AGENT-BASED CONTROL OF A NEIGHBORHOOD: A GENERIC APPROACH BY COUPLING MODELICA WITH PYTHON

Arnout Aertgeerts[1,2], Bert Claessens[2], Roel De Coninck[1,3] and Lieve Helsen[1,2]
[1]KU Leuven, Leuven, Belgium
[2]Energyville, Waterschei, Belgium
[3]3E, Brussels, Belgium

ABSTRACT

The research of optimal control in residential building clusters is approached from different disciplines: building simulation and control engineering. Control engineers focus mainly on the research and development of sophisticated optimal control strategies combined with high-level simulation tools but less accurate building models for fast prototyping of new control strategies. On the other hand, building simulation experts develop detailed building models which provide realistic and accurate building representations, however often in a simulation environment which is less suited for control.

This paper proposes several methodologies to extend building and cluster models in Modelica, an object-oriented modelling and simulation language, with a Python control layer in order to bridge the gap between both disciplines. Co-simulation tries to leverage the advantages of both approaches by enabling the combination of both in an integrated simulation and keeping the development of the building models and control strategies separate. Control algorithms developed in Python can then easily be tested on detailed models in Modelica. As such, the Modelica simulation model is used as an emulator or virtual test bed. These integrated co-simulations can provide new insights in the behavior of building clusters when using sophisticated control algorithms.

INTRODUCTION

Background and motivation

In the EU, building energy use accounted for 37% of the total energy use in 2004 (Pérez-Lombard, Ortiz, and Pout 2008) of which 50% is used by the heating, ventilation and air-conditioning (HVAC) installation of the building (Conti 2012). Furthermore, the increased integration of wind and solar energy imposes extra challenges for the power system due to the high intermittency of electricity production. The grid might face severe stability issues if corrective actions that can counteract this intermittency are not taken (Makarov et al. 2009). Optimal control strategies for the control of HVAC systems can be used to lower the energy use of buildings and are able to provide the power system with flexibility of the energy use which can be part of the solution to incorporate high amounts of renewable energy resources. Collaboration between the control group and the thermal systems group is necessary to obtain efficient control strategies and accurate results.

The control group focuses on the development of sophisticated optimal control schemes and algorithms but uses simple, low-order building models which represent reality in a simplified way (Koch 2012; Urban and Vermeulen 2011; Morvaj, Jurisic, and Holjevac 2013). These simple models allow the control group to rapidly develop and research new control or optimization algorithms for optimal control in buildings but give a simplified representation of reality. The building physics and thermal systems group focuses on accurate and realistic building models to study the exact potential and impact of control techniques but primarily uses local control algorithms without communication between dwellings or simple rule-based strategies for cluster control (Baetens et al. 2012; De Coninck et al. 2014; Reynders, Nuytten, and Saelens 2013). Software environments that are able to give accurate results of the building response to such control algorithms are becoming a necessity.

This paper proposes several methodologies to couple Python with Modelica and shows the importance of this tool coupling by implementing a central model predictive control (MPC) distributed by a multi-agent control system in Python on a residential neighborhood in Modelica. The results are promising: the test implementation shows improved control performance compared to rule-based control using a simple aggregated controller model for the MPC. Using this methodology, more complex and sophisticated MPC models could easily be implemented. Besides illustrating the methodology for interfacing both disciplines, this case study allows identifying the shortcomings and potential future work related to current simulation tools (Aertgeerts 2014).

Previous research

Co-simulation

Some efforts have been made to combine the research of control algorithms with accurate building models. First, the EnergyPlus (Crawley et al. 2001) framework provides accurate building models and can be extended with different toolboxes that allow co-simulation of control and building models (Wetter 2011). However, it focuses on the control of a single building and the control algorithms are often written in Matlab which is not freely available.Second, the Mosaik framework looks promising due to its wide scope and goal to provide models for all fields related to the Smart Grid in a co-simulation framework but remains in a conceptual phase for now (Schütte, Scherfke, and

Sonnenschein 2012). Third, the BCVTB provides a software environment that allows users to couple different simulation programs for co-simulation (Wen 2011) and can be a good alternative to the methodology proposed here but adds unnecessary complexity for control applications.

DSM of a small neighborhood

The proof of concept presented here is an extension of the work done by De Coninck et al. 2014 and compares rule-based control (RBC) strategies with a central model predictive control (MPC) of a neighborhood using a market-based multi-agent system (MAS) control scheme in a Python-Modelica co-simulation environment. The neighborhood suffers from over-voltages during periods characterized by high solar radiation due to the high penetration of PV systems resulting in curtailing losses. The inverters of the PV prevent these over-voltages by disconnecting the PV from the local feeder when this occurs. Following EN 50438:2007 the voltage limit is set at 253 V which is 10% over-voltage of the nominal voltage of 230 V at the local feeder. A more extensive literature review of DSM of building clusters is given by Aertgeerts 2014.

EXPERIMENT AND SIMULATION

Coupling Modelica with Python

In co-simulation, communication between both environments is mandatory and often the bottleneck in terms of simulation speed due to the overhead present in the communication. In this paper we assume a fixed communication step size. A Modelica environment simulates the model for this duration after which a control signal is computed in Python based on the current state of the model. This control signal is then fed back into the model and a new simulation is started. The smaller the communication step size, the more Modelica and Python need to communicate, increasing the overhead in the simulation.

We explore three options for a simple but effective coupling between Modelica and Python:

1. **FMU**: Python is used as the master and the Modelica model is compiled to a Co-Simulation FMU using the CVOde solver which can be run in Python.
2. **Dymosim**: Python is used as the master and the Modelica model is run as a dymola executable called dymosim.exe in a new external process after every simulation step.
3. **C-function**: The Modelica simulator is used as the master and a python function can be run by calling an external C-function which spawns a new Python interpreter after every simulation step.

The functional mockup interface (FMI) is a tool independent standard for the exchange of dynamic models and co-simulation (Blochwitz et al. 2011). Modelica models, such as the considered neighborhood, can be exported to a Functional Mockup Unit (FMU) using the FMI standard. FMUs have the advantage to be tool independent and can be run by any software that supports the FMI standard (for example PyFMI by JModelica.org in Python). Modelica parameters such as sensors and control variables of modeled devices can then be read and set easily using a Python FMU object, creating minimum overhead. However, it is very expensive to use the high performant Dymola solvers in a dedicated co-simulation FMU making the current Dymola generated FMUs less robust then Dymola. Furthermore, it is not always possible to compile complex models to an FMU.

The Dymosim method uses the commercial program Dymola to export the Modelica model as an executable file called dymosim which can run on a Windows or Linux environment (Dynasim 2013). This executable file is the same as used by Dymola and can use the high performant solvers without extra cost making it extremely robust. The Modelica parameters are read and set by reading and writing a text file. For large models such as the neighborhood investigated in our study, this text file can become large (e.g. 55 MB) and creates a large overhead every communication step. Dymosims can only be exchanged across machines with a normal Dymola license which is needed to run the executable or on any machine when the dymosim is exported using the binary model export license.

By using an external **C-function** to call a Python function from within the Modelica simulator, all benefits of the optimized Modelica simulator can be exploited (e.g. the high performant Dymola solvers). The technique used here is implemented in the Modelica buildings library by Wetter et al. 2014. The models can easily be exchanged across machines as simple Modelica files and run in any Modelica simulator.

Method	Python	Modelica
FMU	Master	Separate process
Dymosim	Master	.txt file
C-function	file	Master

Table 1: State persistent methods for the different co-simulation methods.

When doing co-simulation, the persistence of state of the program in some form is an important feature in order for both environments to know where to start the next simulation or control cycle. For Modelica, it is important to know where to start the next simulation to ensure continuity while a sophisticated control in Python might need information of previous optimization cycles (e.g. a hysteresis control or a Kalman Filter). This persistence of state can be handled in different ways such as reading and writing a file or keeping a process alive for the entire duration of the simulation and plays an important role in the communication overhead between both software environments.

Table 1 gives a concise overview of how the different co-simulation methods persist the state of both Python and Modelica. Of course, the state is always automatically preserved in the master.

Note that the dymosim and C-function method as implemented by the buildings library are unable to keep a separate process alive and thus need to read and write files to persist the state. These files are especially heavy to persist the simulation (Modelica) state because they need to incorporate all state information of the previous simulation cycle whereas the control state is almost always several orders smaller and thus more easily read and set creating less overhead. Furthermore, the implementation of the buildings library needs to start a python interpreter in each call slowing down the simulation further.

The simulation speed of the three methods is compared using two detailed three zone building structure models where each zone has a window. Heat is injected into the first model as an input signal where in the second model an extra layer of complexity is added by modelling a detailed hydronic heating system. The single building model consists of 153 states and 3332 equations and the building model with heating system consists of 160 states and 8857 equations. A simple hysteresis control between 20°C and 21°C is implemented in Python to control the heating input to the building. The response of both models to a hysteresis input with a discrete step size of 900 seconds for the period of one week is shown in figure 1.

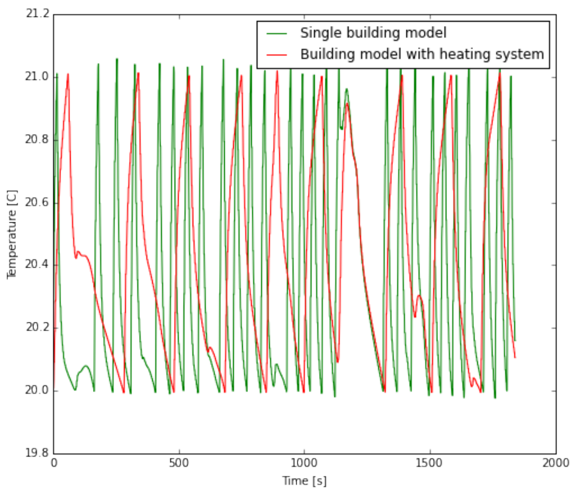


Figure 1: Response of a single building model (green) and building model with heating system (red) to a hysteresis control input.

Five different communication step sizes are simulated: 60s, 120s, 900s, 1800s and 3600s. Simulations are done on a Dell Latitude E6540 with a 2.60GHz processor and the CVOde by Hindmarsh et al. 2005 integrator is used for all simulations.

Building model

The open source Modelica modeling environment for Integrated District Energy Assessment by Simulations or IDEAS library allows simultaneous transient simulation of thermal and electrical systems at both building and feeder level (Baetens et al. 2012). The IDEAS modeling environment is already widely tested and used in several publications (Baetens et al. 2012; Reynders, Nuytten, and Saelens 2013; De Coninck et al. 2014). Baetens et al. frame the IDEAS modeling and simulation environment well within the broader picture of simulation environments.

The neighborhood model differs from the models used for the speed evaluation of the co-simulation techniques but is exactly the same neighborhood as presented in detail by De Coninck et al. 2014 and quickly reviewed here. It consists of 33 identical dwellings connected to a 34-node feeder as standardized by IEEE (Kersting 2001).

Dwelling

All dwellings are identical and thus use the same topology. A dwelling consists of one thermal zone, is designed according to a low-energy standard and has massive walls and floors. Stochastic variations in temperature and load are provided by a probabilistic occupant behavior to counteract the fact that the model uses 33 identical dwellings.

HVAC System

The heating system of each dwelling consists of floor heating and a domestic hot water (DHW) system. Heat is produced by a modulating air-to-water heat pump (HP) which can consume a variable power. The HP produces heat for either the floor heating by controlling a FH pump or for the DHW tank by controlling a DHW pump. Cooling is not considered, which is justified by the Belgian climate.

Hot water is drawn from the DHW storage tank via a thermostatic mixing valve at 45°C and new cold water is injected into the tank. The HP consumes power whenever there is a flow through the HP depending on the temperatures of the input and output flow. Notice that controlling the HP using a hydraulic pump results in a variable power consumption of the HP that needs to be estimated. This research focusses on the control of the DHW pumps to heat the DHW storage tanks.

Electricity generation

Each dwelling has a building integrated solar panel (BIPV) system which is sized to cover, on a yearly basis, the total electricity consumption of the dwelling as would be present in a modern neighborhood. The electrical connection between the dwellings BIPV system and the feeder is protected by switching-off the inverter when the voltage reaches a predefined limit due to a large power production of the BIPV. Preventing this switching behavior is the objective of this research by using the thermal flexibility of the DHW storage tanks.

Occupant behavior

Assessments based on purely deterministic baseline models do not adequately represent the statistical nature of the impacts of many practical load systems making stochastic occupant behavior mandatory for Smart Grid research (Urban and Vermeulen 2011). Four different types of occupants are modelled which all show stochastic behavior on household electricity loads, domestic hot water consumption and presence.

Optimal control algorithm

In this proof of concept, the neighborhood is controlled by a market-based multi-agent system which uses an aggregated central MPC to control different houses. It is impossible to implement this control solely in Modelica without interacting with optimization capable software such as Python. The operation of this optimal control algorithm is described below.

Market-based multi-agent system

In a market-based MAS every application is represented by a device agent. This device agent is responsible for (i) creating a bidding curve of a distributed energy resource (DER), (ii) sending it to a central control, called the business agent, who finds a market optimum using an optimal control algorithm, (iii) evaluating the bidding curve when the market optimum is found and (iv) controlling the DER accordingly. An important advantage of the market-based MAS control scheme is the use of a bidding curve instead of a single bid. A curve is able to send all of the negotiation information in one communication step making the control lightweight.

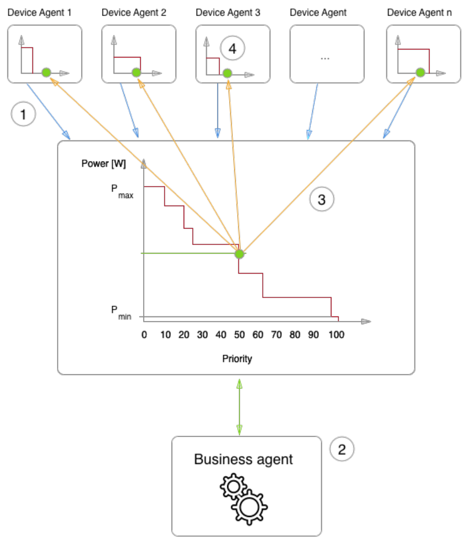


Figure 2: Five step process of the multi-agent market-based system

The DHW storage tanks present in the neighborhood are the applications used for DSM. The device agents create the bidding curve of the DHW storage tanks by using a step curve with a breakpoint at the state temperature relative to the comfort boundaries as a function of a priority p. This priority can be understood as "the willingness" of a tank to consume power from the device agent's point of view and varies from 0 to 100. A priority of zero means the tank is unable to consume power whereas a priority of 100 represents a state where the device is obliged to consume power. In both cases the tank shows no flexibility: a tank with a state out of the comfort boundaries cannot be used for DSM. The step curve results from the assumption that a HP is either on or off and consumes P_{nom} when the HP is on.

Let T_M be the maximum comfort temperature, T_m the minimum comfort temperature and T_s a measured temperature which represents the state temperature at the top of the water tank. Note that we use a single sensor value and not the real state temperature that is a primitive approximation of the state of the water tank which consists of different layers at different temperatures. Vanthournout et al. (2012) present a more rigorous manner to find the state of a DHW storage tank and the power consumption which can be used in further research. The resulting bidding curve $P(y)$ with P the power of the HP for the next simulation step and y the priority from 0 to 100 is:

$$P(y) = \begin{cases} P_{nom} & : y < 100\frac{T_s - T_m}{T_M - T_m} \\ 0 & : y > 100\frac{T_s - T_m}{T_M - T_m} \end{cases} \tag{1}$$

The parameters used in equation 1 are summarized in table 2. The fixed nominal power is an approximation of the real power which depends on additional external factors such as the efficiency of the HP, the current state of the thermal storage tank and more.

Variable	Unit	Value
T_M	[K]	333
T_m	[K]	325
T_s	[K]	Tank top temperature
P_{nom}	[W]	1200

Table 2: Bidding curve algorithm parameters

The business agent is the top level agent to which all device agents are connected and is responsible for determining and communicating the market equilibrium. Each device agent sends his bidding curve to the business agent who accumulates the curves and solves an optimization problem to find the market equilibrium. The control is implemented in a total of five steps which are shown in figure 2. The five steps are:

1. Every device agent creates his bidding curve and sends it to the business agent (blue arrows).
2. The business agent aggregates the bidding curve and finds the optimal priority by solving an optimization problem (green dot).
3. The optimal priority is broadcasted to all device agents (orange arrows).

4. Every device agent evaluates his bidding curve using the optimal priority to obtain the power set point for the next control step.
5. Every device agent consumes or produces his power set point for a time period.

Central MPC of the neighborhood
As stated before, to control the neighborhood a central MPC is used. Within this MPC we use a simple linear aggregated model, derived from the thermodynamic equation for sensible heat, for the state estimation of the DHW storage tanks in each discrete time step i. The model uses a single state aggregated state of all water tanks θ_i and takes one disturbance into account: the outside temperature θ_{out}. Parameter identification of the model is done using least square error minimization (LSQE) through the Levenberg-Marquardt-Fletcher algorithm (Wang et al. 2013). Every time step has a duration of 15 minutes and the number of time steps in the horizon of the MPC is called N.

$$\theta_{i+1} = \theta_i + a(\theta_i - \theta_{out,i}) + cu_i \tag{2}$$

The controllable input of the MPC is the total power consumption of all DHW storage tanks u_i for every control time step i and is the focus of the optimization. Information available to the MPC optimizer in addition to the aggregated model are (i) temperature measurements of the tanks, (ii) perfect predictions of the disturbances, (iii) the bidding curves of the device agents, and (iv) the thermal comfort boundaries. These four bits of information result in the boundary conditions of the optimization problem. The average of the tank temperature measurements for each dwelling k and a total amount of dwellings equal to 33 is used as the initial state temperature for the aggregated model .

$$\theta_0 = \frac{\sum_{k=1}^{33} \theta_{k,tank}}{33} \tag{3}$$

The perfect predictions of the disturbance give exact values for every discrete time step of the MPC future horizon.

$$\forall i \in [0, N-1] : \theta_{out,i} = \theta_{out,i,\text{ predicted}} \tag{4}$$

The accumulated bidding curve of the device agents defines a minimum and a maximum power consumption for the first control time step (figure 2). Let $P_a(p)$ be the sum of all bidding curves:

$$P_a(p) = \sum_{k=1}^{33} P_k(p) \tag{5}$$

DHW storage tanks with a temperature below the comfort zone impose a minimum power consumption and DHW storage tanks with a temperature above the comfort zone cannot consume power at all. The maximum power consumption, P_M, and minimum power consumption, P_m of the aggregated model are then given by:

$$\begin{aligned} u_0 &\leq P_M = P_a(0) \\ u_0 &\geq P_m = P_a(100) \end{aligned} \tag{6}$$

Bidding curves are only made for the first time step ($i = 0$) because this is the only state known by the device agents. The aggregated model does not provide a state prediction for each device individually which makes these boundary conditions only valid for the first time step. Boundary conditions for the subsequent time steps are set by a maximum power equal to the maximum power consumption of all HPs and a minimum of zero, assuming all tanks are flexible.

$$\begin{aligned} &\forall i \in [1, N-1] : u_i \leq 33 \cdot P_{nom} = 39600 \text{ W} \\ &\forall i \in [1, N-1] : u_i \geq 0 \end{aligned} \tag{7}$$

Finally, every state temperature except the first one is limited by the comfort boundaries. No boundary conditions are set on θ_0 because the MPC is unable to control this temperature and the problem would be immediately infeasible if θ_0 falls out of the comfort zone. The boundary conditions for the other state temperatures are:

$$\forall i \in [1, N] : T_M \geq \theta_i \geq T_m \tag{8}$$

Given the above statements, the total MPC optimization problem can be formulated with a quadratic objective:

$$\begin{aligned} \min \quad & J = \sum_{i=0}^{N-1} (u_i + P_{occ,i} - P_{PV,i})^2 + W \cdot u_i \\ \text{s.t.} \quad & \text{equations [2-8]} \end{aligned} \tag{9}$$

In which $P_{occ,i}$ is the total predicted electric power consumption in time step i for the occupants of all dwellings k, $P_{PV,i}$ the total predicted solar panel electricity production in time step i of all dwellings k and W a weighting factor. The objective tries to maximize the electric power consumption during periods of high PV production while the weighting factor W ensures no useless power is consumed during periods of high PV production and a high state temperature of the aggregated model. Thanks to the predictions, the MPC can shift the neighborhood's power consumption to minimize the objective function by controlling the consumption of the HPs.
Simulations are done for two periods of three days: a winter period from 2011-03-18 until 2011-03-21 and a summer period from 2011-07-22 until 2011-07-25. Both periods experience curtailing problems due to high solar radiation leading to high power production of the PV systems. The dymosim co-simulation method was the only method robust enough to simulate the complex model but slowed down the simulation speed significantly. Although the simulation periods are limited, the periods are chosen to show high overproduction of electricity and are sufficient for the

Name	Trigger	Rules
Clock$_{\text{DHW}}$	Clock	Between 12h00 and 16h00 the set temperature of the DHW storage tank is increased.
VGrid$_{\text{var}}$	Voltage at dwelling's grid connection	When the voltage surpasses V_{lim}, the set temperature of the DHW storage tank is increased. The voltage limit depends on the position of the dwelling in the grid.

V_{lim} was set to 251 V, 2 V lower than the shutdown limit to create a safety margin.

Table 3: Overview of the RBC strategies used in the comparison (De Coninck et al. 2014).

comparison of the different control strategies in this proof of concept.

RESULTS AND DISCUSSION

Simulation speed comparison

Figure 3 shows the simulation speed as a function of the communication step size of the hysteresis control input for the different co-simulation methods compared with a discrete hysteresis implemented solely in Modelica (blue line Dymola) for the detailed building and figure 4 does this for the detailed building with the heating system. Note that the simulation time of the dymosim co-simulation method is not included in these figures due to a very high simulation time of over 300s, even for a hysteresis time step of 3600s due to the severe overhead in persisting the Modelica state with text files.

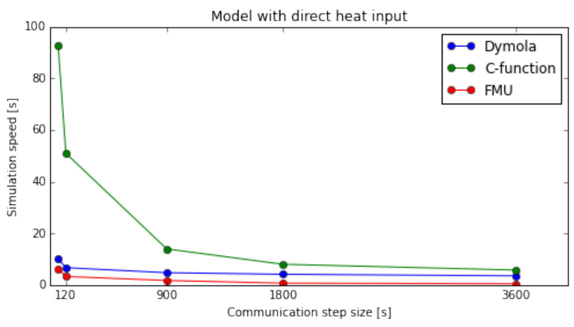


Figure 3: Simulation speed for the different cosimulation methods for the detailed building.

The simulation time in these models is largely dependent on two main factors: (i) The hysteresis step size and (ii) the overhead when communicating between the two software environments. First, a larger hysteresis step size implies less calculations of the hysteresis control equations and less communication steps between the two environments. For very small time steps (about 60s) this can have an effect even on the Modelica simulation without communication with Python. Second, the communication overhead slows down the simulation every time Python has to communicate with Modelica or vice-versa and leads to a fixed time increase every communication step. This effect kills the performance of the dymosim and (to a lesser degree) C-function method for small step sizes due to a high overhead every time step but deteriorates with an increase in step size.

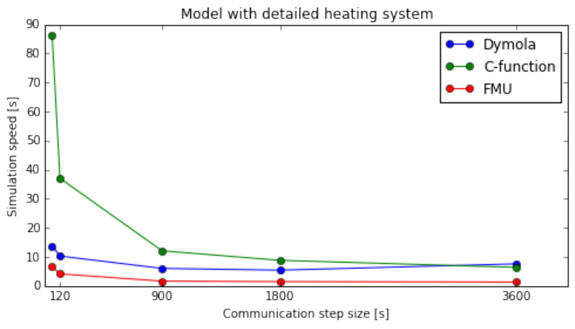


Figure 4: Simulation speed for the different co-simulation methods for the detailed building model with heating system.

All simulations show a decrease in simulation time with an increase in hysteresis step size with a limited effect for the Dymola simulation where only the hysteresis step size plays an important role and the communication overhead is non existing leading to a constant simulation time for step sizes above 900 s. For all co-simulation methods, increasing the hysteresis step size has a large effect on the simulation time due to the decrease in communication overhead.

The FMU method shows the lowest communication overhead of all co-simulation methods and simulates even faster with a hysteresis control implemented in Python than the pure Modelica simulation, implying that there is no slowing-down effect when doing co-simulation with an FMU. This can be contributed due to the fact that the state of both Modelica and Python is constantly persisted in separate processes. The C-function method comes close to the Dymola simulation for larger hysteresis time steps indicating a reasonable effect of overhead for a communication cycle due to the reading and writing of the Python state.

Of course these results heavily depend among others on the integrator choice, model structure but still give a good indication of the overhead for the different co-simulation methods and are able to capture the general tendencies.

DSM with a central MPC

Figure 5 shows the loss-benefit space, presented by De Coninck et al. 2014 for rule based control (RBC) strategies, and here extended to MPC strategies,, which is our base for the comparison of DSM control strategies and characterized by two components. The first component lies on the horizontal axis and represents the relative increase in electricity demand compared to a reference case which lies on the origin. The second component is the relative reduction in curtailing and ohmic losses which lies on the vertical axis and represents the gain in electricity production due to prevention of curtailing losses. Both components are equally important for the evaluation of a control strategy. The resulting net electricity saving compared to a reference case ΔE_{NBH} is the difference between the relative increase in electricity production and consumption. A control strategy for which the relative consumption equals the relative extra PV production lies on the status-quo line and is not better compared to the reference case which lies in the origin. The reference case applies a simple hysteresis control to the temperature of the DHW storage tank to ensure the tank stays between 53°C and 55 °C, as is a common implementation in current households.

The RBC strategies that are used in the comparison are listed in table 3 (De Coninck et al. 2014). All control strategies are represented by a colored dot corresponding to the strategy and a number in the dot. For RBC strategies, this number gives the increase in set temperature when the control is active relative to 55°C. Dots that correlate to an MPC strategy have a number which represents the upper thermal comfort boundary relative to 60°C. For example, a dot that has zero inside corresponds to a control with 52°C and 60°C as comfort boundaries whereas a dot with an 8 inside means the thermal comfort boundaries of the DHW storage tank are 52°C and 68°C. We tested two different horizons for the MPC: MPC$_{12}$ has a prediction horizon of 12 hours and MPC$_{24}$ has a prediction horizon of 24 hours, corresponding to $N = 48$ and $N = 96$ respectively.

Figure 5 shows that the MPC$_{24}$ control outperforms all RBC strategies in both periods, although this is less the case for the summer period where only MPC$_{24,8}$ outperforms the RBC strategies. The MPC experiences almost no increase in power consumption while decreasing the PV curtailing losses with more than 7% during the winter period and almost 10% during the summer period. The predictive nature of the MCP control uses the thermal flexibility of the DHW storage tanks at ideal times to reduce curtailing as much as possible.

CONCLUSION

This paper describes three methods that can be used for the co-simulation of Modelica and Python. The FMU method has shown to be the fastest method in terms of simulation speed but can fail during the creation of the FMU or due to solver issues. The C-function and especially the dymosim method show a large overhead every communication cycle but are able to use the Modelica simulation program solvers and are thus more robust than the FMU method. In general, when performing a co-simulation between Python and Modelica, we advice to try the different methods in the following order of decreasing simulation speed and increasing robustness: (i) FMU, (ii) C-function and (iii) the dymosim method. Finally, the Modelica language standardizes a so-called ”ExternalObject” through which a pointer to memory can be passed from one function call to another. Hence, by optimizing the implementation of the C-function in the buildings library, this could probably be used to avoid preserving the state in a file and prevent the method from starting a python interpreter each step.

Due to the complexity of the neighborhood model used in the proof of concept, the dymosim method was used as the co-simulation technique to implement a central MPC in a market-based MAS setting. The central MPC control of the neighborhood clearly outperforms the other RBC strategies using a simple aggregated model and optimization objective to reduce the curtailing losses. This is especially the case when there were unexpected times of PV production or when the flexibility of the tanks was limited. When this happens, the MPC is able to choose the ideal moment of power consumption, optimizing the use of the limited flexibility.

ACKNOWLEDGEMENT

The authors gratefully acknowledge the Agency for Innovation by Science and Technology in Flanders (IWT) for funding the ’Proeftuin Schipjes’ project (contract 140125) and the European commission for supporting the FP7 project PerformancePlus (contract 308991).

References

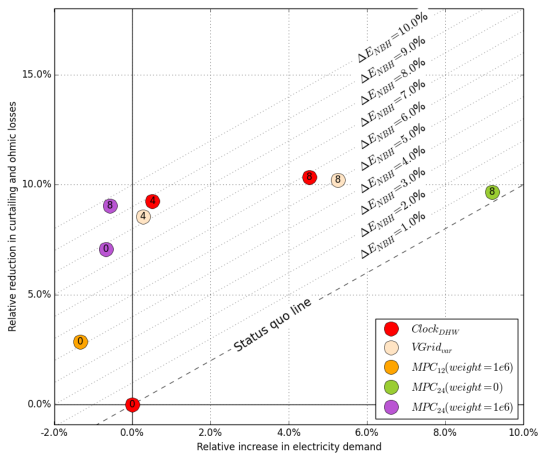
Aertgeerts, Arnout (2014). “Demand side management of the thermal flexibility in a residential neighborhood using a hierarchical market-based multi-agent system”. MA thesis. KU Leuven, Belgium.

Baetens, R. et al. (2012). “Assessing electrical bottlenecks at feeder level for residential net zero-energy buildings by integrated system simulation”. In: *Applied Energy* 96. Smart Grids, pp. 74–83. ISSN: 0306-2619.

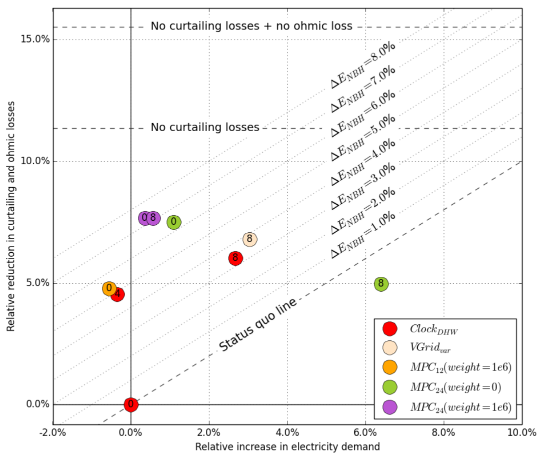
Blochwitz, Torsten et al. (2011). “The functional mockup interface for tool independent exchange of simulation models”. In: *8th International Modelica Conference, Dresden*, pp. 20–22.

Conti, John J. (2012). *Annual Energy Outlook 2012*. Tech. rep. U.S. Energy Information Administration.

Crawley, Drury B et al. (2001). “EnergyPlus: creating a new-generation building energy simulation program”. In: *Energy and Buildings* 33.4, pp. 319–331.



(a) Summer period.



(b) Winter period.

Figure 5: Loss benefit space for the RBC and MPC control strategies.

De Coninck, R. et al. (2014). "Rule-based demand-side management of domestic hot water production with heat pumps in zero energy neighbourhoods". In: *Journal of Building Performance Simulation* 7.4, pp. 271–288.

Dynasim (2013). *Dymola Dynamic Modeling Library User's Manual*. 5.3 a. Dynasim AB. Research Park Ideon SE-223 70 Lund Sweden.

Hindmarsh, Alan C et al. (2005). "SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers". In: *ACM Transactions on Mathematical Software (TOMS)* 31.3, pp. 363–396.

Kersting, William H (2001). "Radial distribution test feeders". In: *Power Engineering Society Winter Meeting, 2001. IEEE*. Vol. 2. IEEE. Columbus, OH, USA, pp. 908–912.

Koch, Stephan (2012). "Demand Response Mehods for Ancillary Sevices and Renewable energy Integraton in Electric Power Systems". PhD thesis. ETH Zurich.

Makarov, Y.V. et al. (2009). "Operational Impacts of Wind Generation on California Power Systems". In: *Power Systems, IEEE Transactions on* 24.2, pp. 1039–1050. ISSN: 0885-8950.

Morvaj, Boran, B Jurisic, and Ninoslav Holjevac (2013). "Stochastic simulation of the smart grid and demand response implementations on a city-wide scale". In: *International Convention on Information & Communication Technology Electronics & Microelectronics (MIPRO), 2013 36th*. IEEE. Opatija, Croatia, pp. 1241–1246.

Pérez-Lombard, Luis, José Ortiz, and Christine Pout (2008). "A review on buildings energy consumption information". In: *Energy and buildings* 40.3, pp. 394–398.

Reynders, Glenn, Thomas Nuytten, and Dirk Saelens (2013). "Potential of structural thermal mass for demand-side management in dwellings". In: *Building and Environment* 64, pp. 187–199.

Schütte, Steffen, Stefan Scherfke, and Michael Sonnenschein (2012). "mosaik-Smart Grid Simulation API". In: Porto, Portugal, pp. 14–24.

Urban, G. J. and H. J. Vermeulen (2011). "The statistical Modelling of Residential Electrical Demand for the Evaluation of Impacts that Result from Demand Side Management Interventions". In: *Universities' Power Engineering Conference (UPEC), Proceedings of 2011 46th International*, pp. 1–6.

Vanthournout, K. et al. (2012). "A Smart Domestic Hot Water Buffer". In: *Smart Grid, IEEE Transactions on* 3.4, pp. 2121–2127. ISSN: 1949-3053.

Wang, Xiaoyu et al. (2013). "Parameter identification of doubly-fed induction generator by the Levenberg-Marquardt-Fletcher method". In: *Power and Energy Society General Meeting (PES), 2013 IEEE*. Vancouver, Canada, pp. 1–5.

Wen, Yao-Jung (2011). "Rapid-prototyping control implementation using the building controls virtual test bed". In: *Philips Res. North Amer., Briarcliff Manor, NY, USA, Tech. Rep*.

Wetter, Michael (2011). "Co-simulation of building energy and control systems with the Building Controls Virtual Test Bed". In: *Journal of Building Performance Simulation* 4.3, pp. 185–203.

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