CALIBRATION OF SCHEDULES IN BUILDING SIMULATION USING RUNTIME MONITORING

C. Tauber, F. Tahmasebi, R. Zach, and A. Mahdavi

Department of Building Physics and Building Ecology
Vienna University of Technology, Vienna, Austria

ABSTRACT

Using runtime measurements from a monitoring system, building performance models can be automatically calibrated on a recurrent basis to decrease deviations from real world conditions. This paper describes algorithms to perform such an automated calibration process. The proposed layer-based structure of the calibration framework allows simple component exchangeability and expandability for implementation of different simulation and optimization tools. The currently proposed approach allows calibration of both non-time dependent model variables (e.g., U-value of a window) and time dependent variables (e.g., schedule of window status).

INTRODUCTION

Numeric performance simulation tools are typically applied to predict the future performance of building designs (Hensen et al. 2011). More recently, the potential of simulation routines is being explored in the buildings' operation phase. Specifically, predictive systems operation approach has shown how simulation engines can be incorporated as an integral component of a building's control system (Mahdavi 2001, Mahdavi et al. 2013). Thereby, to arrive at a preferable control option, implications of alternative control actions for the control task's objective function are evaluated proactively via parametric simulation. Thereby, the reliability of predictions is of essential importance. Monitored data could be used to evaluate and calibrate simulation models to improve their predictive potency (Weber et al. 2012).

Automated recurrent simulation model calibration (see Figure 1) is a key requirement to enable further advanced applications (e.g., virtual sensors) that obtain data from simulation tools. Existing solutions are mostly customized to specific application toolkits and cannot be easily adapted to other systems. Further limitations emerge, when using more complex optimization cost functions, which depend, for example, on external monitored data. An optimization tool such as GenOpt (GenOpt 2014) support multiple simulation applications, but the

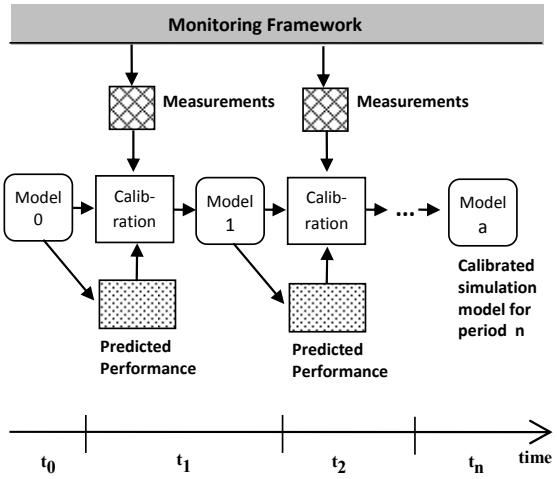

Figure 1: A schema of recurrent simulation model calibration based on measurements obtained from a monitoring framework.

evaluation occurs within the simulation domain. However, complex cost function calculations are difficult to implement within each simulation tool's program flow. Moving the cost function calculation to the optimization domain allows implementing a general framework, independent of the deployed simulation tool and specific solutions (plugins). A layer-model with strict interface definitions can facilitate exchangeability in using different simulation, optimization, and monitoring tools. This paper illustrates an approach to create such a model and reports on an implementation effort within the Monitoring System Toolkit (Most 2014).

APPROACH

Overview

To process automated calibration based on runtime building monitoring, the Monitoring System Toolkit (MOST) is used as the source for measurements obtained from physical sensors. The proposed layer-based structure of the calibration framework allows simple component exchangeability and expandability, including implementations for different simulation and optimization tools. In the current implementation, GenOpt was used for

optimization. Currently, the cost function is typically processed by the simulation application. Technology issues make it very hard to implement cost functions, which depend on values from external data sources (e.g. real-time measurements from a monitoring system). Therefore, the evaluation process was moved to the optimization domain to become independent of the deployed simulation software. A developed plugin-based extension of GenOpt enables the implementation of original cost function algorithms, only restricted by a minimal interface definition that provides the simulation program's output paths and requires the calculated cost function value to be returned.

Figure 2 shows the layer-based structure of the proposed implementation. The Monitoring System Toolkit was extended by a calibration module that serves the functionality for iterative simulation model optimization. It is schematically separated in:

(1) Pre-processing: contains jobs such as the creation of a weather file for the calibration period
(2) Calibration: includes the driver for the optimization program
(3) Post-processing: contains any tasks that should be done after a calibration, e.g. deployment of the calibrated model

The optimization tool GenOpt is extended with a new plugin based cost function calculation functionality. This plugin interface allows the use of custom implementations for the calculation of the cost function used during calibration. This paper describes a plugin, which optimizes an EnergyPlus simulation model based on measurements from the Monitoring System Toolkit. It is further separated in cost function calculation and the basic data reader objects.

This layer approach shows how components could be exchanged easily, for example when switching to another simulation tool without having to adjust the whole solution. In this case a new custom reader object has to be implemented that could provide the simulation output to the cost function algorithm in a standardized way.

Calibration of time dependent variables

Building performance models' input variables can be classified into two fundamental types:

- Time independent variables, which display no (or negligible) change in the regarded time period (e.g. U-Value of a window).
- Time dependent variables, which are defined with an array of values over time, known as schedules (e.g., state of a window: open/closed).

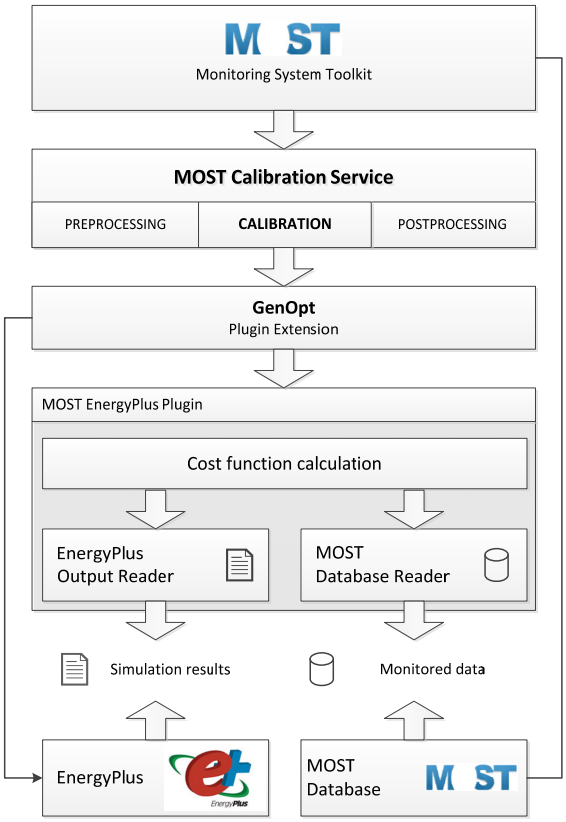

Figure 2: Proposed layer-model using the example of the Monitoring System Toolkit and EnergyPlus.

The calibration of static variables was described in detail in previous publications (Tahmasebi et al. 2012). Time-dependent variables involve more complexities in model calibration because an optimum of a series of values has to be found to reach optimum model performance. A less computationally expensive approach could be populating the model with a pool of possible schedules for time-dependent variables, where the best-fitting one is selected in the calibration process.

To provide this pool of schedules in a simple manner, a number of daily schedules can be picked randomly from the relevant monitored data of a certain period of time. Each set of schedules pertaining to occupant presence and interactions with building systems should be obtained from the same day to avoid inconsistencies in the schedules.

Alternatively, occupants' presence and behavioral models, trained with the monitored data, can be used in order to generate occupancy-related profiles, among other things, based on environmental factors.

Monitoring system toolkit

In a nutshell, the Monitoring System Toolkit (MOST) provides connector interfaces to collect data from various building systems, a database for data storage and data preprocessing, a MATLAB-Framework for complex data processing, and JAVA-

interfaces for different applications. The Building Data Service Interface provides a common data format and abstracted communication to the storage service. Given this structure, a calibration module was implemented that serves periodical calibration processing (Figure 3).

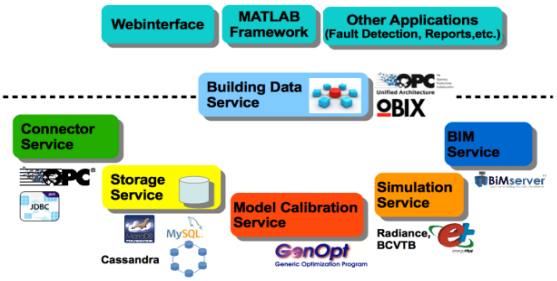

Figure 3: Modular structure of the Monitoring System Toolkit

MOST Standard Data Format

The proposed layer structure of the solution (Figure 2) shows the two different Reader objects in the MOST EnergyPlus Plugin. Both of them use the MOST standard data format to provide the datasets to the next layer above, the cost function calculation. In the Monitoring System Toolkit, there are various data objects specified for standardized data exchange within the project modules.

Table 1:MOST standard data format: DpDataDTO – Represents a single measurement

Property	Type	Description
Timestamp	Date	Precise time mapping
Value	Double	Value of the data object
Quality	Float	Indication of quality

Table 2:MOST standard data format: DpDataset – A list of DpDataDTO objects

Property	Type	Description
DatapointName	String	Name of the data point

The format definitions can be seen in Tables 1 and 2. They illustrate the essential properties to describe single data point measurements and respective sets. DpDataDTO represents such a single measurement object, including the value, the inherent time stamp, and a quality indicator. The measurements of a data point for a period can be listed together in a DpDatasetDTO object. It contains, in addition to regular lists, a property 'DatapointName' for relating it to a unique data point.

GenOpt Plugin Extension

GenOpt 2014 is an optimization tool for minimizing a cost function evaluated by a simulation tool. It can support simulation tools that have input and output based on text file. In the current software design, the cost function calculation must be implemented within the simulation application. If a cost function depends on external data sources (in this example: measurements in the MOST Database), technology issues make it very hard to provide this data during the simulation program's run. Consequently, extensive adjustments would be necessary for every simulation tool deployed.

To provide a universal solution to any given calibration problem, GenOpt was extended and a plugin-extension for cost function calculation implemented. Thus, the evaluation of the cost function is part of the calibration domain, not the simulation domain. At every step in the optimization process, GenOpt does not read-in the evaluated cost function value(s), instead it calls the plugin, passes the path(s) to the simulation program's output and demands the calculated value(s) to be returned. This interface definition is kept minimal, reducing restrictions in implementing customized solutions for a specific use case.

When developing a custom solution for a particular application, a recompilation of the application is not necessary. The specific part could be implemented as plugin and then provided to GenOpt through a plugin folder. To achieve this, a number of adjustments in functionality of GenOpt were necessary when using a plugin:

- The GenOpt initialization file for a particular optimization problem does not have to provide information on the "ObjectiveFunctionLocation", since the plugin serves the cost function calculation. If this part is included it will be ignored.
- Adjustments of the optimization implementations in code, to call the plugin when simulation evaluation should be done, instead of reading the evaluation result of the simulation program's output files.
- Various utilities for loading, serving, and calling plugins.

Plugin interface definition

To standardize the interface between plugin framework and developed plugins, a definition was designed. It contains the properties that are regularly read out of the optimization initialization file of GenOpt. All GenOpt cost function plugins' have to serve this (Table 3).

To enable the plugin to generate the cost function values, it gets the paths to the simulation program's output files when being called at every optimization time step.

Table 3:
Properties of the plugin interface definition

Property	Description
Dimension	Represents the count of the used cost functions that should be optimized.
Names	Represents the designations of the cost functions. They are used for identification at output.
Cost function values	The calculated results of the cost functions, used for evaluation of an optimization step.

Development of plugins

The following instructions should elucidate how to develop a plugin for the GenOpt plugin extension. The used programming language for the MOST project and GenOpt is JAVA. Any JAVA development environment could be used. A new class has to implement the ICostFunction interface that was explained in the previous section. That class represents the plugin solution. It needs a configuration file (META-INF/services), because ServiceLoader is used to load and initialize plugins. The benefit is that the GenOpt plugin extension project does not have to be recompiled when a new plugin is developed. The coded plugin has to be exported as JAR and could be provided in a plugin folder in the GenOpt directory. On startup the proposed plugin extension loads and initializes the plugins.

MOST EnergyPlus Plugin

Based on the interface definition of the GenOpt plugin extension a plugin was developed within the Monitoring System Toolkit to interact with EnergyPlus as simulation tool. The essential functionality is the evaluation of a cost function, which addresses the difference between the measured and simulated values, such as indoor air temperature. The evaluation statistic implemented in current study is the Root Mean Squared Deviation (RMSD). The function parameters include two basic data sources: the simulation program's output and the building monitoring measurements. In the proposed plugin the algorithms to access these sources were abstracted and separated. This layer model makes it possible to replace the simulation application (EnergyPlus) and building monitoring system (MOST).

The data access layer was implemented with two objects:

(1) EnergyPlus Output Reader converts EnergyPlus output files because of interoperability reasons into a desired standard data format used in the MOST-project.

(2) MOST Database Reader provides a caching algorithm to take the load off the database.

The major functionality of the data access layer is to convert the different kinds of used source data, such as database entries from a monitoring system or the simulation program's output to a standard data format. The mapping between sensors defined in the EnergyPlus model and sensor values that are stored in the database is currently realized with naming conventions (identically naming). The next layer upward provides the cost function algorithm realized as RMSD calculation. These two layers are implemented as a plugin for GenOpt.

A MODEL CALIBRATION EXAMPLE

To illustrate how a building performance model is calibrated in the proposed framework, we go through different stages of a calibration example (Tahmasebi et al. 2012).

Building model

The building model was modeled in EnergyPlus, version 7.0. It was assumed that the floor and ceiling elements of the office are adiabatic, as the office is situated between two occupied floors. In the zoning scheme, the open-plan south and north-oriented spaces were separated from the central corridor. However, using the network-based multi zone airflow model of EnergyPlus, the airflow between these connected spaces was simulated. Figure 4 illustrates the building floor plan and the thermal zoning of the building model.

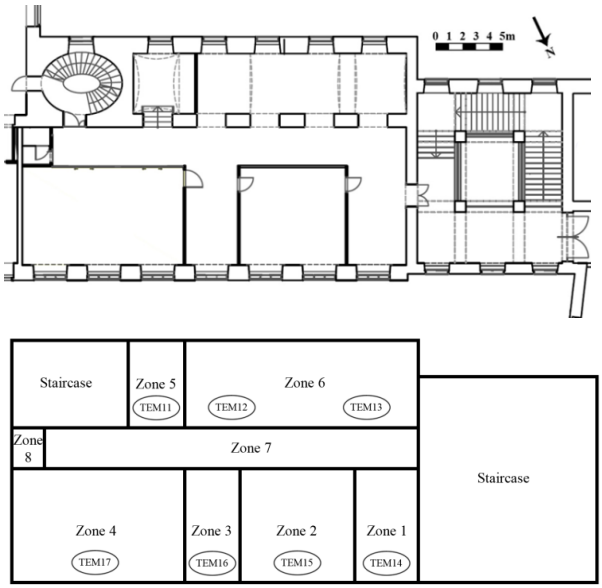

Figure 4: Building floor plan and the thermal zoning of the building model

In this study the zone mean air temperature was used to optimize the simulation model. For the purpose of MOST EnergyPlus Plugin the temperature values per time step were declared as output.

```
== ALL OBJECTS IN CLASS: OUTPUT:VARIABLE ==
Output:Variable,*,tem11,Timestep;
Output:Variable,*,tem12,Timestep;
...
Output:Variable,*,tem16,Timestep;
Output:Variable,*,tem17,Timestep;
```

Figure 5: Output variables of the EnergyPlus model

Preprocessing: Create a local weather file

For the purpose of simulation model calibration, typical-year weather data cannot be used to represent the outdoor conditions. Therefore, real-time measurements should be used to generate the EnergyPlus weather file for the calibration period. The MOST database provides a number of weather parameter measurements which can be used to create a real-time weather data file as a preprocessing step in the calibration process (Table 5).

Table 5: Data used to create weather file

Data point	Unit
Global horizontal radiation	W/m^2
Diffuse horizontal radiation	W/m^2
Outdoor dry bulb temperature	°C
Outdoor air relative humidity	%
Wind Speed	m/s
Wind direction	degree
Atmospheric pressure	Pa

Preprocessing: Populate the model with schedules

Based on the proposed approach, the initial building performance model shall be populated with a pool of schedules, which represent different patterns of occupants' presence and behavior. Toward this end, a number of daily schedules can be picked randomly from the relevant monitored data.

With regards to implementation in the developed calibration framework, a specially designed syntax allows defining placeholders for the schedules in the simulation model. Figure 6 shows the definition within an EnergyPlus model for a schedule "con20", in this example a door open/closed contact sensor. Instead of data a placeholder is inserted. The name inside the start ("<#") and end tag ("#>") declares the data point name in the database of the monitoring system. Before launching a calibration process the EnergyPlus model is searched by this tags and replaced through data of the monitoring system of the run period. The advantage of this approach is the slim simulation model template.

```
Schedule:Compact,
con20,
On/Off,
<#con20#>;
```

Figure 6: Schedule data placeholder

Scheduled variables

The calibration process should find the schedules that cause the best cost function result. GenOpt allows specifying discrete variables that have a set of admissible values. To provide an example, Figure 7 illustrates a part of the EnergyPlus model, where an AirflowNetwork:MultiZone:Surface object represents the interactions with windows in a binary manner via referencing a venting availability schedule. GenOpt applies the schedules provided in the pool (based on the syntax illustrated in Figure 8) and finds the one that minimizes the cost function.

```
AirflowNetwork:MultiZone:Surface,
    SubObj:0225,      !- Surface Name
    SubObj:0225,      !- Leakage Component Name

    …

    %con20%;          !- Venting Availability
                      !- Schedule Name defined
                      !-   as Variable
```

Figure 7: Defining the schedule name as variable

```
Parameter{   // con20 pattern pool
  Name    = con20;
  Ini     = 1;
  Values  = "con20p1, con20p2, con20p3,
             con20p4";
  }
```

Figure 8: Defining the pool of admissible patterns in the GenOpt command file

Calibration: Cost function calculation

For the purpose of building performance analysis error can be defined as the difference between a predicted value and a measured value (Polly et. al. 2011). In the present case, the error was calculated for the indoor air temperature averaged over all office zones. To minimize this error, the "Root Mean Squared Deviation" was used:

$$RMSD = \sqrt{\frac{\sum_{i=1}^{n}(m_i - s_i)^2}{n}} \quad (1)$$

where s_i = simulated values that are read from the simulation programs output; m_i = measured values that are fetched from the database of the Monitoring System Toolkit.

In the proposed calibration framework, a developed plugin-based extension of GenOpt enables the implementation of the cost function outside the simulation model. Thus, the optimization tool GenOpt, which is extended with this plugin optimizes an EnergyPlus simulation model based on measurements from the Monitoring System Toolkit (MOST).

DISCUSSION

Optimization-based simulation model calibration has a great potential to improve the runtime performance of embedded simulation engines in buildings' control and automation systems. A possible approach was presented on how this could be implemented software-aided. The layer-based structure makes it possible to easily exchange parts of the implementation and improve single layer functionality. The provided example shows how the previous work done by scripts in one simplified application can be improved. However, the calibration process itself requires further improvement in terms of efficiency and consistency. Needless to say, automated software implementations could enhance handling big data that would not be possible manually with script support. Moreover, more detailed model parameters could be included.

CONCLUSION

The approach presented in this paper demonstrates how a building performance model could be adjusted to be used in the MOST calibration module, how preprocessing works in the proposed approach, and what are the major benefits of automating a periodic re-calibration of simulation models. Basic calibration operations are implemented already and give a good direction for further research and implementations. Hence, automated simulation model calibration based on a runtime monitoring represents a promising opportunity for performance enhancement in applications pertaining to building automation, diagnostics, facility management, and model-based system control.

ACKNOWLEDGEMENT

The research presented in this paper was supported by funds from the "Klima- und Energiefonds" within the program "Neue Energien 2020".

REFERENCES

EnergyPlus 2012. http://apps1.eere.energy.gov/buildings/energyplus/.

GenOpt 2014. http://simulationresearch.lbl.gov/GO/

Hensen, L. M., Lamberts R. (ed.) 2011. Building performance simulation for design and operation. Spon Press, ISBN: 978-0-415-47414-6.

Mahdavi, A. 2001. Simulation-based control of building systems operation. Building and Environment 36, 789–796.

Mahdavi A, Schuss M. 2013. Intelligent Zone Controllers: A Scalable Approach to Simulation-Supported Building Systems Control. Building Simulation 2013, Building Simulation 2013 - 13th International Conference of the International Building Performance Simulation Association, IBPSA (ed.); IBPSA, (2013), ISBN: 978-2-7466-6294-0; 1498 - 1505.

Most, 2014. http://most.bpi.tuwien.ac.at/.

Polly B., Kruis N., Roberts D. 2011. Assessing and improving the accuracy of energy analysis for residential buildings, U.S. National Renewable Energy Laboratory's (NREL), Sponsoring organization: U.S. Department of Energy, Report number: DOE/GO-102011-3243.

Tahmasebi F, Zach R. Schuss M. Mahdavi A. Simulation model calibration: An optimization-based approach. BauSim2012 - IBPSA Germany-Austria (2012), Paper ID 170, 6 pages.

Tahmasebi F, Mahdavi A. 2013. A two-staged simulation model calibration approach to virtual sensors for building performance data. 25.08.2013 - 30.08.2013. Building Simulation 2013, France.

Weber J., Zach R., Tahmasebi F., Mahdavi A. 2012. Inclusion of user-related Monitoring Data in the run-time calibration of building performance simulation models. BauSIM 2012, Berlin Germany.