The Journal of Open Source Software

AixCaliBuHA: Automated calibration of building and HVAC systems

Fabian Wüllhorst[1], **Thomas Storek**[1], **Philipp Mehrfeld**[1], **and Dirk Müller**[1]

1 Institute for Energy Efficient Buildings and Indoor Climate, RWTH Aachen University

DOI: 10.21105/joss.03861

Software

- Review
- Repository
- Archive

Editor: Frauke Wiese

Reviewers:

- @samanmostafavi
- @shamsiharis

Submitted: 02 October 2021
Published: 22 April 2022

Summary

`AixCaliBuHA` enables an automated calibration of dynamic building and HVAC (heating, ventilation and air conditioning) simulation models. Currently, the package supports the calibration of Functional Mock-up Units (FMUs) based on the Functional Mock-up Interface (FMI) standard (Modelica Association Project, 2021) as well as Modelica models through the `python_interface` of the Software Dymola (Dassault Systems, 2021). As the former enables a software-independent simulation, our framework is applicable to any time-variant simulation software that supports the FMI standard. Using these interfaces, we enable the automated calibration of state-of-the-art building performance simulation libraries such as the `Buildings` (Wetter et al., 2014) or the `AixLib` (Müller et al., 2016) library. Figure 1 illustrates the overall toolchain automated by `AixCaliBuHA`. At the core of `AixCaliBuHA` lays the definition of data types, that link the python data types to the underlying optimization problem and are used for all subsequent steps.

Before executing the calibration, an automated sensitivity analysis can be performed to identify relevant parameters using the `SALib` (Herman & Usher, 2017). As the derivative of simulations is typically not available, the optimization behind the calibration is solved by using already published gradient-free solvers (e.g. Virtanen et al. (2020); King (2009); Blank & Deb (2020)). The whole process is visualized by optional progress plots to inform the user about convergence and design space exploration. Although the toolchain can be fully automated, users are free to perform semi-automatic calibration based on their expert knowledge. Especially for complex models, expert knowledge is still required to define useful parameter boundaries and asses the result quality.

As most of the classes originally created for `AixCaliBuHA` can be used to automate other tasks in simulation based research, we collect them in the separated project ebcpy. ebcpy aims to collect modules commonly used to analyze and optimize building energy systems and building indoor environments. Last but not least the lightweight Modelica Library Modelica Calibration Templates (`MoCaTe`) provides a standardized interface for coupling of Modelica models to the calibration toolchain. However, its usage is optional.

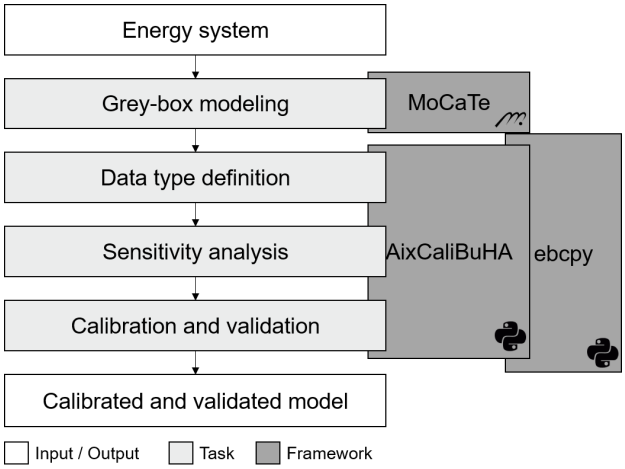

Figure 1: Steps to perform in order to calibrate a model using `AixCaliBuHA`.

Statement of need

Simulation based analysis is a key enabler of a sustainable building energy sector. In order for the simulation to yield profound results, model parameters have to be calibrated with experimental data. Since 74 % of calibrations are performed manually (Coakley et al., 2014), there is a desperate need for an automated software tool. As building energy systems are inherently time dependent, such a tool should enable dynamic calibration. Existing frameworks are either not open-source or a tailored towards a single simulation and optimization method (Arendt et al., 2018; Bonvini et al., 2014). Therefore, we developed `AixCaliBuHA` to automate the calibration process of dynamic building and HVAC system models. `AixCaliBuHA` was developed and tested for Modelica models. However, other simulation environments, for instance EnergyPlus, can be integrated by extending the `SimulationAPI`. For the underlying optimization, we build upon various existing gradient-free frameworks. Currently, switching between different frameworks requires substantial implementation effort and programming knowledge. We thus created a wrapper class to handle different available frameworks. Thus, existing calibration frameworks such as `EstimationPy` or `modestpy` can be integrated (Arendt et al., 2018; Bonvini et al., 2014). Also, novel sensitivity analysis methods may be used, too. In general, `AixCaliBuHA` does not aim to provide new methods. Instead, it uses methods from the rich pool of already existing open-source frameworks. `AixCaliBuHA` was already used in various contributions concerning calibration and digital twins (Mehrfeld et al., 2021; Storek et al., 2019; Vering, Borges, et al., 2021; Wüllhorst et al., 2021). We hope to extend the circle of users and developers by making the code fully open-source.

While implementing `AixCaliBuHA`, we found that some classes and functions may be useful for other research questions. First, the class `SimulationAPI` can be used to automate any simulation based task. Second, the gradient-free `Optimizer` is able to optimize any model parameter of a dynamic simulation model. Third, the custom `TimeSeriesData` may be useful for tasks encompassing energy system analysis. Thus, we bundled these three features into ebcpy. Using the library as a base, various tools for the analysis and optimization of dynamic simulation models may be derived. For instance, it has already been used for the design optimization of heat pump systems in a recent publication (Vering, Wüllhorst, et al., 2021).

Link between optimization and class definition

Before any automated calibration of models can take place, the underlying optimization problem has to be formulated. The goal of any calibration is to minimize the deviation between some

measured data $y(t)$ and simulated data $\hat{y}(t)$ by varying the model parameters p:

$$\begin{aligned} \min_{p} \quad & \sum_{i=0}^{N} w_i \cdot f(y_i(t), \hat{y}_i(t)) \\ s.t. \quad & p_{\mathrm{LB}} \leq p \leq p_{\mathrm{UB}}, \\ & \hat{y}(t) = F(t, p, u(t)) \quad \forall t \in [t_{\mathrm{start}}, t_{\mathrm{end}}] \end{aligned}$$

Inhere, N is the number of variables to be matched by the simulation, w_i is the weighing of the i-th target data and f is some function to evaluate the deviation between y and $\hat{y}$, e.g. the root mean square error (RMSE). As constraints, the parameter may have some upper (UB) and lower boundaries (LB). Additionally, $\hat{y}(t)$ is output of the simulation model F taking the time t, tuneable model parameters p and time-variant input data $u(t)$ as inputs.

This mathematical formulation is transformed into python using a `CalibrationClass`. This class contains the goal of the calibration (mathematically speaking the objective), the parameters to tune (the optimization variables) and further information like simulation time and inputs. Lastly, F is included by calling one child-class of the `SimulationAPI` of `ebcpy`. Figure 2 displays all mentioned links.

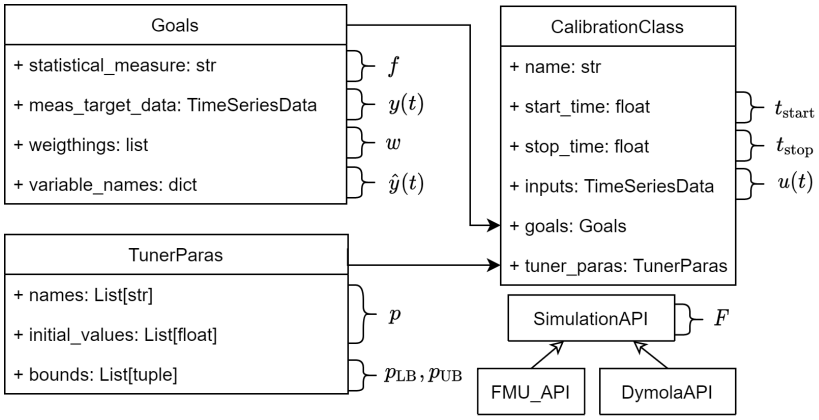

Figure 2: Link between the optimization problem and the `CalibrationClass` object.

Once instances of `CalibrationClass` and `SimulationAPI` are generated, the calibration can run fully automated. However, the user can decide which steps to automate and which steps to perform manually using expert knowledge.

Even though one `CalibrationClass` is enough to run an automated calibration, the quality of calibration can be improved by assigning tuner parameters to different time intervals in the calibration. This idea, initially described in Storek et al. (2019), will be shortly submitted to a corresponding journal. Using segmentation methods, input data can be automatically split into multiple `CalibrationClass` instances. In future work we are going to link this segmentation onto multiple-class calibration. Thus, we further decrease manual user input and increase model accuracy. As the `SimulationAPI` enables parallelization, we plan to apply parallelization for sensitivity analysis and calibration in future versions.

References

Arendt, K., Jradi, M., Wetter, M., & Veje, C. T. (2018). ModestPy: An open-source python tool for parameter estimation in functional mock-up units. *Proceedings of the 1st American Modelica Conference*. https://doi.org/10.3384/ecp18154121

Blank, J., & Deb, K. (2020). Pymoo: Multi-objective optimization in python. *IEEE Access*, *8*, 89497–89509. https://doi.org/10.1109/ACCESS.2020.2990567

Bonvini, M., Wetter, M., & Sohn, M. D. (2014). An fmi-based framework for state and parameter estimation. *Proceedings of the 10th International Modelica Conference*. https://doi.org/10.3384/ECP14096647

Coakley, D., Raftery, P., & Keane, M. (2014). A review of methods to match building energy simulation models to measured data. *Renewable and Sustainable Energy Reviews*, *37*, 123–141. https://doi.org/10.1016/j.rser.2014.05.007

Dassault Systems. (2021). *DYMOLA systems engineering*. https://www.3ds.com/products-services/catia/products/dymola/

Herman, J., & Usher, W. (2017). SALib: An open-source python library for sensitivity analysis. *The Journal of Open Source Software*, *2*(9). https://doi.org/10.21105/joss.00097

King, D. E. (2009). Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, *10*, 1755–1758.

Mehrfeld, P., Nürenberg, M., & Müller, D. (2021). Model Calibration of an Air Source Heat Pump System for Transient Simulations in Modelica. *13th IEA Heat Pump Conference*, 11.

Modelica Association Project. (2021). *FMI: Functional mock-up interface*. https://fmi-standard.org/

Müller, D., Lauster, M., Constantin, A., Fuchs, M., & Remmen, P. (2016). AixLib – An Open-Source Modelica Library within the IEA-EBC Annex 60 Framework. *Proceedings of the CESBP Central European Symposium on Building Physics and BauSIM 2016*.

Storek, T., Esmailzadeh, A., Mehrfeld, P., Schumacher, M., Baranski, M., & Müller, D. (2019). Applying Machine Learning to Automate Calibration for Model Predictive Control of Building Energy Systems. *Proceedings of the 16th IBPSA Conference*, 8. https://doi.org/10.26868/25222708.2019.210992

Vering, C., Borges, D., Sebastian Coakly, Krützfeldt, H., Mehrfeld, P., & Müller, D. (2021). Digital twin design with on-line calibration for HVAC systems in buildings. *Proceedings of the 17th IBPSA Conference*, 8.

Vering, C., Wüllhorst, F., Mehrfeld, P., & Müller, D. (2021). Towards an integrated design of heat pump systems: Application of process intensification using two-stage optimization. *Energy Conversion and Management*, *250*, 114888. https://doi.org/10.1016/j.enconman.2021.114888

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2

Wetter, M., Zuo, W., Nouidui, T. S., & Pang, X. (2014). Modelica buildings library. *Journal of Building Performance Simulation*, *7*(4), 253–270. https://doi.org/10.1080/19401493.2013.765506

Wüllhorst, F., Jansen, D., Mehrfeld, P., & Müller, D. (2021). A modular model of reversible heat pumps and chillers for system applications. *Proceedings of the 14th International Modelica Conference*. https://doi.org/10.3384/ecp21181561