Prototyping the BOPTEST Framework for Simulation-Based Testing of Advanced Control Strategies in Buildings

David Blum[1], Filip Jorissen[2], Sen Huang[3], Yan Chen[3], Javier Arroyo[2], Kyle Benne[4], Yanfei Li[4], Valentin Gavan[5], Lisa Rivalin[1], Lieve Helsen[2], Draguna Vrabie[3], Michael Wetter[1], Marina Sofos[6]
[1]Lawrence Berkeley National Laboratory, Berkeley, USA
[2]KU Leuven, Leuven, Belgium
[3]Pacific Northwest National Laboratory, Richland, USA
[4]National Renewable Energy Laboratory, Golden, USA
[5]Engie Lab, Pierrefitte-sur-Seine, France
[6]Department of Energy, Washington D.C., USA

Abstract

Advanced control strategies are becoming increasingly necessary in buildings in order to meet and balance requirements for energy efficiency, demand flexibility, and occupant comfort. Additional development and widespread adoption of emerging control strategies, however, ultimately require low implementation costs to reduce payback period and verified performance to gain control vendor, building owner, and operator trust. This is difficult in an already first-cost driven and risk-averse industry. Recent innovations in building simulation can significantly aid in meeting these requirements and spurring innovation at early stages of development by evaluating performance, comparing state-of-the-art to new strategies, providing installation experience, and testing controller implementations. This paper presents the development of a simulation framework consisting of test cases and software platform for the testing of advanced control strategies (BOPTEST - Building Optimization Performance Test). The objectives and requirements of the framework, components of a test case, and proposed software platform architecture are described, and the framework is demonstrated with a prototype implementation and example test case.

Introduction

Background

Advanced control strategies (ACS) for building HVAC systems, such as model predictive control (MPC), have the potential to provide significant energy savings for reducing operational costs, greater demand flexibility for providing grid services, and improved occupant comfort (Afram and Janabi-Sharifi 2014). While such ACS have demonstrated their potential in research and field tests, their adoption at scale in industry is still limited. The primary barriers to adoption for any new building technology include: installation costs, performance risks and uncertainties, and lack of understanding and quantification of benefits (Chan et al., 2017). Particularly for MPC, implementation costs (Rockett and Hathway 2016) and lack of approach comparisons (Afram and Janabi-Sharifi 2014) are problems. Simulation-based testing of ACS can help solve these challenges by reducing risks associated with malfunction, reducing costs for equipment and installation, reducing real implementation costs by testing software and deployment processes in advance, and controlling the testing environment for comparison of strategies and evaluation over varying conditions. Despite these benefits, modeling limitations within simulation programs, research-grade co-simulation environments, and lack of publicly available benchmark cases have prevented simulation-based testing from reaching its potential for scaling ACS.

Therefore, this paper introduces the development of a framework for simulation-based testing and comparison of building ACS, called BOPTEST - Building Optimization Performance Test, depicted in Figure 1. Key elements of the framework includes use of the Functional Mockup Interface (Blochwitz et al. 2012) standard and Modelica (Mattsson and Elmqvist 1997) to simulate dynamic building response, implementation in a software architecture for scalable deployment and use, and provides a platform for making benchmark test cases publicly available. The current focus of the framework is the testing and objective comparison of ACS algorithm performance and does not cover issues such as network cyber security, communication protocols, and bandwidth.

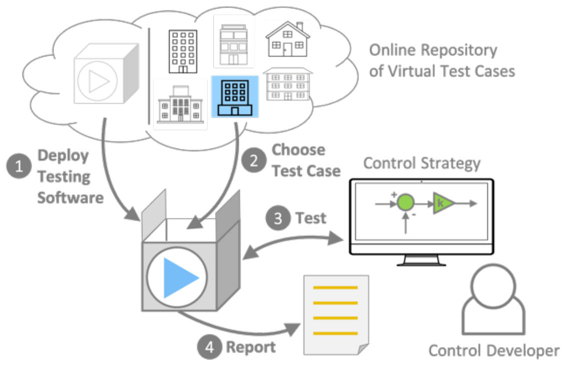


Figure 1. Framework concept.

Related Previous Work

The concept of using simulation of HVAC systems for controls testing has been explored previously by the participants of the IEA-ECBCS Annex 17 (IEA 1997). Six participants built building emulators using TRNSYS (Klein et al. 2017) and HVACSIM+ (Park et al. 1985) as simulation programs. Notably, after testing four emulators of the same building with the same BEMS, differences in calculated energy use between any one of the emulators and the average of all emulators was as high as 6%. Further development introduced new dynamic component model libraries and emulators.

The first was SIMBAD (SIMulator of Building and Devices), developed by France's Centre Scientifique et Technique du Bâtiment in the early 90's (Husaunndee et al. 1997), for the MATLAB/Simulink (Mathworks, 2000) environment. SIMBAD was then used to build Simbad GTB and SIMTEST (Lahrech et al. 2002) for testing complete building management systems and certification of control products according to EN 15500 as well as Simtrain (Soethout, 1998) and Qualisim (Riederer et al. 2001) for training and innovative development purposes. A second component library was for the HVACSIM+ and TRNSYS environments (Haves and Norford 1997) used as the simulation basis for the Virtual Cybernetic Building Testbed (VCBT) (Bushby et al. 2010). The development of EnergyPlus (Crawley et al. 2001) and Modelica Buildings Library (Wetter et al. 2015) led to emulators using the Building Controls Virtual Testbed (Wetter 2011) and the VOLTRON platform (Huang et al. 2018). To train the skills of operators, the development of an emulator with a BACnet interface will hold the 1st World Championship in Cybernetic Building Optimization (SHASEJ, 2019). Finally, NREL has recently developed Alfalfastack (https://github.com/NREL/alfalfa), a web-hosted emulator using EnergyPlus as simulation engine.

While these emulators and tools facilitate development of ACS, they do not meet all of the requirements outlined below for ACS comparison and benchmarking.

Requirements

The requirements have seven aspects worth considering:

1) Reference emulation models must simulate the physics, dynamics, and time-resolution necessary for controls design and assessment at the supervisory and local-loop levels. This requires modeling of not only envelope heat transfer and airflow networks, but also dynamic actuators like valves and dampers.

2) The simulation environment must be standardized so that results for benchmarking are consistent. This includes the solver and tolerance, computing environment, and implementation tools.

3) Data exchange between a test controller and the emulator should be facilitated by an interface that is independent of the modeling and controller programming languages, preventing limits on the implementations of controllers and interfaces to the framework.

4) All exogenous data that defines a test case should be provided by the framework, such as weather, occupancy schedules, and energy prices. In cases of MPC testing, it needs to be made available as forecasts. Providing deterministic forecasts is the priority. Stochastic forecasts will be considered as a future extension.

5) A standard set of key performance indicators (KPI) should be specified to facilitate benchmarking and comparison of controllers. The specification needs to include equations or guidelines to unambiguously quantify the KPIs, enabling a fair and clear comparison between controllers.

6) Flexibility in synchronizing simulation and controller times to meet different application requirements. In Option 1, the simulation is advanced to the next time step according to real time, representing a realistic building-controller interaction. In Option 2, the simulation is advanced to the next time step when the controller returns with an updated control action, which is easier for controller development and allows for reproducible tests.

7) All software is open source and documented to allow for inspection of the models, their underlying assumptions and the computing platform.

Value to Stakeholders

The requirements of the BOPTEST framework translate into features designed to facilitate the needs and objectives of stakeholders interested in either investing in or developing ACS from the design through implementation stages (i.e., algorithm researchers and industry developers, building owners and facility managers, and administrators of research and development (R&D) programs). First, the menu of KPIs and test cases, along with the flexibility in synchronizing simulation and controller times, offers the ability to cater performance evaluation metrics based on the parameters and conditions that a particular stakeholder is optimizing for using an ACS along with the building type or configuration targeted. The KPIs, along with the emulation environment, also provide a standardized comparison for consistency in making fair comparison between algorithms. Finally, the open-source emulation environment and documentation are useful tools for transparent testing when expertise or resources (e.g., detailed model, simulation environment) are not readily available. It provides a critical first step in evaluating an ACS before moving to physical test beds and real, operational buildings. These features can accelerate the technology development process.

For researchers and developers, a fair comparison of different ACS is currently hampered by the use of different boundary conditions and variety of assessment indicators. A common framework, consisting of test cases, KPIs, and a deployment platform for the testing of ACS, allows this fair comparison of ACS developed by researchers world-wide. This approach generates insights into which ACS performs best in which building type and associated boundary conditions according to a specific KPI, facilitating further developments in ACS.

For building owners and facility managers, a big challenge is maintaining low operating costs. Advanced sensing and metering technologies, data collection and analytics, and energy modeling combined with ACS offer the ability to monitor and optimize in near-real-time. The framework offers a platform to de-risk ACS by demonstrating performance and affordability with supporting test data.

Finally, for decision-makers investing in a portfolio of R&D strategies, the framework provides a useful methodology with which to assess the level of maturity, how ACS approaches will complement each other, contribute to achieving overarching sectoral goals with respect to reducing energy consumption and cost, and eliminate any portfolio gaps.

Objective

The goal of the BOPTEST framework is to enable simulation-based testing and benchmarking of advanced control strategies in buildings for researchers and industry that satisfy the specified requirements. This will occur through the development of the following components, each described in subsequent sections:

- Reference building emulation test cases that are available to all users (Section: Test Cases).
- Key performance indicators (KPIs) that quantify and assess the performance of a control strategy (Section: Key Performance Indicators).
- Software platform to select and manage test cases, exchange control and measurement data between the user's control software and the building emulation, calculate KPIs, and generate reports (Section: Software Platform).

A final section of the paper presents a prototype example of the framework (Section: Prototype Example).

Test Cases

Test cases contain the building model, boundary conditions, documentation, and other content related to the framework application programming interface (API) to run the emulation. Reference cases will be provided, satisfying the fourth requirement of the framework, while user-specified cases will also be functional if they abide by established development guidelines.

Building Energy Models

The framework uses FMI v2.0 for simulating Functional Mockup Units (FMU). It supports model-exchange and co-simulation FMUs using pyFMI as part of the JModelica.org distribution (Modelon 2017). For reference cases, the models will be written in Modelica and compiled into FMUs. Using Modelica addresses the first and seventh requirement of the framework as an open-source, equation-based, object-oriented language developed to model hybrid physical systems. It offers the ability to utilize shareable component libraries and variable time-step solvers for systems of nonlinear differential-algebraic systems of equations (DAE). The models will be developed using Modelica libraries extending the IBPSA Modelica Library, formerly called Modelica Annex 60 Library (Wetter et al., 2015).

Ten building models have been identified as references, presented and described in Table 1. The models range in building type, size, and HVAC system to test controllers over a wide range of operating cases.

Boundary Conditions

Boundary conditions of a test case define the exogenous operating conditions, including weather, internal load and occupancy schedules, energy prices, and carbon emission factors, and will be provided specifically to each reference case. However, variations may test different scenarios. An example is providing three energy pricing schemes, constant, moderately dynamic, and highly dynamic to evaluate a controller's ability to shift load.

Documentation

Documentation for the reference cases will be provided to inform users of building design and use, HVAC system, other systems, such as lighting, shading, or renewable generation, model implementation details, such as infiltration or media, and sensor and control signals.

Framework API Interaction

One important interaction is the exchange of input and output data between the test controller and emulation model at each simulation step. Other interactions include the starting, stopping, and resetting of simulations, setting of options, and choosing between test case scenarios. More detail is provided in the section Prototype Software.

Table 1 Overview of selected reference models

Type	**Size**	**Water-Based**	**Air-Based**
Residential	**Single Zone**	BESTEST (ANSI/ASHRAE 2007) Case 900 construction with hot water radiator.	BESTEST (ANSI/ASHRAE 2007) Case 900 construction with forced air heating and cooling.
	Multi Zone	8-zone detached residential building with hot water radiator heating and central boiler with controller. The cooling is provided by room split systems controlled simultaneously by a central controller.	Detached House Central Air (Basis of Design TBD)
Commercial	**Single Zone**	Single zone of teaching/office building with hot water radiator heating. CO_2-controlled VAV ventilation from AHU with heat recovery wheel and heating coil.	Single-zone building with RTU containing direct expansion (DX) cooling coil and gas-fired heating.
	Multi Zone	28-zone office with concrete core activation (4 sections, 1 circuit), 2 geothermal heat pumps and geothermal passive cooling, 1 cooling coil, 1 thermal wheel, 22 heating coils for 1 AHU, 15 VAVs and 11 CAVs.	5-zone office floor with ASHRAE 90.1 construction. VAV hot water reheat for each zone and 1 AHU containing economizer, chilled-water cooling coil, and hot water heating coil. Air system only.
	Complex Multi Zone	32-zone office using concrete core activation (24 sections, two circuits), 4 geothermal heat pumps and geothermal passive cooling, 1 pellet furnace, solar collectors and 2 indirect evaporative heat exchangers, 26 heating coils for 2 AHUs, 24 VAVs and CAVs.	15-zone office with VAV hot water reheat. Three AHUs with five terminal boxes each, economizer, chilled water cooling coil, and hot water heating coil. Central plant includes chiller and boiler.

Key Performance Indicators

Key performance indicators (KPI) constitute the basis on which ACS performance is evaluated. While many possible KPIs exist, a core set has been chosen to serve as standard KPIs to be evaluated by default for every test using the BOPTEST framework. These are described below, presented mathematically in Table 2, and satisfy the fifth requirement of the framework.

Thermal discomfort

This KPI is calculated as the integral of the deviation of the temperature with respect to a predefined comfort range during a given time period, expressed in *Kh*.

Total building energy use

This KPI represents the total building energy use in *kWh* when accounting for all energy end uses over a given time period. The scenarios defined in each test case determine which components are included.

Total building CO_2 emissions

This KPI quantifies the total amount of CO_2 emissions in *kg* over a given time period using a fixed emission factor profile for each emulator. This emission factor is chosen based on source-to-site energy profiles for energy use types at the testing location.

Total operational cost

This KPI quantifies the total operational cost over a given time period using a price profile for each energy end-use. Profiles are fixed for each emulator and three specific archetypes of profiles are defined: constant, moderately dynamic, and highly dynamic.

Capability of the controller to steer flexibility

A controller capable of estimating and steering energy demand flexibility presents added value to the grid. This KPI considers how well a controller follows an artificial external signal within a predefined scenario where boundary conditions are given. Then, characteristics such as those defined by Junker et al. (2018) and considered in Annex 67 (Pernetti et al., 2017) are calculated.

Installation metrics

The installation metrics refer to the effort and cost required to implement in real life. Many aspects play a role and are intrinsically subjective. Therefore, a set of qualitative metrics and associated descriptions are developed and the user who is testing a controller shall assign a description to each installation metric.

Maximum allowed capital cost

The maximum allowed capital cost is the installation cost that would lead to a maximum payback period of five years. The reason for calculating the maximum allowed capital cost instead of payback period directly is the qualitative nature of installation metrics which could hamper the quantification of payback period. On the contrary, the maximum allowed capital cost to obtain a fixed payback period of five years can be objectively quantified with a reference baseline controller.

Computational time ratio

This KPI is defined as the average ratio of computation to sampling times. The computation time is the time required by the controller to compute control outputs during one iteration. The sampling time is the real time lapse between two instants where the control outputs are computed and applied in the building. This KPI quantifies the computational effort required by ACS.

Table 2 List of core KPI definition/calculation

Key Performance Indicators	Calculation formula / Definition	Nomenclature
Thermal discomfort	$D(t_0,t_f)=\sum_{z\in\mathbb{Z}}\int_{t_0}^{t_f}\lvert s_z(t)\rvert\,dt$	ϵ - total amount of CO_2 emissions ξ- the set of equipment in the system with an associated energy use of any type e_i- the emission factor of equipment$i$ n- the number of iterations that take place between t_o and t_f p^τ- the price profile of equipment i with a tariff τ $s_z(t)$- the deviation (slack) from the lower and upper set point temperatures established in zone z t_o- initial time t_f- final time $t_c(k)$-the computational time at iteration k $T_s(k)$-the sampling time at iteration k z - the zone index for the set of zones in the building $\mathbb{Z}$ C^τ- the total cost with a tariff τ D- total discomfort time E- total amount of energy use P_i- instantaneous power use of equipment i
Total building energy use	$E(t_0,t_f)=\sum_{i\in\xi}\int_{t=t_0}^{t=t_f}P_i(t)dt$	
Total building CO_2 emissions	$\epsilon(t_0,t_f)=\sum_{i\in\xi}\int_{t=t_0}^{t=t_f}e_i(t)P_i(t)dt$	
Total operational cost	$C^\tau(t_0,t_f)=\sum_{i\in\xi}\int_{t=t_0}^{t=t_f}p_i^\tau(t)P_i(t)dt$	
Capability of the controller to steer flexibility	To be defined as capability of a controller to follow an artificial external signal within a predefined boundary conditions.	
Installation metrics	To be defined as a set of metrics to evaluate the effort and cost required to get the controller implemented and running.	
Maximum allowed capital cost	$CAPEX_{max}^{5years}=5(C_{1year}^{old}-C_{1year}^{new})$	
Computational time ratio	$t_r(t_0,t_f)=\frac{\frac{\sum_{k=1}^{n}t_c(k)}{n}}{\frac{\sum_{k=1}^{n}T_s(k)}{n}}=\sum_{k=1}^{n}\frac{t_c(k)}{T_s(k)}$	

Software Platform

Architecture

The software platform architecture is proposed in Figure 2 and is based on the previously mentioned Alfalfastack project to promote useability and scalability. This architecture consists of four major components:

Emulator pool - Contains source files of the test cases and temporary files generated during the simulation.

Database - Contains updated values and metadata of all input/output points. Allows data exchange between emulator and external controller to be synchronous or asynchronous, satisfying the sixth requirement.

Simulation Manager - Provides the environment to run the simulation, parses the source files of the emulators to obtain the simulation information, configures/conducts simulations, exports metadata to the database, and exchanges data between the database and simulation.

HTTP Rest API - HTTP Rest API is the main point of interaction with the BOPTEST platform and satisfies the third requirement. Via the HTTP Rest API, the external controller as a client can submit requests for actions such as adding or selecting an emulator to test, extracting information about the emulator, setting simulation settings, starting a simulation, and reading/writing control signal and measurement data.

Docker (https://www.docker.com/) containers are used to implement the proposed architecture. They allow for standardized, rapid, and scalable deployment of the platform on a range of local and cloud-based computing resources using Linux, Windows, and macOS. Also, specifically for the Simulation Manager, Docker addresses the second requirement of the framework. The FMU simulator, solver, and dependencies can be exactly specified within the Docker container.

Modelica Blocks for Signal Exchange

Two Modelica blocks for handling input and output signal exchange have been developed for the IBPSA Modelica Library (development at https://github.com/ibpsa/modelica-ibpsa commit `e51759a`) to:

- Facilitate the propagation of many input/output signals in large Modelica models.
- Allow such models to contain local-loop controllers where either the setpoint or actuation signal can be written by the test controller, enabling testing of supervisory or local loop controllers.
- Facilitate tagging of signals for KPI calculations.

The overall concept is presented in Figure 3. The first signal exchange block is `IBPSA.Utilities.IO.SignalExchange.Overwrite`, which can switch the output of the block between input and external signals. The second signal exchange block is `IBPSA.Utilities.IO.SignalExchange.Read`, which passes an input signal through to an output. Each of the two blocks contains a protected parameter (hidden from user adjustment), `boptestOverwrite=true` and `boptestRead=true` as appropriate, used by a Python parser to identify block locations throughout the model. The read block contains an additional parameter `KPIs`, with which the user can associate KPIs with the signal.

A Python parsing script writes a new top-level Modelica model in which the original model is instantiated, unique activation and signal inputs are added and connected to corresponding Overwrite blocks, and unique outputs are added and connected to corresponding Read blocks. The new top-level Modelica model is exported as an FMU. In addition, a json file is exported containing a list of FMU outputs that are needed to calculate each KPI.

Prototype Example

An implementation of the framework is being developed at https://github.com/ibpsa/project1-boptest to prototype the key components. Open development satisfies the seventh requirement of the framework. The development site can be referred to for more detail about use and API than presented in this paper.

Test Case

The example test case illustrates the capabilities of the framework and is presented in Figure 3. The emulation

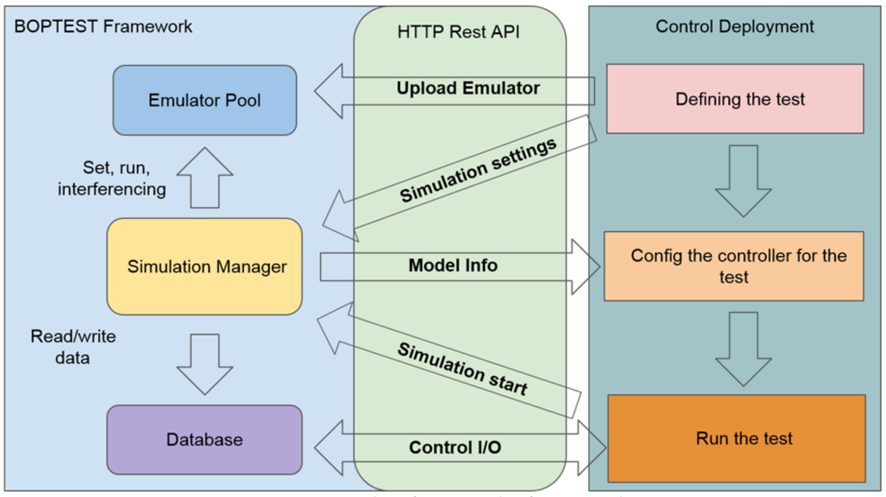


Figure 2. Proposed software platform architecture.

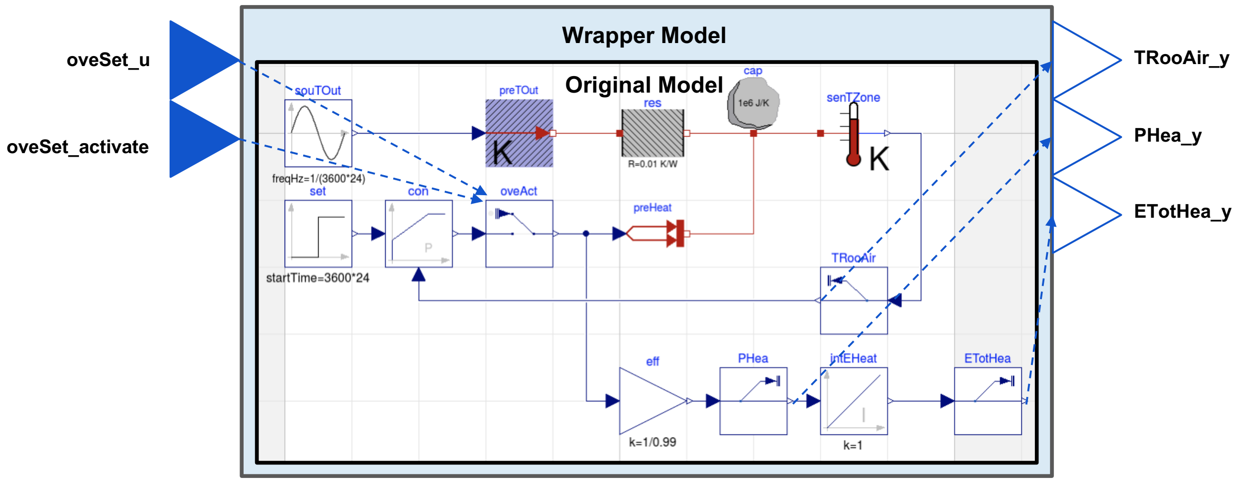


Figure 3. Signal exchange blocks (tan) and Python parser facilitate use of a Modelica model within the framework.

model is a single thermal zone with heater, represented by an RC network and direct heat input to the heat capacitor. The outside air temperature is represented by a sinusoidal signal with offset 20 °C, amplitude 10 °C, and period of 24 hours. A proportional feedback controller is included in the model that tracks a zone temperature setpoint by adjusting the heater output.

An Overwrite block is added between the output of the proportional controller and input of the heater in order for an external controller to control heater actuation. If external actuation is not activated, the actuation determined by the modeled feedback controller would be used. Read blocks are added to the zone temperature measurement, power measurement, and energy measurement. The zone temperature Read block is parameterized with KPI "comfort" and the energy read block with KPI "energy." The Python parser is invoked to export `Wrapper.mo`, `Wrapper.fmu`, and `kpis.json`, which are the final emulation model components.

Software

The deployment solution implemented for this prototype test case demonstrates core components of the proposed architecture in Figure 2, namely the Simulation Manager and HTTP Rest API. As presented in Figure 4, the solution utilizes Docker to package the test case emulation model components outlined in the previous section into a container with Ubuntu 16.04, Python 2.7, JModelica and pyFMI (Modelon 2017), required Python packages, and two core Python scripts. The first of these scripts `testcase.py` acts as the simulation manager by instantiating the model FMU (`Wrapper.fmu`), defining structures for data trending, containing functions for getting/setting. communication step, having a function for stepping the simulation forward, and implementing modules for calculating KPIs as directed by the `kpis.json`. The second script `restapi.py` implements the HTTP Rest API, mapping web requests to functionality provided by `testcase.py`.

A makefile builds the Docker container image and deploys the test case container. From the user's point of view, the deployment of the test case requires only Docker software and interaction requires only HTTP requests.

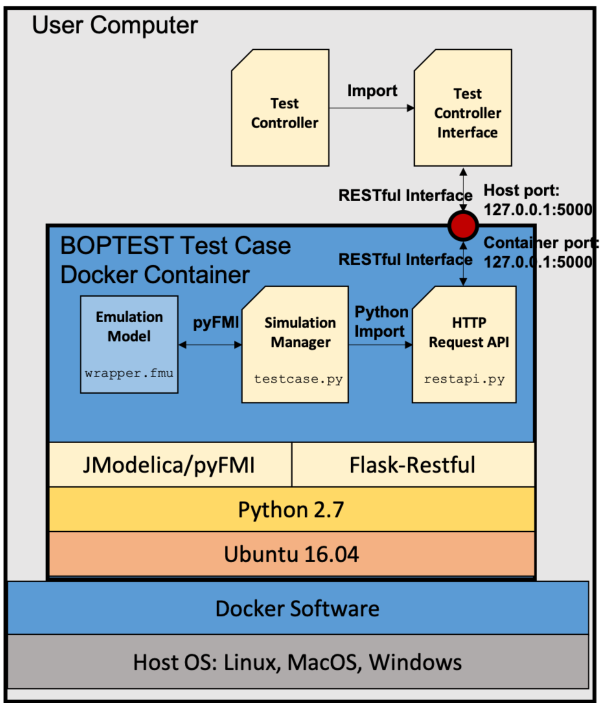


Figure 4. Prototype of BOPTEST software platform.

Controller Testing

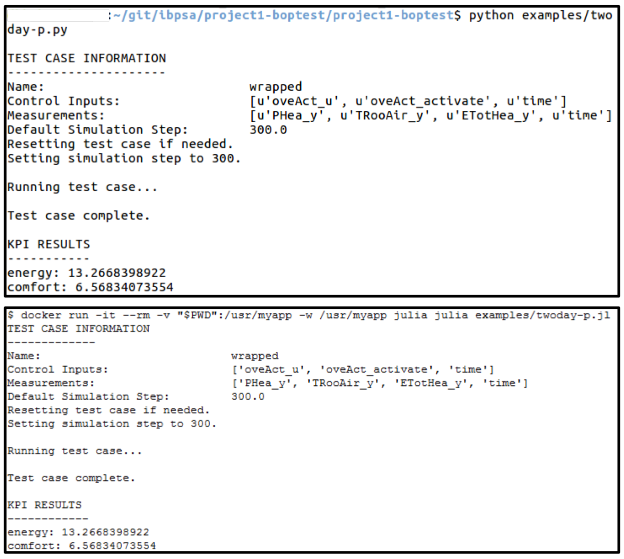
Interaction with a deployed test case is demonstrated with a simple test proportional controller to actuate the heater in response to zone temperature measurements and a setpoint of 20 °C. To demonstrate the combined flexibility and consistency of the software platform, the controller is written in two languages and tested on two computers. A Python controller test took place on an Ubuntu 16.04 virtual machine with Intel Core i7 processor where the controller was hosted on the virtual machine. A Julia controller test took place on an Ubuntu

16.04 virtual machine with Intel Xeon processor where the controller was hosted in a separate Docker container.

For each controller, two scripts have been written, one to implement the controller, and a second to implement a testing interface. The interface has four main steps:

1. Get test information - Uses the `/name`, `/inputs`, `/measurements`, `/step` GET requests to retrieve the test case name, available control inputs, available measurements, and current communication step.
2. Run test - Uses the `/advance` POST request with control signal data in the form of `{"oveAct_u":<value>,"oveAct_activate":1}` in a loop for the length of the test to advance the simulation forward one communication step, receive measurement data after the step is completed, and compute a control signal for the next step.
3. View results - Uses the `/kpi` GET request to retrieve the calculated KPIs. They are heater energy [*kWh*] and thermal comfort violation [*Kh*].
4. Post-process additional data - Uses the `/results` GET request to retrieve data trends and make plots.

After deploying the test case as described in the previous section (commit `f500b0b`), running the two controller tests for two days of simulation and a communication step of 300 seconds produces identical results, as presented in Figure 5.



```
:~/git/ibpsa/project1-boptest/project1-boptest$ python examples/two
day-p.py

TEST CASE INFORMATION
---------------------
Name:                           wrapped
Control Inputs:                 [u'oveAct_u', u'oveAct_activate', u'time']
Measurements:                   [u'PHea_y', u'TRooAir_y', u'ETotHea_y', u'time']
Default Simulation Step:        300.0
Resetting test case if needed.
Setting simulation step to 300.

Running test case...

Test case complete.

KPI RESULTS
-----------
energy: 13.2668398922
comfort: 6.56834073554
```

```
$ docker run -it --rm -v "$PWD":/usr/myapp -w /usr/myapp julia julia examples/twoday-p.jl
TEST CASE INFORMATION
-------------
Name:                           wrapped
Control Inputs:                 ['oveAct_u', 'oveAct_activate', 'time']
Measurements:                   ['PHea_y', 'TRooAir_y', 'ETotHea_y', 'time']
Default Simulation Step:        300.0
Resetting test case if needed.
Setting simulation step to 300.

Running test case...

Test case complete.

KPI RESULTS
------------
energy: 13.2668398922
comfort: 6.56834073554
```

Figure 5. Results of running example controller test interface in Python (top) and Julia (bottom).

Conclusion

This paper presents the development of a framework and software platform for simulation-based testing of ACS in buildings, called BOPTEST. We first outlined the requirements of the framework. Then, we presented the core components, including test cases, KPI specifications, and a software platform for deployment and interaction with controllers. Finally, we demonstrated prototypes of core pieces of the proposed architecture and an example test case.

Continued work includes development of a forecast module to retrieve boundary condition forecasts from the emulator for MPC testing, completed reference test case development, full implementation of KPI calculation and reporting, and full architecture implementation. Future considerations include the addition of stochastic occupant behavior and uncertainty in the forecasts of boundary conditions as well as use of the framework for testing automated fault detection and diagnosis and operator dashboard design and training.

Acknowledgement

This research was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Office of Building Technologies of the U.S. Department of Energy, under Contract No. DE-AC02-05CH11231.

The views expressed in the article do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

The financial support of the European Union's Horizon 2020 research and innovation programme under grant agreements No 723649 (MPC-;GT), No 731231 (FHP) and No 656889 (GEOTeCH), and the support of the Flemish Institute for Technological Research (VITO) through PhD Fellowship 1714 are acknowledged.

This work emerged from the IBPSA Project 1, an international project conducted under the umbrella of the International Building Performance Simulation Association (IBPSA). Project 1 will develop and demonstrate a BIM/GIS and Modelica Framework for building and community energy system design and operation.

References

Afram, A. and Janabi-Sharifi, F. (2014). Theory and applications of HVAC control systems: a review of model predictive control (MPC). *Building and Environment 72*, 343-355.

ANSI/ASHRAE (2007). Standard Method of Test for the Evaluation of Building Energy Analysis Computer Programs, Standard 140-2007, The American Society of Heating, Refrigerating and Air-Conditioning Engineers. 2007.

Blochwitz, T., Otter, M., Akesson, J., Arnold, M., Clau, C., Elmqvist, H., et al (2012). Functional mockup interface 2.0: the standard for tool independent exchange of simulation models. *Proceedings from 9th International Modelica Conference.* Munich (Germany), 3-5 September 2012.

Bushby, S.T., Galler, M.A., Ferretti, N.M., and Park, C. (2010). The virtual cybernetic building testbed - A building emulator. *ASHRAE Transactions 116*(1), 37-44.

Chan, A., Darko, A., Ameyaw, E., and Owusu-Manu, D. (2017). Barriers affecting the adoption of green building technologies. *Journal of Management in Engineering* *33*(3). https://doi.org/10.1061/(ASCE)ME.1943-5479.0000507.

Crawley, D.B., Lawrie, L.K., Winkelmann, F.C., Buhl, W.F., Huang, Y.J., Pedersen, C.O., Strand, R.K., Liesen, R.J., Fisher, D.E., Witte M.J., and Glazer, J. (2001). EnergyPlus: creating a new-generation building energy simulation program. *Energy and Buildings 33*(4), 319-331.

Haves, P. and L.K. Norford. 1997. A standard simulation testbed for the evaluation of control algorithms and strategies. ASHRAE 825-RP Final Report.

Huang, S., Wang, W., Brambley, M.R., Goyal, S., and Zuo, W. (2018). An agent-based hardware-in-the-loop simulation framework for building controls. *Energy and Buildings 181*, 26-37.

Husaunndee, A., Lahrech, R., Vaezi-Nejad, H., and Visier, J. C. (1997). SIMBAD: A simulation toolbox for the design and test of HVAC control systems. *Proceedings from 5th International IBPSA Conference*. Prague (Czech Republic), 8-10 September 1997.

International Energy Agency (IEA) (1997). Technical Synthesis Report: A summary of IEA Annexes 16 and 17 Building Energy Management Systems. Available online at http://www.iea-ebc.org/Data/publications/EBC_Annex_16-17_tsr.pdf. Last accessed Jan. 28, 2019.

Junker, G. R., Ghasem Azar, A., Amaral Lopes, R., Byskov Lindberg, K., Reynders, G., Relan R., Madsen, H. (2018). Characterizing the energy flexibility of buildings and districts. *Applied Energy 225*, 175-182.

Klein, S. A. et al, 2017, TRNSYS 18: A Transient System Simulation Program, Solar Energy Laboratory, University of Wisconsin, Madison, USA. Available online at http://sel.me.wisc.edu/trnsys. Last accessed Jan. 28, 2019.

Lahrech, R., Gruber, P., Riederer, P., Tessier, P., Visier, J.C. (2002). Development of a testing method for control HVAC systems by emulation. *Energy and Building 34*(9), 909-916.

Mattsson, S.E., and Elmqvist, H. (1997). Modelica – an international effort to design the next generation modeling language. *Proceedings from 7th IFAC Symposium on Computer Aided Control Systems Design*. Gent (Belgium), 28-30 April 1997.

Modelon AB (2017). JModelica.org User Guide: Version 2.4. Available online at https://jmodelica.org/downloads/UsersGuide-2.4.pdf.

Park, C., Clark, D.R., and Kelly, G.E. (1985). An overview of HVACSIM+: A dynamic building/HVAC/Control simulation program. *Proceedings from 1st Annual Building Energy Simulation Conference*. Seattle (USA), 21-22 August 1985.

Pernetti, R., Reynders, G., Knotzer, A., Østergaard Jensen, S., Madsen, H., Amaral Lopes, R., Junker, R., Aelenei, D., Li, R., Metzger, S., Lindberg, K., Marszal, A., Kummert, M., Bayles, B., Mlecnik, E. Lollini, R., Pasut, W. (2017). Annex 67: Energy Flexible Buildings Energy Flexibility as a key asset in a smart building future. Available online at http://annex67.org/media/1470/position-paper-energy-flexibility-as-a-key-asset-i-a-smart-building-future.pdf.

Riederer, P., Vaezi-Nejad, H., Husaunndee A., Bruyat, F. (2001). Development and quality improvement of HVAC control systems in virtual laboratories. *Proceedings from 7th International IBPSA Conference.* Rio de Janeiro (Brazil), 13-15 August 2001.

Rockett, P., and Hathway, E.A. (2017). Model-predictive control for non-domestic buildings: a critical review and prospects. *Building Research and Information 45*(5), 556-571.

Soethout, L., Arditi, I., Husaunndee, A., Macé, E., Marchio, D, et al., (1998). SIMTRAIN, A simulation package for BEMS education. *Proceedings from System Simulation in Buildings*. Liege (Belgium).

The MathWorks, Inc. (2000), MATLAB 6.0. Natick, Massachusetts, United States.

The Society of Heating, Air-Conditioning, and Sanitary Engineers of Japan (2019). World Championship in Cybernetic Building Optimization. Available online at http://www.wccbo.org/index_en.php. Last accessed Jan. 28, 2019.

Wetter, M. (2011). Co-simulation of building energy and control systems with the Building Controls Virtual Test Bed. *Journal of Building Performance Simulation* 4(3), 185-203.

Wetter M., Fuchs M., Grozman P., Helsen L., Jorissen F., Lauster M., Müller D., Nytsch-Geusen C., Picard D., Sahlin P. and Thorade M. (2015). IEA EBC Annex 60 Modelica Library - An international collaboration to develop a free open-source model library for buildings and community energy systems. *Proceedings from 14th IBPSA Conference*. Hyderabad (India), 7-9 December 2015.