Parameter Estimation of Modelica Building Models Using CasADi

Carlos Durán Cañas[1] Javier Arroyo[1] Joris Gillis[1,2] Lieve Helsen[1,3]

[1]Department of Mechanical Engineering, KU Leuven, Heverlee, Belgium
carlosandres.durancanas@student.kuleuven.be,
{javier.arroyo,joris.gillis,lieve.helsen}@kuleuven.be
[2]Flanders Make@KU Leuven
[3]EnergyVille, Thor Park, Waterschei, Belgium

Abstract

Predictive control can substantially improve the energy performance of buildings during operation, but it requires a model of the building to be implemented. Gray-box model identification starts from a physics-based model (white-box element) and complements it with measurements from the operation of the building (black-box element). The level of detail of the original model is limited by the optimization problem that needs to be solved when estimating its parameters. Consequently, it is common to heavily simplify building models hindering the intelligibility of their parameters and limiting their application potential. This paper investigates the accuracy and scalability of different transcription methods for parameter estimation of building models. The methodology starts from a Modelica model as an initial guess which is transferred to CasADi using the Functional Mockup Interface to solve the parameter estimation problem. The study demonstrates the high effectiveness of multiple shooting. Single shooting and direct collocation could be more suitable for setups with faster integration times or with increased granularity in the training data, respectively.
Keywords: Gray-Box modeling, CasADi, Shooting Methods, Direct Collocation, OpenModelica

1 Introduction

Commercial and domestic buildings worldwide account for 30-45% of the global energy use (Mariano-Hernández et al. 2021). Therefore, efficient building energy use is a topic of growing interest. A popular strategy for building energy management systems (BEMS) is the use of model predictive control (MPC) (Drgoňa et al. 2020). MPC uses a building model and an optimizer to minimize a cost function generally comprised of two competing elements like occupant thermal discomfort and operational cost.

Gray-box model identification is a powerful tool for obtaining building models for predictive control through two input sources: *prior system knowledge* and *operational data* which are referred to as the white and black elements of a gray model, respectively (Bohlin 2006). This approach benefits from the advantages of both physics- and data-based modeling by calibrating the physical model parameters with operational data extracted from the actual building. Moreover, gray-box modeling can automatically and systematically tune the parameters for a model while having the reliability offered by physics (Bohlin 2006).

Prior knowledge can be introduced with well-known equations describing physical phenomena in buildings like heat transfer and thermal inertia. Other inputs like weather variables and internal gains are subject to uncertainty and need to be estimated with forecasting models. It is also crucial to obtain accurate values for parameters such as heat conductivities of materials, solar transmittance of the glazing in the windows, etc. However, obtaining detailed information on the thermal systems for the building is a difficult and time-consuming process in practice (Yu et al. 2019). Optimization is required to estimate the parameter values that minimize a predefined error function, which usually leads to a non-linear and non-convex problem that needs to be solved. Traditionally, this problem is solved by lumping the building parameters into basic models usually represented by a lower-order RC (resistance-capacitance) network. Some examples include (Beneventi et al. 2012), (Drgoňa et al. 2020) and (Saurav and Chandan 2017). These simplifications can decrease the usability and intelligibility of the building model and hamper the ability of the model to be used for fault detection and diagnostics mechanisms in the context of predictive maintenance.

Another approach to estimate the model's parameters is through black-box optimizers like brute-force or genetic algorithms. However, these methods can have scaling issues when increasing the number of parameters to be estimated. Since building models can have several parameters to calibrate, traditional gradient-based optimization methods are preferred for the envisaged application. Moreover, unlike off-line parameter estimations, efficient non-linear problem solvers are key for the application of on-line parameter estimations such as the ones involved in adaptive control. For these reasons, the use of efficient, gradient-based, parameter estimation methods is crucial in order to develop more advanced building models.

This work investigates the performance of different transcription methods for parameter estimation of building models by coupling two open-source toolboxes: OpenModelica (Fritzson et al. 2005) and CasADi (Andersson et al. 2019). The former is a tool for modeling physical systems using the Modelica language and the latter is

a numerical optimization and algorithmic differentiation framework designed to solve highly complex optimization problems. Similar workflows were adopted by (Shitahun et al. 2013) and by (Decker 2021), but they were using deprecated versions of the software and their focus was not on parameter estimation of building models. Criteria such as accuracy, scalability, convergence time, and computational cost are investigated in this work. These criteria are then used to compare all investigated methods as well as to determine their optimal application range.

The outline of the paper is as follows: Section 2 gives theoretical background related to this work; Section 3 elaborates on the methodology used in this work to implement and compare different transcription methods for parameter estimation; Section 4 presents the results obtained from the implementation of each parameter estimation method. Finally, Section 5 draws the main conclusions and Section 6 suggests lines of further research.

2 Theoretical Background

2.1 Parameter Estimation Problem

The process of parameter estimation is a crucial step when configuring a building model. The process of calibrating the parameters of a physical model can be expressed mathematically as shown in Eqs. 1[1] The objective function to be minimized $f(\boldsymbol{p})$ is expressed as a (non-linear) least squares problem by means of the Eucledian (ℓ-2) norm for the error between the outputs from the physical model $\boldsymbol{y}(\boldsymbol{p},t)$ and the historical measurements $\hat{\boldsymbol{y}}(t)$ within a time horizon $t \in [t_0, t_T]$. Here, the error function is squared to achieve the sum of the squared differences. This in turn is divided by two as a convention in order to remove constants during the calculation of its derivative. The optimization is subject to equality and inequality constraints ($\boldsymbol{h}(\boldsymbol{p})$ and $\boldsymbol{g}(\boldsymbol{p})$, respectively) which are derived from the physics and limits of the real system, represented by the physical model. Here, the objective function as well as the equality and inequality constraints are dependent on the model parameter values $\boldsymbol{p} \in \mathbb{R}^m$, which must be tuned to minimize the residual between the model and measurements.

$$\min_{\boldsymbol{p}} f(\boldsymbol{p}) = \int_{t_0}^{t_T} \frac{1}{2} \left(\|\boldsymbol{y}(\boldsymbol{p},t) - \hat{\boldsymbol{y}}(t)\|_2 \right)^2 dt \tag{1a}$$

$$\text{subject to: } \boldsymbol{h}(\boldsymbol{p}) = 0 \tag{1b}$$

$$\boldsymbol{g}(\boldsymbol{p}) \leq 0, \tag{1c}$$

Parameter estimation problems for building models are usually non-convex because of the multiplication of thermal resistances and capacitances that are commonly optimization variables. This non-convexity dictates the importance of the initial parameter guesses $\boldsymbol{p}_0$ with minimum and maximum values $\boldsymbol{p}_{min}$ and $\boldsymbol{p}_{max}$, respectively. For the case of a convex problem, initial parameter guesses become trivial since, by definition, all local minima in a convex problem are the global minimum (Wright, Nocedal, et al. 1999). The solution of a non-convex problem is most likely a local minimum which is accepted in practice due to the high complexity involved in the calculation and (in some cases) the non-physicality of a global minimum. For this reason, accurate initial guesses for the parameter values are paramount to obtaining a physical local minimum. Figure 1 depicts the results of a bad initial guess for an estimation problem with a single parameter. The importance of an accurate initial guess becomes clear even in this case with a single parameter for which all initial guesses lead to different local minima, none of which are the global minimum.

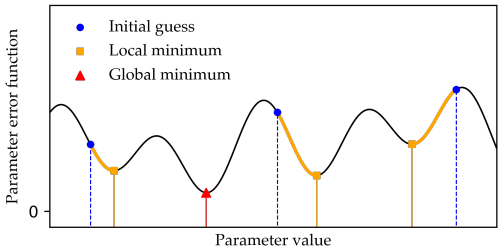

Figure 1. Non-convex minimization problem dependence on initial parameter guess.

2.2 Transcription Methods

Transcription methods are responsible for discretizing the originally continuous parameter estimation problem into a discrete non-linear program (NLP) in the form of Eqs. 2.

$$\min_{\boldsymbol{p}} \sum_{i=1}^{N_T} \frac{1}{2} \left(\|\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i\|_2 \right)^2 \tag{2a}$$

$$\text{subject to: } h_i = 0 \tag{2b}$$

$$g_i \leq 0, \tag{2c}$$

for $i = 1, \ldots, N_T$ with N_T being the instance at the end of the time horizon. Once this transcription takes place, NLP solvers are used to minimize the objective function and output the optimal parameter values (M. P. Kelly 2015). The transcription method used to discretize the problem is critical to the complexity and the outcome of a parameter estimation problem since it dictates the number of decision variables and the sparsity of the eventual NLP to be solved. Multiple algorithms exist for this process, yet they can all be classified into two main groups: *shooting methods* and *collocation methods*.

2.3 Shooting Methods

Shooting methods take states as decision variables and integrate over a set of intervals, approximating the function

[1] Vectors and matrices are expressed in bold while scalars are expressed in regular text ($\boldsymbol{y}$ vs y).

DOI 10.3384/ecp204301

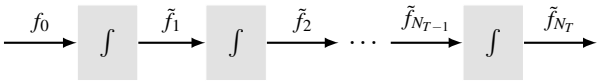

Figure 2. Block diagram for a single shooting transcription algorithm.

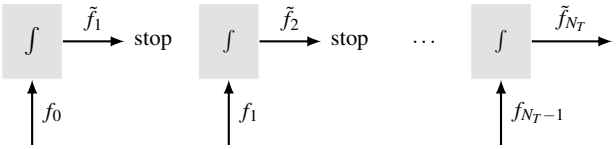

Figure 3. Block diagram for a multiple shooting transcription algorithm.

dynamics along the integration path based on given constraints. The term shooting methods refers to their resemblance to the operation of a projectile deployment device such as a cannon. In this example, the decision variables are only known in the first instance (e.g. firing angle and power), while the trajectory is subject to the projectile dynamics and physical constraints.

The simplest variation of the shooting algorithms is the single shooting method. Single shooting works by taking only the very first state as a decision variable which is used to determine a prediction for the following state. Subsequently, this prediction is used for the integration of the following state, starting a chain reaction of predictions until the integration horizon is reached (Schittkowski 2002). Figure 2 shows a block diagram demonstrating the integration process in a single shooting algorithm with f_i being decision variables and $\tilde{f}_i$ being predictions. Here, it can be seen that for every integration step, the previous prediction is taken as an input, generating the following prediction until the instance N_T is reached.

The main advantages of a single shooting algorithm stem from its simplicity and the compact representation of the eventual NLP. They are suitable for simple differential-algebraic equations (DAE) where extremely good initial guesses can be provided. Nevertheless, they may pose convergence problems for complex systems because of the need to integrate over the entire time horizon for every iteration of the optimization (Michalik, Hannemann, and Marquardt 2009).

Multiple shooting algorithms operate in a very similar way to single shooting algorithms but break down the time horizon into multiple intervals. Instead of taking a single decision variable, multiple shooting methods take a decision variable for every time step within the integration horizon, making a single integration over that time step (M. P. Kelly 2015). As the segments become shorter, the integration paths tend to become linear. Additionally, since each step is not dependent on the previous step, each integration can be computed in parallel, leading to shorter integration times. Since each prediction step does not perfectly match the decision variable for the following segment, the difference (known as the *defect*) must be stated in the constraint equations leading to larger and sparser programs. The increase in the number of constraints can increase the total computational time (M. Kelly 2017). A block diagram for this process is visualized in Figure 3. The use of single and multiple shooting methods is highly dependent on the application case and its level of complexity. Figure 4 shows an example of both shooting methods being applied to a parameter estimation problem.

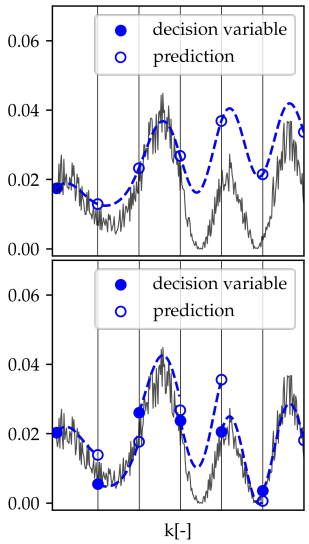

Figure 4. Example for a single (top) and multiple (bottom) shooting algorithm applied to the transcription of an error function.

2.4 Collocation Methods

The basis of collocation methods is the use of spline functions made up of polynomial sequences. The motivation is the effortless derivation and integration of these spline functions as well as their capability to be easily expressed in terms of coefficients (M. Kelly 2017). The integration path followed for each segment in a direct collocation algorithm is determined by two factors: the desired order for the polynomial to be fitted and the slope of the function at the collocation points (Bellomo et al. 2007). Depending on the desired order for the polynomial fit n, $n-1$ collocation points must be placed within the segment. Once these collocation points are determined, the algorithm adjusts the values for them such that a spline function going through the initial decision variable matches the slope of the function at each collocation point. If, like in most applications for collocation methods, this algorithm is applied within a multiple shooting framework, this procedure is completed for multiple segments covering the entire integration horizon. Similarly to multiple shooting, this process can be realized in parallel, yielding a series of predictions and their corresponding defects which must be accounted for in the constraint equations of the NLP.

Two approaches are widely used with the goal of reducing the defects between the predictions and decision variables. These are referred to as mesh refinement pro-

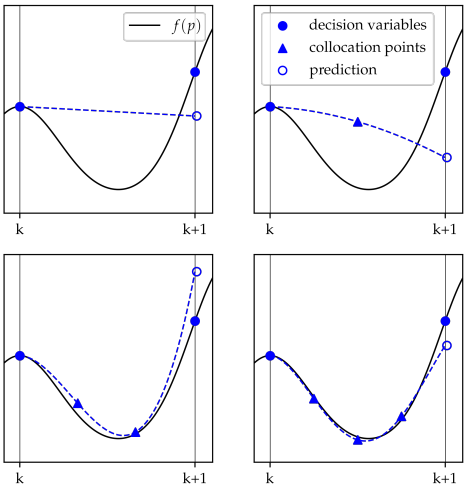

Figure 5. Example for a p-refinement procedure on direct collocation for polynomial orders ranging from first (top left) to fourth (bottom right) order.

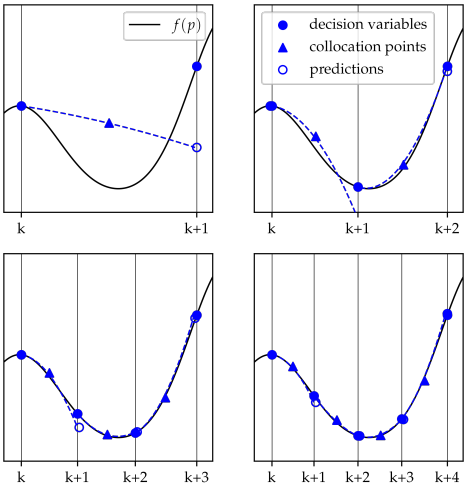

Figure 6. Example for an h-refinement procedure on direct collocation using second degree polynomials.

cedures, namely p- and h-refinement. A *p-refinement* procedure increases the order of the polynomial for each collocation segment such that the integration path can better follow the trajectory dynamics, ultimately resulting in a better prediction. An *h-refinement* procedure reduces the segment length such that the functions become more linear.

Figure 5 shows an example of a p-refinement procedure with varying polynomial orders. Here, a focused view on a single segment (k to $k+1$) is shown for polynomial orders ranging from first to fourth order for a direct collocation method on a function $f(p)$. It is clear that the defect decreases and the dynamics are better represented as the polynomial order increases. Similar to Figure 5, Figure 6 shows an h-refinement procedure for varying segment lengths for a function $f(p)$ using a second-order direct collocation method. A noticeable decrease in the prediction defect can be seen as the segment lengths become smaller.

3 Methods

The workflow followed in this work is hosted in the following open-source repository under a BSD license:

```
https://gitlab.kuleuven.be/positive-energy-districts/mocaspy
```

The necessary steps to generate the original physics-based model are illustrated with white blocks in Figure 7. These white blocks of Figure 7 represent the steps handling the model with true system dynamics. The original model is reconfigured into a so-called wrapped model that redeclares the parameters as inputs to tune its values during the optimization. Multiple open-source Modelica libraries exist for building modeling that can be used to configure the original building model such as IDEAS (Jorissen et al. 2018) and Buildings (M. Wetter et al. 2014). OpenModelica is used to compile a Functional Mockup Unit (FMU) to later transfer the model into CasADi.

The data collection process is illustrated by the black blocks shown in Figure 7. In practice, the data needed for calibration would be directly gathered from the actual building. However, in this work, OpenModelica is also used to emulate operational data. That is, we use the same model to generate the data as the model that is later used for parameter estimation. In this way, the true parameter values are known, and the accuracy of the parameter estimation process can be measured for a given deviation that is artificially introduced. This provides a hermetic environment for the investigation which would be unachievable in a real setting. Measurements of interest comprise weather and building variables such as the ambient temperature $\hat{T}_{amb}$ and the total heat input into the thermal zone $\hat{Q}_{hea,coo}$. Finally, the recorded operative zone temperature $\hat{T}_{zon}$ is used as the target variable in Eqs. 1. The training period t_T is set to one week (604800 seconds) with a sampling time t_s of 30 seconds, leading to 20160 samples in total.

Once the model FMU and the operational data are ready, the parameter estimation problem is formulated with the CasADi framework. This process is shown by the gray blocks of Figure 7. CasADi's `DaeBuilder` class was recently extended to import Model Exchange FMUs of version 2.0 (Andersson 2023 submitted). From Eqs. 1, the parameter estimation problem is relaxed to Eqs. 3 where a slack variable S is introduced to use soft constraints. The objective function minimizes the Euclidean norm of the error between the (virtually) measured $\hat{T}_{zon}$ and the modelled T_{zon} thermal zone temperatures.

$$\min_{\boldsymbol{p}} f(\boldsymbol{p}) = \int_{t_0}^{t_T} \left[\left(\left\| T_{zon}(\boldsymbol{p},t) - \hat{T}_{zon}(t) \right\|_2 \right)^2 + S \right] dt \quad (3a)$$

$$\text{subject to:} \quad \boldsymbol{p} \leq \boldsymbol{p}_{max} + S \quad (3b)$$

$$\boldsymbol{p} \geq \boldsymbol{p}_{min} - S \quad (3c)$$

$$\boldsymbol{S} \geq 0 \quad (3d)$$

DOI
10.3384/ecp204301

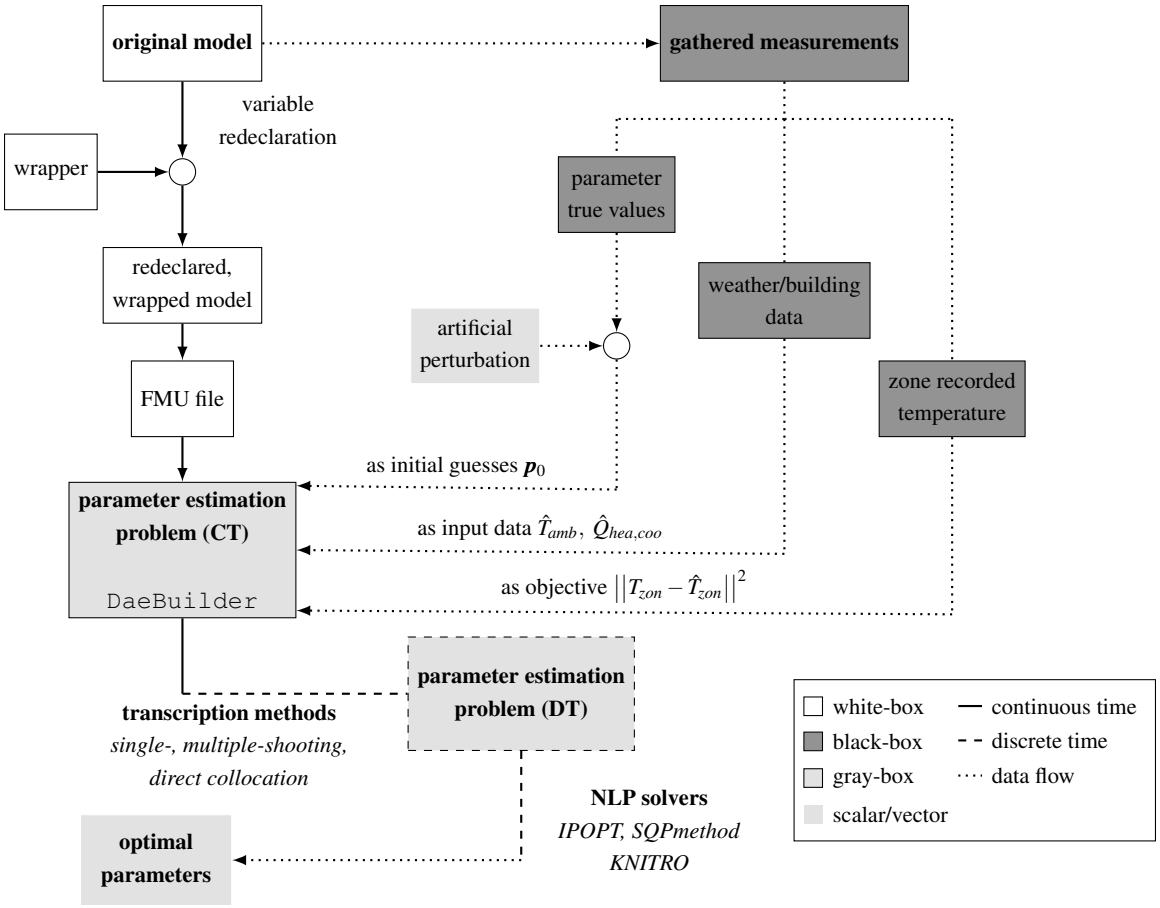

Figure 7. Parameter estimation workflow. The CasADi's `DaeBuilder` class is used to load the building model and to transcribe the continuous-time (CT) parameter estimation problem into a discrete-time (DT) problem that an NLP solver can address.

In Eqs. 3, $\boldsymbol{p}_{max}$ and $\boldsymbol{p}_{min}$ represent the upper and lower boundaries of the parameter vector $\boldsymbol{p}$, respectively.

The parameter estimation problem must be discretized by means of transcription methods. The Rockit framework (Gillis et al. 2020) is used here since it is built on top of CasADi and offers readily available formulations of different transcription methods. In this study, single shooting, multiple shooting, and collocation are investigated. Multiple variations of these algorithms are implemented to study the effect of h- and p-refinement. Upon completion of the problem transcription, the parameter estimation problem is solved by means of an NLP solver. IPOPT is chosen here due to its widely recognized reputation and capabilities for large-scale optimization. It is used in a setting with a limited-memory Hessian approximation. The results are compared based on three factors: accuracy for the parameter estimation, number of iterations, and the computational time required for convergence. These factors determine the application range for each transcription method based on the desired level of accuracy and available computational power.

The model `SimpleRoomOneElement` from the IDEAS library is chosen as a good trade-off between the level of detail and simplicity. This model represents a single thermal zone equipped with multiple, double-paned windows at different wall orientations (declared as `corGDouPan`), and is shown in Figure 8. Additionally, the model is equipped with a heater operating under a simple on/off control at a given time of the day (represented by the data table `intGai`). The weather data is simulated by the module `weaDat` and inputted into the building thermal zone, `thermalZoneOneElement`.

The parameters of the original model are redeclared as inputs in a so-called *wrapped* model to enable 1) their variability as decision variables in the CasADi `DaeBuilder` object, and 2) derivative information of model outputs. Moreover, some elements of the model had to be bypassed with inputs obtained from a previous simulation as the `DaeBuilder` class does not yet support time events.

Three studies are considered: the estimation of a single parameter, three parameters, and five parameters. Table 1 shows the parameters estimated during all studies along their units, true values $\boldsymbol{p}_{real}$, lower $\boldsymbol{p}_{min}$ and upper $\boldsymbol{p}_{max}$ limits, and their initial guesses after being artificially perturbed $\boldsymbol{p}_0$. Parameters that are commonly unknown, desired and/or hard to obtain in practice are selected to be estimated like U_{win} and $h_{con,win,out}$ which represent the transmission and convective coefficients associated with the installed windows, respectively. $h_{con,wall,out}$ represents

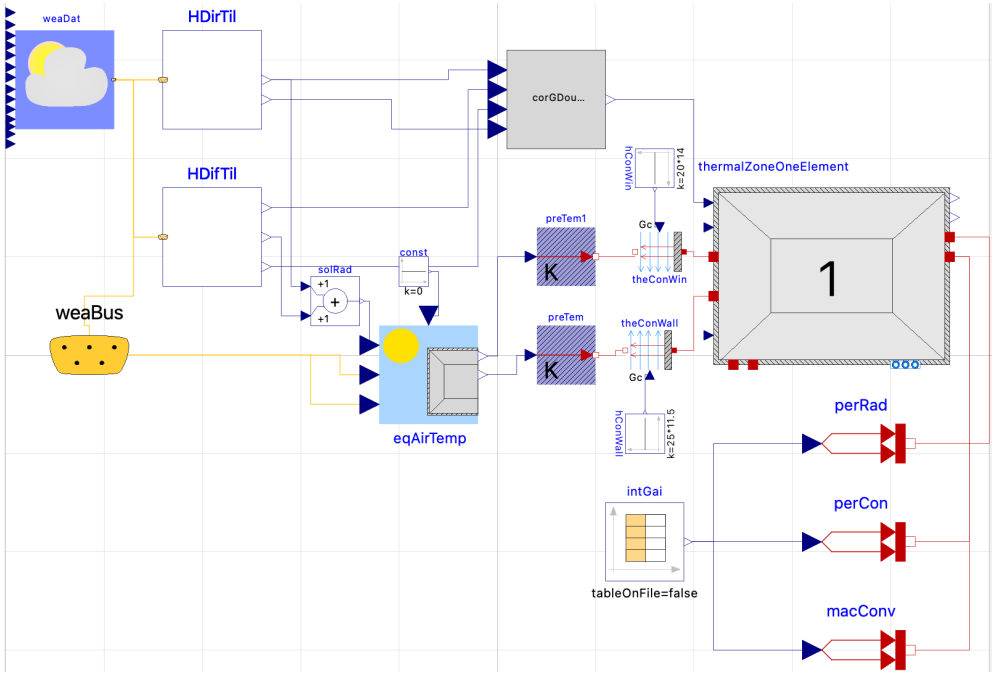

Figure 8. Graphical representation of `SimpleRoomOneElement.mo` model from IDEAS (Jorissen et al. 2018).

Table 1. Parameter values, boundaries and initial guesses for all studies conducted.

# Params.	$\boldsymbol{p}$	*Unit*	$\boldsymbol{p}_{real}$	$\boldsymbol{p}_{min}$	$\boldsymbol{p}_{max}$	$\boldsymbol{p}_0$
1	U_{win}	W/m^2K	2.1	1.9	2.4	6.3
3	$h_{con,win,out}$	W/m^2K	20	18	22	60
	$h_{con,wall,out}$	W/m^2K	20	18	22	60
5	h_{rad}	W/m^2K	5	4	6	15
	a_{ext}	-	0.7	0.5	0.9	1

the convective coefficient for the exterior walls. Finally, h_{rad} is the coefficient of radiative heat transfer for the zone walls, and a_{ext} represents the thermal absorption coefficient for the exterior walls. From Table 1 it can be seen that the first study estimates the window transmission coefficient, the second study includes the window and exterior wall convective coefficients and the final study incorporates h_{rad} and a_{ext}. For all instances, the initial guesses are shown. These are decided to be three times larger than the actual value for the parameter, except for a_{ext} since it is a percentage and has a maximum value of 1.

The number of steps per discretization interval M is decided through a sensitivity analysis for a simple model integration with respect to the results of a reference integrated with an arbitrarily high value of $M = 3000$ steps. A value of $M = 10$ is taken as a compromise between integration error and computational demand. Table 2 contains a detailed list of all variations conducted to compare the transcription methods for parameter estimation. Notably, each variation is implemented for all cases outlined in Table 1 with one, three, and five parameters, which results in a total of 60 optimizations being carried out.

Table 2. Conducted investigations for all considered test cases. First, the refinement scheme is studied in detail for multiple shooting and collocation schemes (Sections 4.1,4.2,4.3). Then, all three transcription methods are compared for specific refinement schemes (Section 4.4).

Method	*Refinement Scheme*		*Transcription methods*	
	h (N)	*p* (deg)	*h* (N)	*p* (deg)
Single Shooting	-	-	100 200	-
Multiple Shooting	100 200 300 400	-	100 200	-
Direct Collocation	100 200 300 400	2, 3, 4, 5, 6, 7, 8, 9	100 200	3

4 Results and Discussion

All optimizations were run on an Apple M1 MacBook Air (2020 version) with 8 GB of RAM and MacOS Ventura 13.0. A comparison between the thermal zone temperatures using the perturbed initial guess values and the estimated parameter values can be seen in Figure 9. All valid estimations, i.e. those optimization runs that converged, led to proper fitting and their solutions resulted in original parameter values within a tolerance of 10%, in many cases landing on the local minima at the parameter boundaries

DOI
10.3384/ecp204301

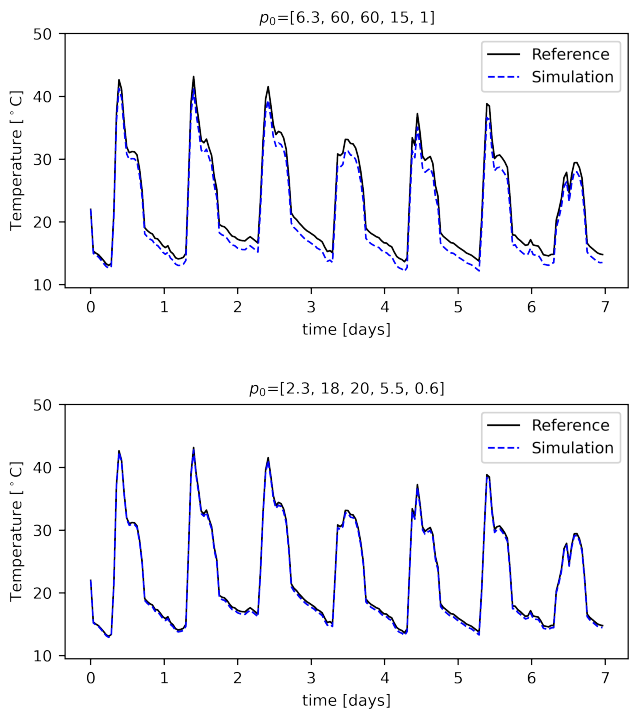

Figure 9. Temperature profiles of the model using the parameter initial guesses (top) and the estimated values (bottom) compared to the actual model temperature profile (Reference).

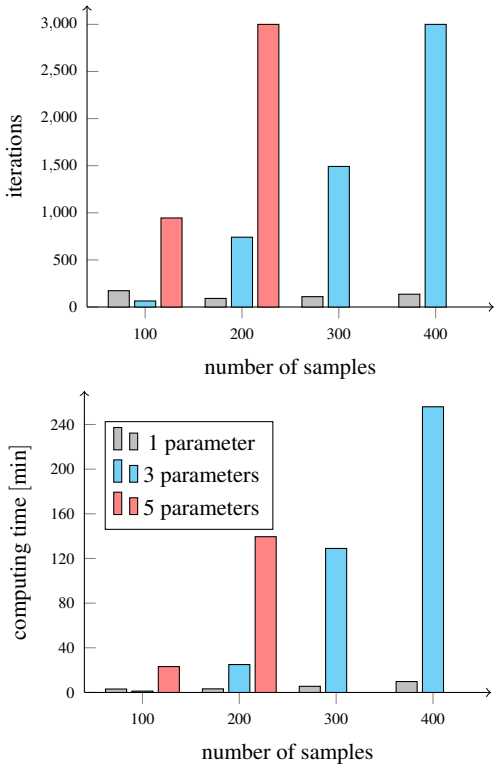

Figure 10. h-refinement investigation for a multiple shooting algorithm.

(see Table 1). Therefore, all achieved solutions are considered successful in terms of accuracy and the focus of this section is on the computational demand of obtaining these values.

4.1 Multiple Shooting h-refinement

The number of iterations and computing times for all variations to investigate h-refinement in multiple shooting are shown in Figure 10. The first aspect to highlight is that the multiple shooting algorithm succeeds to estimate the parameters for all cases with one, three, and five parameters. It is evident that the multiple shooting algorithm demonstrates remarkable suitability in the scenario of a single estimated parameter. It is able to estimate the chosen parameter without encountering any significant challenges despite the increasing estimation problem granularity. Its convergence times remain significantly low compared to the other cases with more parameters. The computational limits when increasing the problem granularity for the multiple shooting are reached in the cases involving three and five estimated parameters. Upon closer examination, it becomes apparent that the number of iterations and computational time for the three-parameter case with 100 samples closely resemble those of the single-parameter case. However, a steep increase is observed starting at the estimation with 200 samples, increasing quasi-exponentially until the iteration limit for IPOPT of 3000 iterations is reached for the case using 400 samples. Similar behavior is observed for the five-parameter case, although at a much earlier point. The computational time necessary for this scenario is approximately eight times larger than that of the single- and two-parameter cases when using a sample size of 100. Moreover, it rapidly reaches the iteration limit with a sample size of 200.

Considering that all estimations led to an accurate representation of the model, it is clear that an h-refinement for the multiple shooting algorithm is not necessary for the envisaged application. Increasing the granularity merely increases the computational resources required for the estimation without providing any additional value to the solution. However, the granularity choice should be made carefully since lower values can result in a loss of detail which can lead to an infeasible problem statement. Infeasible problem statements were encountered for sample sizes of 50, 85 and 90 for the single, three- and five-parameter cases, respectively.

4.2 Direct Collocation p-refinement

After exploring the h-refinement for multiple shooting, the impact of increasing the order for direct collocation is investigated. The sample size is set to 100, which represents the minimum value for a viable problem. Figure 11 presents the resulting number of iterations and computing times for all case studies. A slight increase in computational demand is observed for the three-parameter case around the collocation order of 9 and a sudden increase for the collocation order of 6, which can be qualified as an outlier. However, there is no clear correlation between the

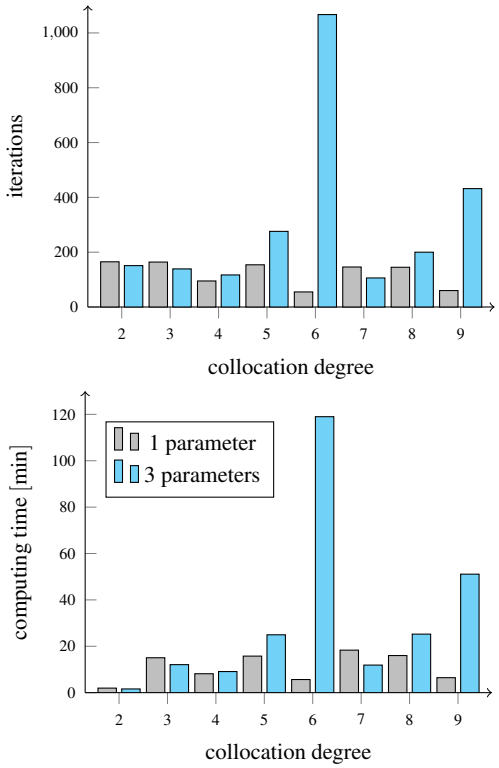

Figure 11. p-refinement investigation for a direct collocation algorithm with 100 samples.

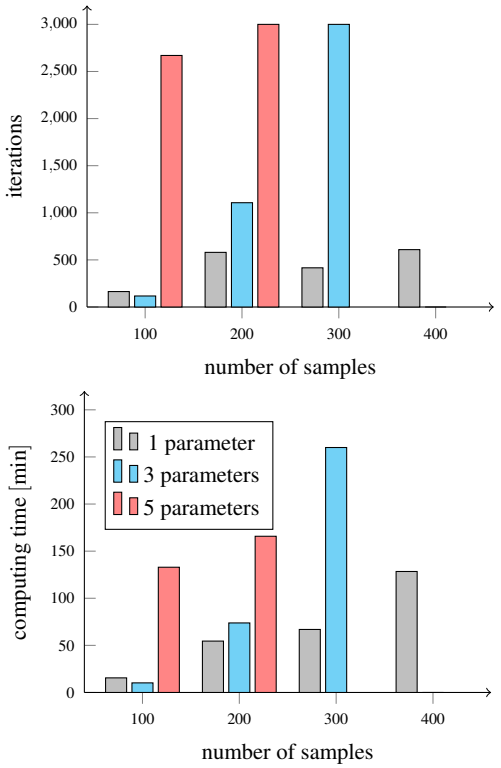

Figure 12. h-refinement investigation for a direct collocation algorithm with collocation degree of 3.

computational resources and the order of the collocation. As discussed in subsection 2.4, a higher collocation order results in a more accurate fit for the objective function, particularly for cases with lower granularity. The results from this study suggest that the collocation order is not affecting the computational strain thanks to a large enough granularity in the problem statement. Therefore, an increase in computational demand would be expected for a case with a higher number of samples.

4.3 Direct Collocation h-refinement

Similar to the multiple shooting case, an h-refinement investigation is conducted for a direct collocation algorithm. Here, a collocation degree of three is used based on the results of the p-refinement investigation. The results are shown in Figure 12. Similar patterns to those observed in the multiple shooting case emerge with a notable difference for the single parameter case. In comparison to the multiple shooting case, the estimation process using 400 samples takes approximately 13 times longer to converge, indicating a significantly higher computational strain for the direct collocation algorithm. Similar observations can be made when comparing the three- and five-parameter cases. While the overall trends are similar to the multiple shooting case, it is clear that the direct collocation algorithm experiences a higher computational burden. This is particularly evident in the three-parameter case, where the iteration limit is reached at 300 samples, compared to 400 samples in the multiple shooting case. In the five-parameter case, both studies reach the iteration limit at 200 samples. However, the computational strain is higher for direct collocation, as evidenced by the significantly longer computational time and more than twice the number of iterations required in the case with 100 samples.

Again, since all successful estimations yielded identical results, it is recommended to select a sample size that ensures proper convergence while minimizing the computational load. Excessively large sample sizes can lead to unnecessary waiting times or, in complex scenarios such as the five-parameter case, even convergence failure in the estimation process.

4.4 Transcription Method Comparison

Finally, a comparison of all transcription methods applied is carried out. Two sample sizes are investigated: one of 100 (shown in Figure 13) and another of 200 (shown in Figure 14). Single shooting, multiple shooting, and direct collocation with a collocation degree of three are compared for each sample size when estimating one, three, and five parameters. From examining the results in Figure 13 and Figure 14, several observations emerge.

Single Shooting This algorithm fails to converge to a solution for the five-parameter case. In the single- and three-parameter cases, it shows the fewest iterations required for convergence with the highest computing times for all scenarios. This could be seen as an advantage for a setup with more powerful computational resources. It is worth noting that the FMI simulations triggered by the

DOI
10.3384/ecp204301

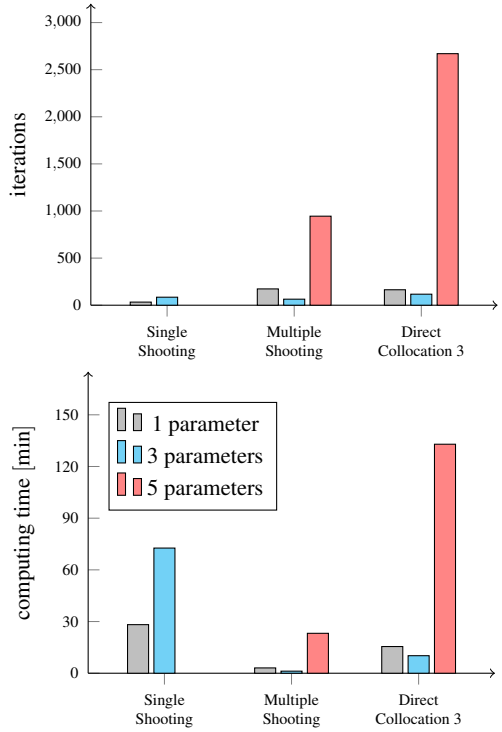

Figure 13. Transcription method performance comparison using 100 samples.

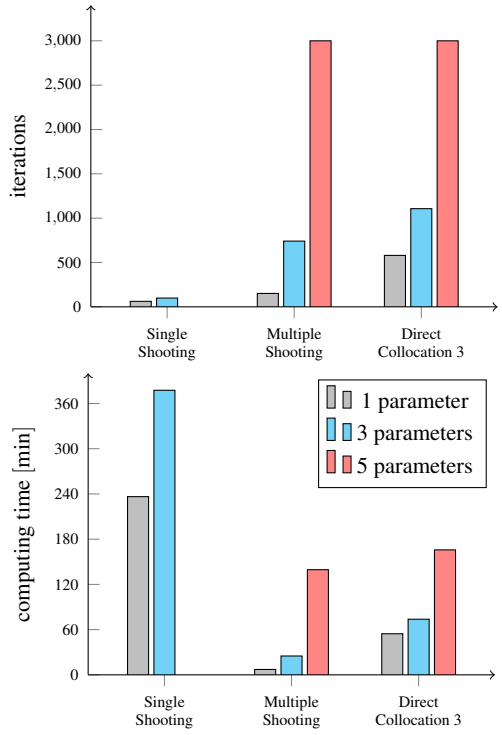

Figure 14. Transcription method performance comparison using 200 samples.

`DaeBuilder` object took more than 20 times longer than the calculations needed in each iteration for the optimization. Hence, single shooting could become competitive if the FMU integration times can be sped up.

Multiple Shooting Overall, multiple shooting shows the best performance for all investigated transcription methods. As shown for both sample sizes, it offers the lowest computing times for all cases, demonstrating its scalability potential. Moreover, when compared to its usual counterpart, direct collocation, multiple shooting requires significantly fewer iterations to achieve these fast convergence times, particularly evident in the investigation with 100 samples. Furthermore, if the efficiency of the FMI is improved, multiple shooting has the potential to achieve even better performance. The advantage of multiple shooting over direct collocation is further supported by the h-refinement investigation where multiple shooting achieved the same level of model granularity at a much lower computational cost. In general, multiple shooting is able to converge to accurate solutions for all investigated cases while requiring less tuning compared to single shooting and direct collocation.

Direct Collocation Similar to multiple shooting, direct collocation is able to converge to an optimal solution for all investigated cases though it has a higher computational burden. However, by comparing the results from both sample sizes (Figs 13 and 14) a decrease in the performance gap between both algorithms is observed. While multiple shooting still achieves convergence with a lower computational burden, it experiences a significant increase in both iterations and computing time when the granularity of the problem increases. This suggests that for higher sample sizes direct collocation could outperform multiple shooting. Additionally, direct collocation offers the highest level of tunability, which can be optimized for a specific application, perhaps leading to better performance. However, this tuning process is time-consuming and requires deep system knowledge. Overall, direct collocation algorithms exhibit great potential for rapid convergence, although meticulous fine-tuning is necessary to fully exploit their capabilities.

5 Conclusion

Predictive control can substantially enhance energy efficiency in buildings. To calibrate building models with operational data, efficient discretization methods are needed for the associated parameter estimation problem. The implemented methodology formulates the original building model in Modelica and transfers the model to CasADi through its `DaeBuilder` class, which relies on the Functional Mockup Interface. Multiple variations of transcription methods and the effect of different refinement schemes are investigated. The study demonstrates the high effectiveness of the multiple shooting algorithm for the envisaged application. Multiple shooting successfully converges for all cases investigated and shows the smallest

convergence time. Single shooting has the highest computing times but requires fewer iterations to converge, so it may also work for simpler models or setups with more computational resources. Finally, a trend is observed toward better performance in cases with increased granularity for direct collocation.

6 Future Work

Although the use of the CasADi's `DaeBuilder` to calibrate Modelica models shows huge potential, there are still some challenges when following this methodology. A large setback for the proposed workflow relates to the fixed variability for the parameters of an FMU compiled with OpenModelica. This requires a manual redeclaration of all parameters to be estimated as inputs to enable their variability in CasADi as decision variables. Furthermore, FMU simulations are slow when compared to the time needed in the NLP solver per iteration (approximately 20 times longer for the model used in this work) , which leads to excessive computing times. Finally, the lack of support for time events in CasADi's `DaeBuilder` required major changes in the model to accommodate their introduction as inputs, similar to the parameter variability issue. The introduced methodology shows promise in coupling Modelica and CasADi for optimization. However, as shown in this work, it is still in its early stages and there is ample room for further improvements. Joint efforts are needed to come up with a workable solution, which will then be illustrated on multiple applications.

Acknowledgements

The authors acknowledge the financial support by KU Leuven through the TECHPED - C2 project (C24M/21/021). The TECHPED project investigates TECHnically feasible and effective solutions for Positive Energy Districts. This research was partially supported by Flanders Make, the strategic research centre for the manufacturing industry, by SBO projects WORKDRIVE and DIRAC.

References

Andersson, Joel A E (2023 submitted). "Import and Export of Functional Mockup Units in CasADi". In: *Proceedings of the 15th International Modelica Conference*. Aachen, Germany.

Andersson, Joel A E et al. (2019). "CasADi – A software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation* 11.1, pp. 1–36. DOI: 10.1007/s12532-018-0139-4.

Bellomo, Nicola et al. (2007). *Generalized collocation methods: solutions to nonlinear problems*. Springer Science & Business Media.

Beneventi, Francesco et al. (2012). "An effective gray-box identification procedure for multicore thermal modeling". In: *IEEE Transactions on Computers* 63.5, pp. 1097–1110.

Bohlin, Torsten P (2006). *Practical grey-box process identification: theory and applications*. Springer Science & Business Media.

Decker, Sebe De (2021). "Generic Optimization Toolbox for Physical Systems Coupling OpenModelica, CasADi and Python". Master thesis. Leuven, Belgium: KU Leuven.

Drgoňa, Ján et al. (2020). "All you need to know about model predictive control for buildings". In: *Annual Reviews in Control* 50, pp. 190–232.

Fritzson, Peter et al. (2005). "The OpenModelica modeling, simulation, and development environment". In: *46th Conference on Simulation and Modelling of the Scandinavian Simulation Society (SIMS2005), Trondheim, Norway, October 13-14, 2005*.

Gillis, Joris et al. (2020-03). "Effortless modeling of optimal control problems with rockit". In: *Proceedings of the 39th Benelux Meeting on Systems and Control*. Elspeet, The Netherlands.

Jorissen, Filip et al. (2018). "Implementation and Verification of the IDEAS Building Energy Simulation Library". In: *Journal of Building Performance Simulation* 11 (6), pp. 669–688. DOI: 10.1080/19401493.2018.1428361.

Kelly, Matthew (2017). "An introduction to trajectory optimization: How to do your own direct collocation". In: *SIAM Review* 59.4, pp. 849–904.

Kelly, Matthew P (2015). "Transcription methods for trajectory optimization". In: *Tutorial, Cornell University, Feb*.

Mariano-Hernández, D et al. (2021). "A review of strategies for building energy management system: Model predictive control, demand side management, optimization, and fault detect & diagnosis". In: *Journal of Building Engineering* 33, p. 101692.

Michalik, Claas, Ralf Hannemann, and Wolfgang Marquardt (2009). "Incremental single shooting—a robust method for the estimation of parameters in dynamical systems". In: *Computers & Chemical Engineering* 33.7, pp. 1298–1305.

Saurav, Kumar and Vikas Chandan (2017). "Gray-box approach for thermal modelling of buildings for applications in district heating and cooling networks". In: *Proceedings of the Eighth International Conference on Future Energy Systems*, pp. 347–352.

Schittkowski, Klaus (2002). *Numerical data fitting in dynamical systems: a practical introduction with applications and software*. Vol. 77. Springer Science & Business Media.

Shitahun, Alachew et al. (2013). "Model-Based Dynamic Optimization with OpenModelica and CasADi". In: *IFAC Proceedings Volumes* 46.21. 7th IFAC Symposium on Advances in Automotive Control, pp. 446–451. ISSN: 1474-6670. DOI: https://doi.org/10.3182/20130904-4-JP-2042.00166. URL: https://www.sciencedirect.com/science/article/pii/S1474667016384117.

Wetter, M. et al. (2014). "Modelica Buildings library". In: *Journal of Building Performance Simulation* 7.4, pp. 253–270.

Wright, Stephen, Jorge Nocedal, et al. (1999). "Numerical optimization". In: *Springer Science* 35.67-68, p. 7.

Yu, Xingji et al. (2019). "Investigation of the model structure for low-order grey-box modelling of residential buildings". In: *Proc. Build. Simul.*