

CO-SIMULATION FOR BUILDING CONTROLLER DEVELOPMENT: THE CASE STUDY OF A MODERN OFFICE BUILDING

Carina Sagerschnig¹, Dimitrios Gyalistras², Axel Seerig¹, Samuel Prívará³, Jiří Cigler³, Zdenek Vana³

1: Gruner AG, Gellertstrasse 55, 4020 Basel, Switzerland, carina.sagerschnig@gruner.ch

2: Automatic Control Laboratory, ETH Zurich, Switzerland

3: Department of Control Engineering, Faculty of Electrical Engineering, CTU Prague, Czech Republic

ABSTRACT

We report our experience from using co-simulation in the context of a demanding building control study. The object under investigation was a 20'000 m² newly-built office building in Munich. Considered was the control of radiant ceilings, heating convectors, mechanical ventilation, blinds and lighting for one floor that was subdivided in 28 building zones. We employed the Building Controls Virtual Test Bed (BCVTB) middleware to couple the EnergyPlus building energy performance simulation software with the MATLAB numerical computing environment. We found that this set-up enables a structured and flexible approach to controller development and building simulation, even for sophisticated control problems. On the downside, debugging of the simulations became more difficult and simulation times were found to increase by a factor 2 to 10. Considerable know-how and effort are currently necessary to set up and operate the co-simulations. A newly developed software layer that helps minimizing this effort is presented.

CONTROL IN BUILDING ENERGY PERFORMANCE SIMULATION TOOLS

Improved building control is gaining increasing attention as a means to improve energy efficiency and comfort of buildings, to handle the complexity of modern building systems, and to make optimal use of modern information and communication technology. The development and testing of novel control technologies can only be accomplished efficiently with the aid of computer simulations. Moreover, dynamic simulations are the only way to study the role of control already in a building's planning phase. It is in this context that co-simulation provides an attractive option to combine existing building simulation software's sophisticated capabilities with the extensive expertise and software tools available from the field of control engineering.

However, modern building energy performance tools were developed and optimized for building energy performance simulation in the first place. Although they can be customized in many ways their flexibility in terms of control is very limited. I.e. they often support building control only in a very rudimentary manner, e.g. by allowing for simple manipulation of set-points or the setting of component availability. Crawley et al. (2005) provide a summary of simulation tool capabilities. Advanced control requires programming features not available in most traditional building simulation tools. Amongst others, this includes the needs for flexible I/O and manipulation of control parameters, matrix algebra, and calling of external optimization routines. One line of current research consists in developing a new

generation of building simulation tools that overcomes these problems (Wetter, 2009). In this paper we consider as an alternative the combination of traditional tools by means of so-called co-simulation.

COUPLING CONTROL AND BUILDING SIMULATION

Co-simulation consists in the integration of different software components by run-time coupling. Fundamentals for building simulation such as coupling strategies and data transfer are described in Trčka et al. (2009).

Several energy performance simulation tools already feature interfaces to other software, e.g. the well-known TRNSYS software offers an interface via the so-called Trnsys Type 155 for coupling to the MATLAB numerical computing environment and fourth-generation programming language (MATLAB, 2011). However, the value of such solutions is often limited by the fact that the building simulators will typically maintain full run-time control over the course of a simulation. Efficient controller development requires a different, much more flexible approach that supports structured and hierarchical coupling of subsystems into overall systems. Precisely this functionality is provided by the Building Controls Virtual Testbed (BCVTB) software developed by the Lawrence Berkeley National Laboratory (Wetter and Haves, 2008).

BCVTB is a middleware that allows coupling of various software codes for distributed simulation. Currently, programs to be linked via the BCVTB are EnergyPlus, MATLAB, Modelica and Radiance. Data exchange with BACnet building automation systems is also possible. Here we report our experiences from using BCVTB to couple the EnergyPlus building energy performance simulation software (EnergyPlus, 2011) with the MATLAB environment in the context of a demanding case study.

CASE STUDY

The object under investigation was a 20'000 m² newly-built office building in Munich (Figure 1). Our goals were (i) to assess the performance of present-day rule-based control (RBC) in terms of energy usage, thermal and light comfort, peak electric power demand and (ii) to investigate possible advanced control approaches including Model Predictive Control (MPC; e.g., Maciejowski, 2001) with the aid of simulations.

In a first step we developed an EnergyPlus model comprising 28 coupled zones that covered an entire floor of the office building. 24 zones were controlled from within MATLAB, 4 auxiliary zones were controlled from within EnergyPlus using a simple two-point controller. The RBC strategies for plant equipment plus radiant ceilings, heating convectors, mechanical ventilation, blinds and lighting as specified by the building planners were analyzed and translated into MATLAB codes such that they could be simulated outside of EnergyPlus. BCVTB was used to close the control loop between the MATLAB controller and the EnergyPlus model by exchange of input and output signals at each simulation time step (Figure 2).

In the case study it was decided to only have high-level control decisions outside of EnergyPlus. Low-level control of zone equipment (such as massflow rates) remained in the EnergyPlus controller domain. Temperature setpoints as well as plant and zone equipment availability signals were determined from within MATLAB. By means of BCVTB, corresponding schedules related to equipment operation and internal gains, as well and

actuator manipulations were passed to EnergyPlus. Table 1 gives an overview of the signals involved in the RBC of the office floor.

	MATLAB to EnergyPlus	EnergyPlus to MATLAB
Controller Outputs: Temperature setpoints	101	–
Controller Outputs: Plant and equipment availability	45	–
Controller Outputs: Blinds positions and slat angles	62	–
Internal Gains Status: Occupancy, equipment fraction	48	–
Controller Inputs: Zone temperatures, return temperatures	–	53

Table 1: Overview of information flows managed by BCVTB for the rule-based control of 24 building zones.

IMPLEMENTATION

Implementation of the co-simulations required correct handling of a large number of software objects related to the exchanged signals (Table 1) plus the global simulation parameters (simulation begin and end time, time step etc.) in all three involved software components:

BCVTB requires the specification of the global simulation parameters, the co-simulation actors, their respective in- and outputs, and their coupling. All this data had to be provided in an Extensible Markup Language (XML) coded text file.

EnergyPlus requires that the simulated building's geometry and construction details, the zone and plant equipment, the plant's water and air loops, the internal gain schedules, the control, and the definition of the requested outputs are specified in a so-called input data file (IDF). IDFs contain records defining various types of objects plus optional, non-standard control functions that are written in the EnergyPlus Runtime Language (Erl). IDFs are human readable text files, but they can become very large – in our case the IDF without high-level controls had 28'000 lines of code. Including the high-level RBC controller would have added another ca. 3'000 lines of code.

MATLAB offers a powerful programming language that supports among other things elementary typed variables, vectors and matrices, structured variables (consisting of elementary plus structured variables) and functions. In our case, all inputs to the MATLAB controller were provided by BCVTB at runtime in a real valued vector. Its elements had to be mapped to corresponding MATLAB data structures for processing by the control algorithms that in turn returned a real valued vector to be provided by BCVTB as an input to EnergyPlus.

Figure 3 shows the architecture of a general software that was developed for the support of the co-simulations. It consisted of three main elements:

- (i) Preparation of EnergyPlus IDFs ("PrepEPIDF"). This program takes EnergyPlus objects from a model objects database as an input. The objects are defined in Excel files with predefined structures. The program produces the EnergyPlus simulation's IDF as an output.
- (ii) Preparation of BCVTB and controller files ("PrepBCVTB"). This program retrieves the specifications of all EnergyPlus control inputs and of all EnergyPlus outputs required by the external controller from the above-mentioned Excel database. It produces two kinds of outputs: the XML file required by BCVTB, and MATLAB codes that define all objects needed for the interaction of MATLAB with BCVTB.

(iii) Running of the co-simulation and automated post-processing of its results (“RunBCVTB”).

All above programs were implemented as Unix/Linux shell scripts. They were based on standard Unix/Linux tools plus some auxiliary scripts involving programs that were written in the Python and Pearl programming languages.

In order to assess the performance of the co-simulations we considered annual simulation runs with two different building models. Each model was first run stand-alone to obtain a benchmark, and then it was run using BCVTB. All simulation experiments were done with EnergyPlus 6.0.0 and BCVTB 1.0.0 under the Debian Linux 5.0 operating system on a computer with 8 Intel Xeon processors at 2.53GHz. From Table 2 can be seen that for the 28 zone model the co-simulation was ca. 2.5 times slower than the stand-alone simulation, whereas for the smaller 9-Zone model the co-simulation was slower by a factor of 10.

	28-Zone Model	9-Zone Model
Number of exchanged inputs	315	76
Number of exchanged outputs	261	83
Computation time stand-alone simulation	71 min	7 min
Computation time co-simulation	181 min	71 min

Table 2: Comparison of BCVTB performance for annual runs of different sized EnergyPlus models (simulation time step 10 minutes)

DISCUSSION

The proposed co-simulation approach was found to have several advantages as compared to stand-alone simulations with EnergyPlus:

Firstly, it allows the controllers to be coded in a highly structured and modular way. This dramatically enhances code reusability and maintainability. For instance, we used for each given RBC rule (e.g. for heating availability or for blind positioning) a separate function that can be easily replaced without interfering with the rest of the controller.

Secondly, it allows all code related to control to be pulled together such that it is much easier and safer to modify. Moreover, controller parameters can be conveniently summarized in Excel spreadsheets and can be dynamically adjusted at run-time. Quite differently, IDFs support only static control specifications, and these are typically spread out over several hundred lines of code and multiple instances of EnergyPlus objects.

Thirdly, the proposed solution makes it possible to flexibly drive the building model with arbitrary signals in order to study the system’s dynamics, or to identify simplified models for MPC. These possibilities are far beyond the support currently provided by EnergyPlus for parametric simulation runs.

Finally, we note that the rich programming capabilities provided by MATLAB are not limited to control purposes. Integration of MATLAB also enabled efficient post-processing of EnergyPlus output files.

On the downside, the proposed co-simulation approach requires additional effort to define and implement interfaces for data exchange. However, the general software developed (Figure 3) was found to drastically reduce the probability of errors and to shorten development cycles.

A further disadvantage is the found increase in simulation time. This was partially due to communication overhead. However, our results (Table 2) suggest that the number of exchanged variables could be of secondary importance. We suspect that the found increase in simulation time relates to the discontinuity introduced by the breaking up of control into external high-level control and EnergyPlus internal low-level control. This could cause an increase in the average number of iterations EnergyPlus has to perform internally within each simulation time step in order for its simulation results to converge.

Finally, debugging of the co-simulations proved not very easy, since BCVTB in its default configuration can only listen to ports i.e. transmitted values. Adding of breakpoints into the MATLAB code is not possible since it terminates the entire simulation run. Advanced features are available only after recompilation of a given BCVTB actor, which is however only possible for open source software. Therefore the MATLAB debugger cannot be used during co-simulation. This problem can at present only be alleviated by writing of extensive output files.

CONCLUDING REMARKS

The application of advanced control approaches to the built environment makes it necessary that highly sophisticated building and plant models such as EnergyPlus can be integrated in controller development. This can be well achieved by the proposed tools suite, when enhanced by an additional software layer to support the co-simulations.

This case study presented a joint effort of building and control engineers. From an organizational point of view we can conclude that co-simulation can much facilitate multi-disciplinary collaboration and presents a powerful tool to efficiently integrate dispersed knowledge and expertise.

ACKNOWLEDGEMENTS

The support of this research by ICADE REIM Deutschland GmbH is gratefully acknowledged.

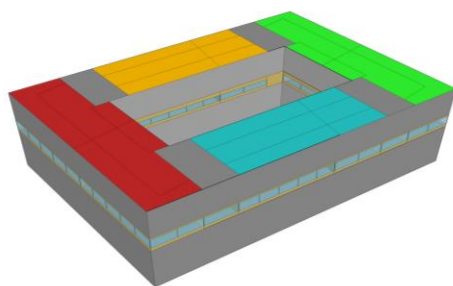


Figure 1: The building investigated and its 3D representation for the simulation of the third floor (other floors are used for shading purposes only). For illustration the zone layout is shown on top. Colored zones were controlled via BCVTB.

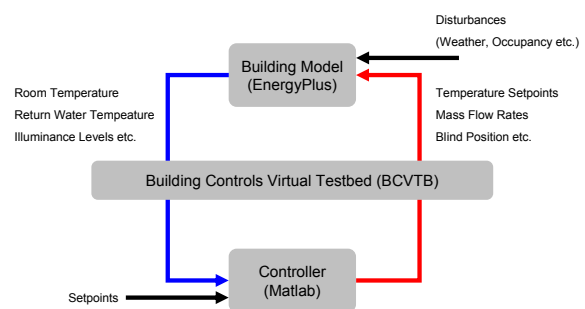


Figure 2: Schematic of the implemented control loop.

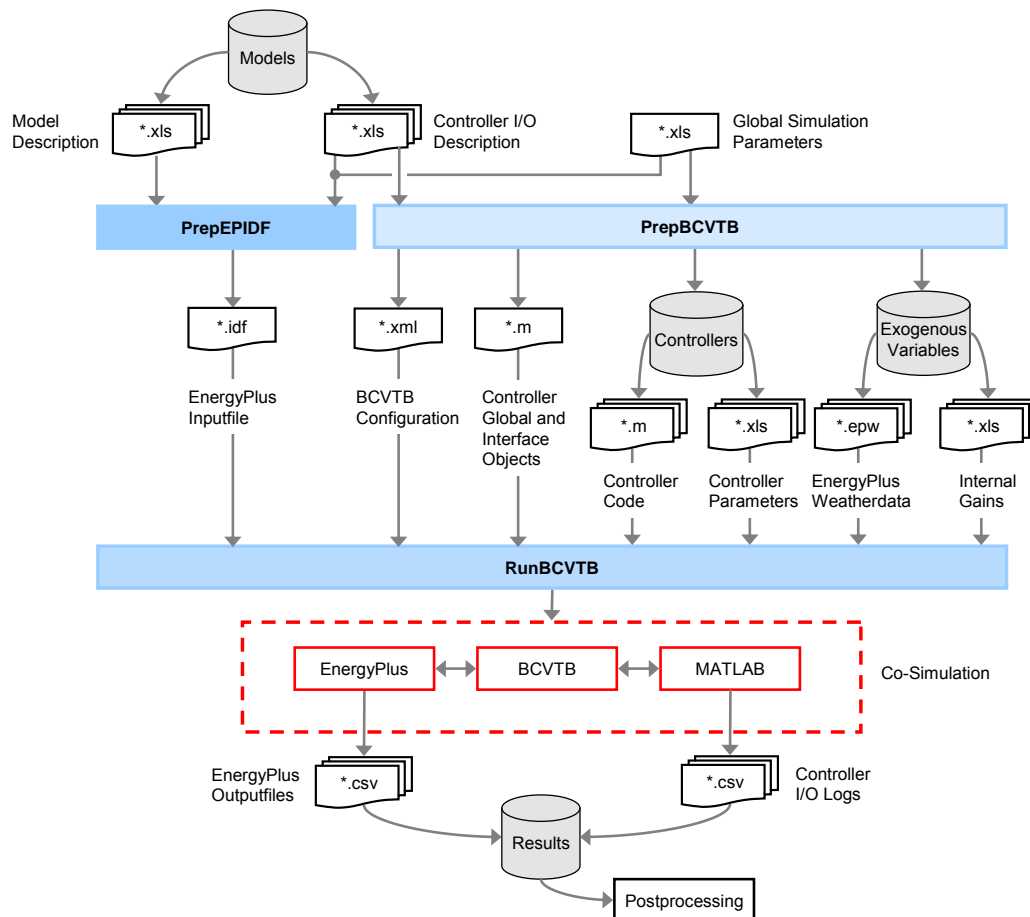


Figure 3: Data flow in the proposed software solution for co-simulation support

REFERENCES

1. Crawley et al.: Contrasting the Capabilities of Building Energy Performance Simulation Programs, U.S. Department of Energy, Building Energy Software Tools Directory, 2005
2. EnergyPlus: Building Energy Performance Simulation Tool, www.energyplus.gov, 2011
3. Maciejowski, J. M.: Predictive Control with Constraints. Prentice Hall, 2001
4. MATLAB: Numerical computing environment by MathWorks, 2011
5. Prívora S., Cigler J., Váňa Z., Sagerschnig C., Gyalistras D., Morari M., Ferkl L.: Modeling and Identification of Large Multi-Zone Office Building. In Preparation. 2011
6. Trčka M., Hensen J., Wetter M.: Co-simulation of Innovative Integrated HVAC Systems in Buildings, Journal of Building Performance Simulation, Vol. 2, No. 3, September 2009, p.209-230
7. Wetter M., Haves P.: A Modular Building Controls Virtual Test Bed for the Integration of Heterogenous Systems, Third National Conference of IBPSA-USA, Berkeley/California, <https://gaia.lbl.gov/bcvtb>, 2008
8. Wetter, M.: Modelica-based modeling and simulation to support research and development in building energy and control systems. Journal of Building Performance Simulation, 2(2):143-161, 2009