometimes, meta-information can be replaced by an analysis of the available data. For example, a large window area on a specific orientation can be discovered automatically by a correlation analysis on zone temperatures with incident radiation on different orientations. This information is then used to select which solar radiation components are used as disturbances in the model.

For more complex models however, more meta-information is needed. When a multi-zone model is created, information about the position of available zone measurements (temperature, humidity, electricity consumption etc.) is very useful to decide on the zoning strategy. If the model has to contain the HVAC system, some information is necessary, in particular about the presence of specific equipment. This information is used to adapt the model structure to the installed HVAC system.

The first requirement for the monitoring dataset is that the meaning of all variables is clear. The dataset has to contain at least the indoor and ambient temperature and heating/cooling loads at hourly interval. The ambient temperature (and other weather variables) can also be obtained from a weather service if the location of the building is known. The availability of more variables is beneficial and will improve the model. Electricity consumption monitoring (with sub-metering for plug power) is strongly recommended. More submeters for electricity always improve the information

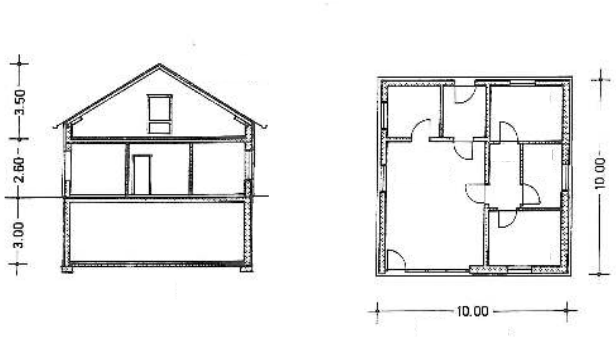

Figure 4: Front view and ground floor of the experimental house (north is up). For experiment 2, Zone 2 *refers to the 3 small rooms in the north.*

content and usability, for example for creating equipment scheduling profiles. When the HVAC system is to be modelled, a measurement of the energy use of different components is required.

Occupancy measurements are often not available. Mostly, the model does not need the occupancy itself, but the internal gains from body heat transfer. In offices, these are correlated with plug power. Alternatively, occupancy can be modelled based on measurements of relative humidity or CO_2. This is not yet implemented in the current version of the toolbox.

Part II
Validation

Methodology

Experimental setup

A detailed experiment was set up by Fraunhofer IBP (Holzkirchen, Germany) in order to collect monitoring data from well-known buildings near Munich, the *twin houses* (Kersken et al., 2014). We use monitoring data from one of these houses. A schematic overview of the building is given in Figure 4.

Two experiments are performed, resulting in two datasets of about 40 days each. Each experiment consists of consecutive periods of free floating operation, a randomly ordered logarithmically distributed binary sequence (ROLBS) for heat inputs and a temperature controlled operation. The differences between the experiments are detailed in Table 1.

There are no users in the experimental house. The heating consists of electrical heaters in each of the spaces. By consequence, the models presented in this paper focus on the building only and do not include components for users or HVAC.

Setup	**Exp. 1**	**Exp. 2**
Period	Summer	Winter
Blinds	Closed	Open
Zoning	Single zone	Two zones (doors closed and sealed)
Heating	Sequence	Different sequence

Table 1: Overview of the differences between both experiments.

Control versus forecasting

The presented grey-box approach aims at identifying models for forecasting and control. We have argued that the simulation performance is the correct criterion to validate the models, and it is sufficient to validate models for forecasting. However, a model with a good simulation performance is not necessarily suited for optimal control. Some additional criteria are:

- observability: models that are not observable cannot be initiated in the right state with an adapted state estimation procedure;
- complexity: the model has to fit in an optimal control framework. In particular, non-linear models are difficult to optimise;
- solver time: even if the requirements above are fulfilled, solving the OCP may require more time than is available between subsequent control steps.

However, we are confident that the models presented in this paper are suited for MPC. Firstly, state estimation has been implemented on identical and very similar models as those presented in this paper (Vande Cavey et al., 2014). Secondly, the presented models are all linear (even if the *FastBuildings* library and our our optimal control framework *JModelica.org* both allow non-linear models). Thirdly, we have tested those models in MPC and the computation times for solving the OCP are around one minute or much less depending on the initialisation and forecasting horizon.

The ultimate validation of a control model is by assessing its control performance, but this is subject to future work. We therefore assume that the models considered here are suitable for control if they have a good simulation performance.

Simulation performance

The simulation performance is quantified by the simulation error *SE*. It is a weighted average of the root mean square error (RMSE) for a set of n model outputs:

$$SE = \sum_{j=1}^{n} q_j(RMSE_j), \tag{5}$$

where

$$RMSE_j = \sqrt{\frac{\sum_{i=1}^{m} (y_j(t_i) - M_j(t_i))^2}{m}}.$$

The weighting factors q_j are the diagonal elements of matrix Q of (1a). For each selected output variable, $y_j(t_i)$ is the model output at time instant t_i and $M_j(t_i)$ is the corresponding measurement. The model outputs are taken at m time instants, corresponding to the data points. These m data points may be the raw measurement data or the result after a downsampling operation. The measurement data M is used to form y_M in (2) by interpolation. However, as the toolbox sets the collocation points so that they coincide with the time instants t_i defined by the (resampled) measurement data, no interpolation is actually needed.

It is important to implement the validation as cross-validation (as opposed to auto-validation). This means computing the simulation performance on a section of the dataset that was *not* used for identification. For validation of the control performance, a short validation period of about 1 day is sufficient. However, in order to validate the load forecasting application, we need a much longer dataset. As our experiments contain each 40 days of data, we split them in two equal parts: the identification and (cross-)validation subsets. We will refer to the simulation performance in cross-validation as *prediction performance*.

For the validation simulation, an initial state vector is required. This state vector can be identified by filtering techniques from measurement data up to the start time of the validation dataset. In this study, the validation dataset starts where the identification dataset ends. Therefore, the state vector can be determined as the model state at the end of the identification period.

As explained in Part I, the forward selection approach results in a single grey-box model for a given dataset. We will call this model the *accepted* model. It is the model resulting in the lowest *SE* on the cross-validation dataset that is *valid*.

For the experimental house however, almost all meta-information is available: dimensions, construction, window positions, material properties, ventilation rates etc. In order to validate the grey-box toolbox for its intended use and work flow, none of this meta-information is used in the modelling phase.

Experiment 1

Data handling and zoning

The available dataset is very detailed. There are different temperature sensors (in different rooms and also within a single room to measure stratification), heat flux measurements, humidity sensors etc. The sampling period of the data is 10 minutes. We start the forward selection procedure with a simple single-zone model lumping all heated spaces, and we neglect the interaction with the boundary spaces (attic and cellar). For this thermal zone, an average zone temperature T_{Zon} has to be defined. As we know nothing about the building (we do not use the available meta-information) we just average all temperatures of the heated spaces and sum their heating loads. We also resample the 10 minute data to hourly values. This reduces the size of the numerical problem because the toolbox automatically chooses the collocation points to coincide with the measurements. An overview of all models that are successfully identified *and* validated in the forward selection procedure is shown in Figure 5.

A first model

The first single-zone model is model A of Figure 5. The model has only one state T_{Zon}. The free parameters are the thermal capacity C_{Zon} and start temperature $T_{Zon}(0)$, the total solar transmittance gA of the windows and a total heat loss coefficient R_{Wal} representing all heat losses to the ambient temperature T_{Amb} (see Table 2).

Model inputs are the ambient temperature T_{Amb}, global horizontal radiation $I_{Glo,Hor}$, and summed heating loads Q_{Hea}. The use of the global horizontal radiation instead of the radiation on several vertical surfaces is an approximation that allows us to estimate only one gA value. In subsequent model refinements we will use the radiation components on different orientations and multiple windows. The identification is based on a minimisation of $RMSE(T_{zon})$.

As for all models that will be discussed below, a Latin hypercube sample with initial guesses has been created and the best result of this sample is presented. Table 2 gives the resulting parameter estimates and their estimated 95% confidence interval for the first order model. The $RMSE_{auto}$ is 0.61 K and the $RMSE_{cross}$

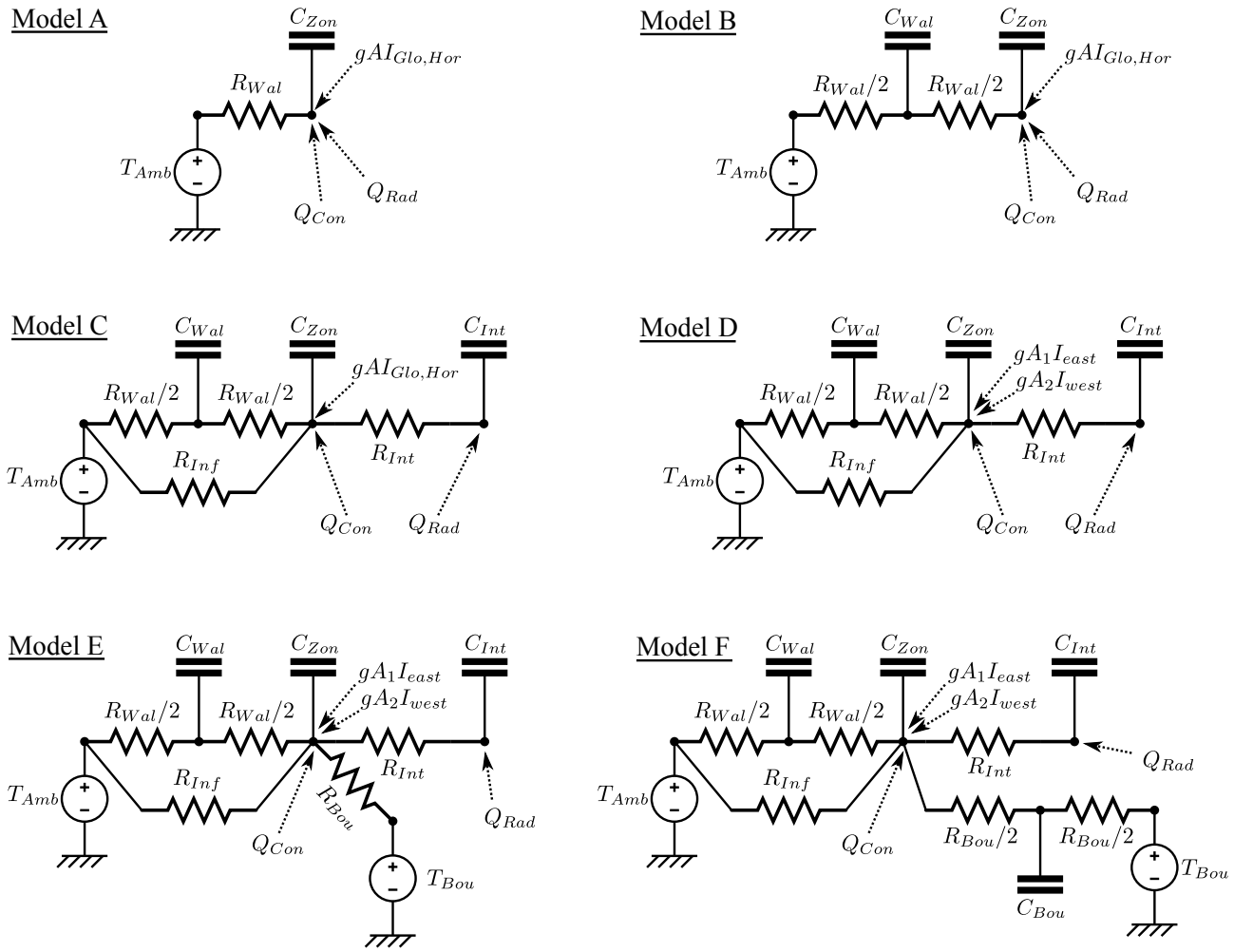

Figure 5: Overview of identified valid models for experiment 1. See the nomenclature at the end of this article for the meaning of the variables.

Parameter	Estimated value	95% conf.
C_{Zon}	5.4e7 J/K	+/- 2.2e6
R_{Wal}	7.0e-3 K/W	+/- 1.2e-4
$T_{Zon}(0)$	30.0 °C	-
gA	2.65 m^2	+/- 0.16 m^2

Table 2: Overview of free parameters and their estimated values for the first order model (model A).

is 2.01 K which is not very accurate. Still, this model is a good starting point and it provides useful initial guesses for the parameters in more detailed models.

Model refinements

Different models of increasing complexity have been identified. Without discussing all attempts in detail, we try to give an overview of the model improvements. An overview of the RMSE values for auto and cross-validation of all successfully identified models is shown in Figure 6 and the model schemes are shown in Figure 5. The accepted model is discussed in more detail in the next section.

Model B The single state model cannot capture all dynamics in the measurement data. More states are needed. A variety of additional states can be suggested, such as internal mass T_{Int}, inertia of the building envelope T_{Wal}, inertia of the heat emission system T_{Hea} etc. Different two-state models have been identified, the best model (on cross-validation) has an additional state for the walls (model B). This model has six free parameters (of which two are initial temperatures of the states). The $RMSE_{auto}$ is 0.31 K and $RMSE_{cross}$ is 0.76 K. This is a substantial improvement compared to the single-state model.

Model C We can still improve the model by increasing its order to three states. Many attempts lead indeed to lower RMSE values in auto-validation, but *not* in cross-validation. This means these models are overfitted. We found one model however that slightly improves $RMSE_{cross}$ to 0.74 K. This model is able to reduce $RMSE_{auto}$ by 50% to 0.15 K but this barely results in a better prediction performance. The model has an additional state for the thermal inertia in the zone and an additional resistance r_{Inf} in parallel with the wall. This leads to 10 free parameters.

Model D An analysis of the residuals reveals a correlation between model error and solar radiation. We still use the global horizontal radiation $I_{Glo,Hor}$ to estimate a single gA value. The incorporation of solar gains can be refined by adding windows and connecting each window to a different solar radiation. In our attempts, we obtained the best results with two windows, connected to the vertical global radiation on East and West respectively. This resulted again in a large reduction of $RMSE_{auto}$ to 0.10 K and a small reduction of $RMSE_{cross}$ to 0.71 K.

Model E When analysing the data, we have found a possible cause for the discrepancy between the results in auto and cross-validation. In the identification dataset, the mean attic temperature is higher than in the validation set, leading to overestimation of temperatures on cross-validation. When we add a thermal resistance to the attic and estimate its value we can indeed improve the prediction performance. The obtained model has an $RMSE_{auto}$ of 0.09 K and $RMSE_{cross}$ is 0.56 K. The model has 12 parameters, and none of the estimated values are physically impossible or are positioned at their minimum or maximum boundary. This is an important validation criterion, it requires however an expert check.

All subsequent attempts to improve the model lead to *non-physical* models. This may seem a non-issue since we are dealing with grey-box models in which the parameters are allowed to represent lumped characteristics. However, experience shows that when models have unrealistic values for the physical (lumped) parameters, these are always accompanied by extremely large confidence intervals.

Model F A different situation occurs when all physical parameters have acceptable values, but the estimated initial temperatures for the states are at the imposed boundaries (270 K and 310 K by default). When this happens for a state corresponding to a large time constant (large RC value), the state does not act very dynamically and the energy balance of the model is biased. However, it is often possible to obtain a valid model by limiting the initial state temperatures to a narrow bound based on physical insight and analysis of the measurement data. This will never lead to a lower RMSE on auto-validation because we exclude the *optimal* solution from the feasible region. However, it may result in a lower RMSE on cross-validation and thus a better prediction performance. What happens is that the numerically (slightly) better solution is shifted to a more physical solution.

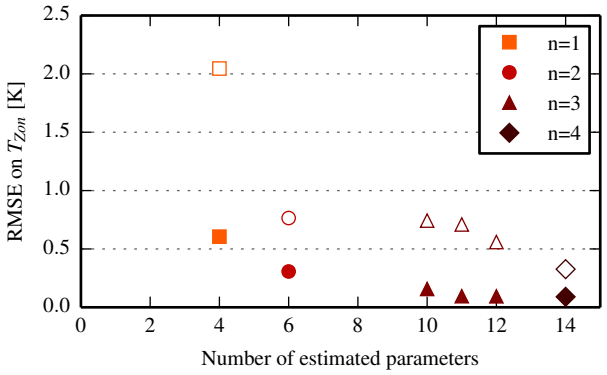

Figure 6: RMSE values for auto-validation (filled markers) and cross-validation (hollow markers) for the different models as a function of the number of estimated parameters and the model order n.

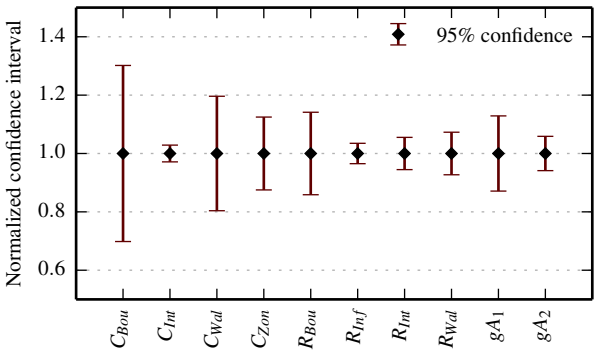

Figure 7: Normalised confidence interval for the parameters in the accepted model for experiment 1.

This is also observed in experiment 1. We try to improve the model with 12 parameters by adding a state for the boundary with the attic (model F). This adds two parameters to be estimated, the thermal capacity of the boundary C_{Bou} and the initial temperature of this state $T_{Bou}(0)$. The optimisation returns $T_{Bou}(0) = 270$ K (-3.15 °C). Physically, we know this temperature should lie somewhere between the temperatures of the zone and the attic. By increasing the lower bound to 22.9 °C we find a valid model with an $RMSE_{auto}$ of 0.09 K and a strongly improved $RMSE_{cross}$ of 0.33 K. This model, with 14 parameters, is the accepted model. It will be discussed in more detail below. Further attempts led to non-physical models or did not improve the forecasting performance while adding unnecessary model complexity.

Model validation

Table 3 gives the resulting parameter estimates for model F. The normalised confidence intervals are shown in Figure 7. The toolbox does not compute confidence intervals for the initial temperatures of the states. We can see that all confidence intervals are reasonably small. Together with the physically meaningful parameters this is an indication of validity for our model. More specifically, this indicates that the model is not overfitted.

Figures 8 and 9 show the measured and simulated zone temperatures for the identification and validation datasets respectively. The latter represents the simulation performance SE as we have only one model output T_{Zon} in (5). We can see that, given perfect predictions of the disturbances, the model is able to predict the measured temperature very well, even in an

Par.	Meaning	Value
C_{Bou}	State boundary to attic	8.1e+07 J/K
C_{Int}	State internal mass	2.6e+07 J/K
C_{Wal}	State building envelope	2.3e+08 J/K
C_{Zon}	State zone	3.4e+06 J/K
$T_{Bou}(0)$	Initial temperature	22.9 °C
$T_{Int}(0)$	Initial temperature	29.6 °C
$T_{Wal}(0)$	Initial temperature	27.1 °C
$T_{Zon}(0)$	Initial temperature	30.3 °C
R_{Bou}	Resistance to attic	3.4e-2 K/W
R_{Inf}	Resistance to ambient	1.5e-2 K/W
R_{Int}	Resistance $C_{Zon} \leftrightarrow C_{Int}$	1.0e-3 K/W
R_{Wal}	Resistance envelope	1.8e-2 K/W
gA_1	gA windows East	0.46 m^2
gA_2	gA windows West	1.03 m^2

Table 3: Overview of estimated parameters for the accepted model (model F) for experiment 1.

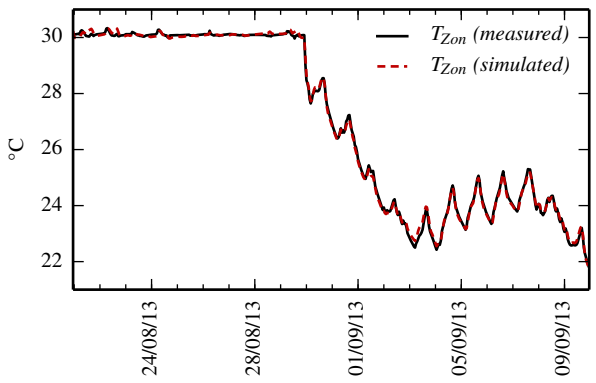

Figure 8: Measured and simulated zone temperature for the identification dataset (auto-validation) in the accepted model for experiment 1.

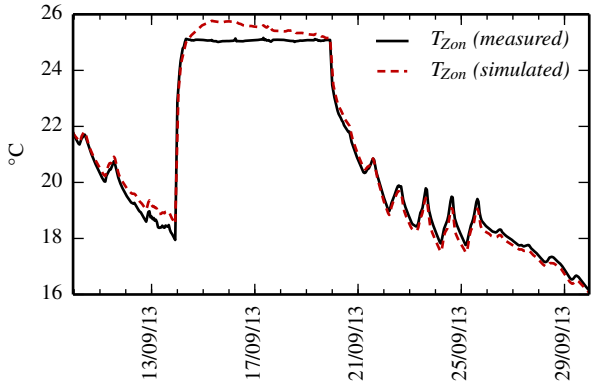

Figure 9: Measured and simulated zone temperature for the validation dataset (cross-validation) in the accepted model for experiment 1.

open-loop simulation over 20 days. However, a disadvantage of the accepted model is that it is dependent on a prediction of the temperature of the attic. Without this information, the simulation performance would be poor. The control performance can still be good if an online estimation and/or state estimation compensates for the slow dynamics caused by the presence of the attic. This will be elaborated in future research.

From these results, we conclude that the grey-box model is validated for both forecasting and control of the dwelling monitored in this experiment.

Experiment 2

Data handling and zoning

One of the fundamental differences compared to experiment 1 is that in this experiment, two different temperature regimes are maintained leading to two distinct thermal zones. Each of these zones is composed of different rooms. The most basic zoning approach consists of modelling only two zones and averaging the measurements in individual rooms accordingly. We will call these zones *Zon*1 and *Zon*2. Models with more than two zones have not been investigated.

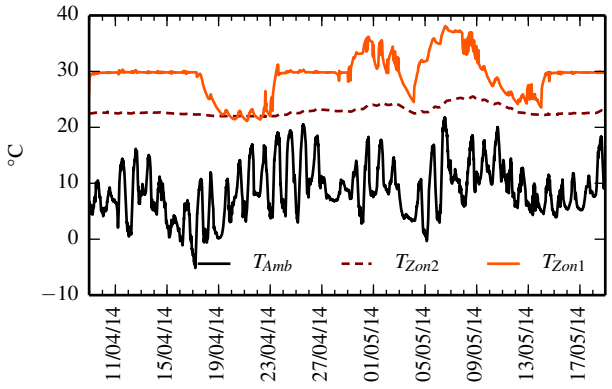

Figure 10: Measured (averaged) temperatures for both zones and ambient temperature for the full experiment 2. The first half of the data is the identification dataset, the second half is the validation set.

Again, we do not use available meta-information but simply average all available measurements and we resample the data to hourly values. From a plot of the averaged measured zone temperatures (see Figure 10), we can see that T_{Zon2} has a different control and is very stable. Also the heating power for zone 2 is about ten times smaller than for zone 1 (not shown). As we will see later, this will complicate the estimation of the dynamics of zone 2.

Single-zone models

In contrast to experiment 1, we aim for a two-zone model. There will be no boundary condition so we will be able to predict the temperature for both zones simultaneously when only their heating power and the weather conditions are known. However, in order to get a grip on the dynamics and the orders of magnitude for the parameters, we first try to identify two single-zone models with the temperature of the other zone as a boundary condition. This will also provide useful indications regarding the order of magnitude of the *SE* of the two-zone model.

Without describing all steps to create the models we briefly discuss the results for the single-zone models. For both zones we have found a good fit with model B of Figure 5 (with an additional resistance R_{Bou} between T_{Zon} and the boundary temperature of the other zone). It is a second-order model with 8 parameters

of which two are initial temperatures. Each zone has a single window connected to the radiation on a vertical, south oriented plane (instead of the global horizontal radiation as indicated in the scheme of model B).

Table 4 gives the resulting parameter estimates for both models. The corresponding RMSE values are given in Table 5. The normalised confidence intervals are shown in Figure 11. From these results we can see the following:

- for all parameters, the order of magnitude is roughly the same for both zones;
- the thermal resistance of the boundary is a factor 5 higher when estimated from zone 2;
- zone 2 has a much better $RMSE_{auto}$ than zone 1, but a higher $RMSE_{cross}$;
- the confidence intervals for zone 2 are much larger, except for R_{Bou} and R_{Wal};
- both zones have higher solar aperture areas than for experiment 1. This makes sense considering that in experiment 1 the blinds were closed, and in experiment 2 they are open.

The low $RMSE_{auto}$ for zone 2 is misleading. Both the confidence intervals and cross-validation show that the model for zone 2 is not very good. We can understand this result by analysing the measurement data as shown in Figure 10. The temperature in zone 2 is extremely flat during the identification period. Therefore, the thermal inertia in this zone is not excited and consequently it is very hard or even impossible to estimate the time constants and other parameters of a dynamic model. We can conclude that poor datasets (with little excitation of the states) cause difficulties for the identification of dynamic models. Whenever possible, the building control system should cause sufficient excitation of all building components during the identification period. This conclusion has been formulated previously in literature, amongst others by Sourbron et al. (2013) and Žáčeková et al. (2014).

We now try to identify a two-zone model by combining the two single-zone models.

Two-zone model

The two-zone model has two outputs on which the simulation error SE will be computed according to (5): T_{Zon1} and T_{Zon2}. We take both weight factors $w_j = 1$.

For the model of the boundary between both zones two options are explored: a thermal resistance, or a wall composed of two resistances and a capacity. In principle, the parameter values should not deviate

Parameter	Zon1	Zon2	Unit
C_{Int}	1.7e7	2.7e7	J/K
C_{Zon}	2.6e6	1.2e7	J/K
$T_{Int}(0)$	26.4	22.7	°C
$T_{Zon}(0)$	29.5	22.5	°C
R_{Bou}	6.2e-3	2.9e-2	K/W
R_{Int}	1.3e-3	6.0e-4	K/W
R_{Wal}	3.9e-2	4.8e-2	K/W
gA	3.1	0.28	m^2

Table 4: Overview of estimated parameters for the accepted single-zone models for experiment 2.

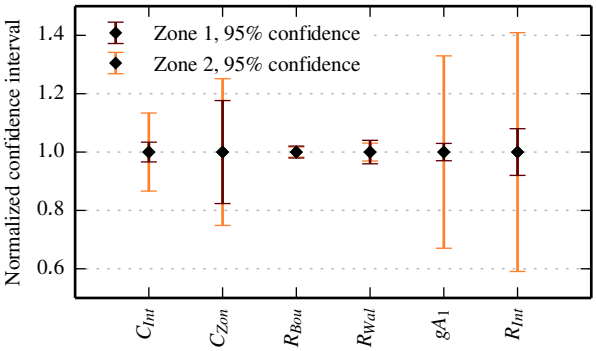

Figure 11: Normalised confidence intervals for the estimated parameters for both single-zone models.

	$RMSE_{auto}$	$RMSE_{cross}$
Zone 1	0.27 K	0.51 K
Zone 2	0.07 K	0.65 K
SE	0.34 K	1.16 K

Table 5: RMSE of individual zone models and resulting SE.

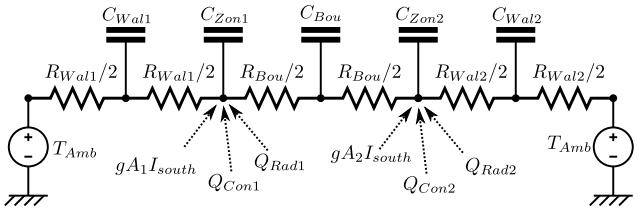

Figure 12: Accepted two-zone model for experiment 2.

much from the ones in Table 4. We also expect the largest parameter deviations for the parameters with the largest confidence intervals.

Without an additional state in the boundary wall between the zones, the results are not very good: the model has an SE_{auto} of 0.37 K and SE_{cross} of 1.74 K. Moreover, the initial temperature of C_{Int} for zone 2 lies at the boundary of 270 K (-3.1 °C) and rises monotonically during the identification period, thus falsifying the energy balance.

With an additional state C_{Bou} as shown in Figure 12, the simulation performance improves. However, analysis of the estimated parameters reveals again an initial temperature $T_{Int}(0)$ of 270 K. This time however, the capacity C_{Int} is not very large, and an attempt to narrow down the feasible region for the initial temperature leads to a valid model. The SE_{auto} becomes 0.335 K and SE_{cross} drops to 1.65 K. We should not be surprised that SE_{auto} drops slightly below the level of 0.343 K obtained with the two single-zone models: an additional degree of freedom is introduced with C_{Bou}.

Further attempts to improve the model were not successful, the accepted model is discussed in more detail below.

Model validation

Table 6 shows the resulting parameter estimates for the accepted model presented in Figure 12. All parameters have physical values. For zone 1, the parameters barely shift compared to the single-zone model. For zone 2 however, most parameters change with a factor of $\pm$ 10. This again indicates that the identification dataset is worse for zone 2.

When the *SE* is split in the RMSE values for each zone separately, Table 7 is obtained. Zone 1 has a very good performance, also in cross-validation. Zone 2 however has a bad $RMSE_{cross}$. By comparison of Table 5 and Table 7, we see that the single-zone model for Zone 2 has a better prediction performance than the two-zone model. If we were more interested in predicting zone 2 than zone 1, we need to increase the weighting factor w_2 from Eq. (5).

Parameter	Zon1	Zon2	Unit
C_{Int}	1.9e7	1.1e8	J/K
C_{Zon}	2.8e6	5.5e8	J/K
$T_{Int}(0)$	26.8	23.9	°C
$T_{Zon}(0)$	29.2	22.5	°C
R_{Int}	1.4e-3	4.0e-5	K/W
R_{Wal}	2.0e-2	5.9e-3	K/W
gA	2.4	18.7	m^2
C_{Bou}	4.0e9		J/K
R_{Bou}	1.7e-2		K/W
$T_{Bou}(0)$	25.3		°C

Table 6: Overview of estimated parameters for the accepted two-zone model for experiment 2.

	$RMSE_{auto}$	$RMSE_{cross}$
Zone 1	0.23 K	0.51 K
Zone 2	0.10 K	1.14 K
SE	0.33 K	1.65 K

Table 7: RMSE and SE of accepted two-zone model.

The bad simulation performance of zone 2 also becomes evident when comparing the measured and simulated zone temperatures. These are shown in Figures 13 and 14. For zone 1 however, the prediction performance is very good, despite the deviation on zone 2. Again, we can stress the importance of online identification and state estimation in order to avoid recurring model bias.

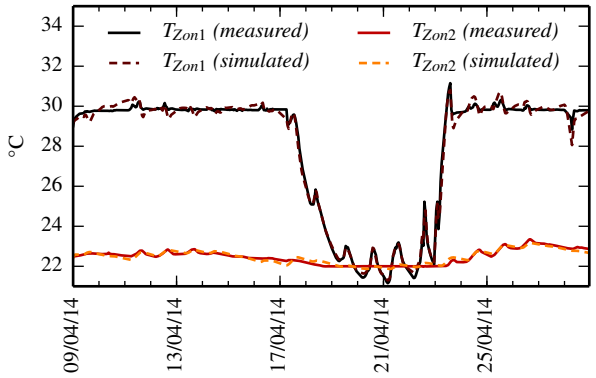

Figure 13: Measured and simulated temperature of both zones for the identification dataset (auto-validation) in the accepted model for experiment 2.

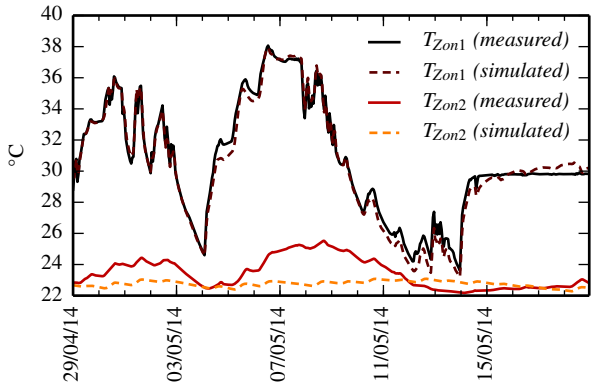

Figure 14: Measured and simulated temperature of both zones for the validation dataset (cross-validation) in the accepted model for experiment 2.

Conclusion

Inverse modelling is gaining attention in the building simulation community. More specifically grey-box modelling is considered as a strong framework for the creation of low-order models for analysis and control of monitored buildings. The first part of this paper presents an approach to obtain grey-box models in a largely automated way, which are applicable in both MPC and forecasting.

The first step is the creation of a building library with many potential model candidates. The Modelica package *FastBuildings* contains low-order models for thermal zones, HVAC, users, single and multi-zone buildings.

Next, a toolbox is presented that largely automates the parameter estimation of the *FastBuildings* models. It is implemented as a Python module that wraps the functionality of *JModelica.org* and presents the user a high-level interface for all common operations. The use of a gradient-based method allows an efficient numerical solution of the parameter estimation problems. Specific attention is paid to robustness and ease-of-use. A Latin hypercube sampling of the parameter search space overcomes local-minima issues related to the non-convexity of the optimisation problem. The toolbox is not publicly available, but can be obtained with an open-source license for research purposes by contacting the authors.

The toolbox is validated on two datasets generated by the detailed monitoring of a single-family house near Munich, Germany. In experiment 1, a single-zone building is identified that has a very good prediction performance. In an open-loop simulation over 20 days on the cross-validation dataset, the model deviations are very small with an $RMSE_{cross}$ of only 0.33 K.

In experiment 2, a two-zone building is identified with mixed performance. For the first zone, a good prediction performance is achieved with an $RMSE_{cross}$ of 0.51 K. The second zone however has more difficulties. Due to a weak excitation in the identification dataset, an $RMSE_{cross}$ of 1.14 K is obtained. This clearly points out the need of good identification data.

Finally, we want to point out two advantages of the proposed methodology that come from the use of Modelica. Firstly, the grey-box model is equation-based. This means that we have an acausal model relating all variables with equations, as opposed to an input-output model with a predefined information flow direction. Therefore, inputs and outputs can be switched as long as the problem is balanced. For example, given a temperature set point, the grey-box model would predict the heating load. Secondly, Modelica creates a large freedom in the model formulation by allowing also non-linear model components. These are typically encountered in heat transfer coefficients and HVAC equipment. Future developments of the grey-box toolbox and the *FastBuildings* library will explore these options.

Nomenclature

Symbol	Meaning
T	Temperature
C	Thermal capacity
R	Thermal resistance
Q	Thermal flux
gA	Solar admittance
I	Solar radiation

Subscript	Meaning
Zon	Zone (mostly denoting air)
Int	Internal
Wal	Walls / building envelope
Emb	Embedded (heating or cooling) system
Inf	Infiltration
Bou	Boundary
Amb	Ambient (outdoor)
Con	Convective
Rad	Radiative

Acknowledgement

Roel De Coninck wishes to thank Innoviris (Region of Brussels) for supporting this work on the side of 3E via

the ITEA2 project *Enerficiency* (contract RBC/11 DS 143a) and the European commission for supporting his work on behalf of KU Leuven by supporting the FP7 project *PerformancePlus* (contract nb. 308991). Fredrik Magnusson acknowledges support from the Swedish Research Council through the LCCC Linnaeus Center and is also a member of the ELLIIT Excellence Center at Lund University.

References

Åkesson, J. (2008). Optimica—an extension of modelica supporting dynamic optimization. In *Proc. 6th International Modelica Conference 2008*.

Åkesson, J., K.-E. Årzén, M. Gäfvert, T. Bergdahl, and H. Tummescheit (2010, November). Modeling and optimization with Optimica and JModelica.org—languages and tools for solving large-scale dynamic optimization problems. *Computers and Chemical Engineering 34*(11), 1737–1749.

Andersson, J., J. Åkesson, and M. Diehl (2012). CasADi – A symbolic package for automatic differentiation and optimal control. In S. Forth, P. Hovland, E. Phipps, J. Utke, and A. Walther (Eds.), *Recent Advances in Algorithmic Differentiation*, Lecture Notes in Computational Science and Engineering, Berlin. Springer.

Bacher, P. and H. Madsen (2011, February). Identifying suitable models for the heat dynamics of buildings. *Energy and Buildings 43*(7), 1511–1522.

Baetens, R., R. De Coninck, J. Van Roy, B. Verbruggen, J. Driesen, L. Helsen, and D. Saelens (2012). Assessing electrical bottlenecks at feeder level for residential net zero-energy buildings by integrated system simulation. *Applied Energy* ((Special issue on Smart Grids, Renewable Energy Integration, and Climate Change Mitigation - Future Electric Energy Systems)).

Biegler, L. T. (2010). *Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes*. MOS-SIAM Series on Optimization. Mathematical Optimization Society and the Society for Industrial and Applied Mathematics.

Blochwitz, T., M. Otter, M. Arnold, C. Bausch, C. Clauß, H. Elmqvist, A. Junghanns, J. Mauss, M. Monteiro, T. Neidhold, et al. (2011, March). The functional mockup interface for tool independent exchange of simulation models. In *"8th International Modelica Conference"*, Dresden, Germany, pp. 20–22.

Bohlin, T. (1995). Editorial - Special issue on grey box modelling. *International Journal of Adaptive Control and Signal Processing 9*, 461–464.

Crawley, D. B., J. W. Hand, M. Kummert, and B. T. Griffith (2008, April). Contrasting the capabilities of building energy performance simulation programs. *Building and Environment 43*(4), 661–673.

Davies, M. G. (2004). Simple Models for Room Response. In *Building Heat Transfer*, Chapter 14, pp. 311–334. Chichester: John Wiley & Sons, Ltd.

Elmqvist, H. (1997). Modelica - A unified object-oriented language for physical systems modeling. *Simulation Practice and Theory 5*(6), 32.

Englezos, P. and N. Kalogerakis (2000, October). *Applied Parameter Estimation for Chemical Engineers*, Volume 81 of *Chemical Industries*. CRC Press.

Henze, G. P. (2013, May). Model predictive control for buildings: a quantum leap? *Journal of Building Performance Simulation 6*(3), 157–158.

Henze, G. P. and C. Neumann (2011). Building simulation in building automation systems. In J. L. M. Hensen and R. Lamberts (Eds.), *Building performance simulation for design and operation*, pp. 401–440.

HSL (2013). A collection of Fortran codes for large scale scientific computation. http://www.hsl.rl.ac.uk.

Kersken, M., I. Heusler, and P. Strachan (2014, September). Erstellung eines neuen, messdatengestützten validierungs-szenarios für gebäude-simulationsprogramme. In *Christoph van Treeck, Dirk Müller (eds.) Proceedings of BauSIM 2014*, Aachen, Germany, pp. 144–151. IBPSA.

Kristensen, N. R., H. Madsen, and S. B. Jorgensen (2004, February). Parameter estimation in stochastic grey-box models. *Automatica 40*(2), 225–237.

KU Leuven and 3E (2014). open-IDEAS source code repository. https://github.com/open-ideas.

Madsen, H. and J. Holst (1995). Estimation of continuous-time models for the heat dynamics of a building. *Energy and Buildings 22*, 67–79.

Magnusson, F. and J. Åkesson (2012, September). Collocation methods for optimization in a Modelica environment. In *9th International Modelica Conference*, Munich, Germany.

Reynders, G., J. Diriken, and D. Saelens (2014, October). Quality of grey-box models and identified parameters as function of the accuracy of input and observation signals. *Energy and Buildings 82*, 263–274.

Sourbron, M., C. Verhelst, and L. Helsen (2013, May). Building models for model predictive control of office buildings with concrete core activation. *Journal of Building Performance Simulation 6*(3), 175–198.

Vande Cavey, M., R. De Coninck, and L. Helsen (2014). Setting up a framework for model predictive control with moving horizon state estimation using JModelica. In *10th International Modelica Conference*, Lund, Sweden, pp. 1295–1303.

Wetter, M. (2009, June). Modelica-based modelling and simulation to support research and development in building energy and control systems. *Journal of Building Performance Simulation 2*(2), 143–161.

Wetter, M. (2011). A view on future building system modeling and simulation. In J. L. M. Hensen and R. Lamberts (Eds.), *Building performance simulation for design and operation*, pp. 28.

Wetter, M. and C. Van Treeck (2013). IEA EBC Annex 60 - New generation computational tools for building and community energy systems based on the Modelica and Functional Mockup Interface standards. http://iea-annex60.org/about.html.

Wächter, A. and L. T. Biegler (2006). On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming 106*(1), 25–57.

Žáčeková, E., Z. Váňa, and J. Cigler (2014, December). Towards the real-life implementation of MPC for an office building: Identification issues. *Applied Energy 135*, 53–62.