
ERNEST ORLANDO LAWRENCE
BERKELEY NATIONAL LABORATORY

# Equation-based languages– A new paradigm for building energy modeling, simulation and optimization

Environmental Technologies Area

October, 2015

LBNL-Technical Report
Cite as: Michael Wetter, Marco Bonvini and Thierry S. Nouidui.
Equation-based languages - A new paradigm for building energy modeling, simulation and optimization.
*Energy and Buildings*, 2015.
DOI:10.1016/j.enbuild.2015.10.017

# Equation-based languages– A new paradigm for building energy modeling, simulation and optimization

Michael Wetter, Marco Bonvini, Thierry S. Nouidui

Lawrence Berkeley National Laboratory
Energy Technologies Area
Building Technology and Urban Systems Department
Simulation Research Group
Berkeley, CA 94720, USA

October 13, 2015

*Abstract*

Most of the state-of-the-art building simulation programs implement models in imperative programming languages. This complicates modeling and excludes the use of certain efficient methods for simulation and optimization. In contrast, equation-based modeling languages declare relations among variables, thereby allowing the use of computer algebra to enable much simpler schematic modeling and to generate efficient code for simulation and optimization.

We contrast the two approaches in this paper. We explain how such manipulations support new use cases. In the first of two examples, we couple models of the electrical grid, multiple buildings, HVAC systems and controllers to test a controller that adjusts building room temperatures and PV inverter reactive power to maintain power quality. In the second example, we contrast the computing time for solving an optimal control problem for a room-level model predictive controller with and without symbolic manipulations. Exploiting the equation-based language led to 2,200 times faster solution.

*Disclaimer*

## 1 Introduction

To meet increasingly stringent energy performance targets and challenges posed by distributed renewable energy generation on the electrical distribution grid, recently more attention is given to system-level integration, part-load operation and operational optimization of buildings. The intent is to design and operate a building or a neighborhood optimally. This requires taking into account system-level interactions between building storage, HVAC systems and electrical grid. Such system-level analysis requires multi-physics simulation and optimization using coupled thermal, electrical and control models. Optimal operation also requires closing the gap between designed and actual performance through commissioning, energy monitoring and fault detection and diagnostics. All these activities can benefit from using models that represent the design intent. These models can then be used to verify responses of installed equipment and control sequences, and to compute optimal control sequences in a Model Predictive Controller (MPC), the latter possibly after simplifying the model.

This shift in focus will require an increased use of models throughout the building delivery stages and continuing into the operational phase. For example, during design, a mechanical engineer will construct a model that represents the design intent. To reduce cost for implementation of the control sequence, and to ensure that the control intent is properly implemented, a control model could be used to generate code that can be uploaded to supervisory building automation systems, thereby executing the same sequence as was used during design[1]. During commissioning, the design model will be used to verify proper installation. During operation, the model will be used for monitoring actual with expected energy use[2], and for fault detection and diagnostics[3]. Also, model calibration offers an opportunity to diagnose why performance as-designed and as-installed differ. Furthermore, the model can be converted to a form that allows its use during operation as part of a MPC algorithm.

[1] Nouidui and Wetter, 2014

[2] Pang et al., 2011

[3] Bonvini et al., 2014

In addition to the focus on closing the performance gap between design and operation, another recent focus is to evaluate how building dynamics, HVAC, thermal and electrical storage, renewable energy generation and grid responsive control affect the electrical grid[4]. Models that integrate building loads, HVAC and electrical systems can be used to develop control sequences that attempt to ensure power quality.

[4] Baetens et al., 2012; and Bonvini et al., 2014

For a larger discussion of functionalities that future building modeling tools will need to provide to address the needs for low energy building and community energy grid design and operation, we refer

to Wetter (2011) and Clarke (2015).

For the aforementioned new foci, the following new needs are emerging for building simulation tools:

1. Mechanical engineers should be able to design, assess the performance and verify the correctness of local and, in particular, supervisory control sequences in simulation. They should then use such a verified, non-ambiguous specification to communicate their design intent to the control provider. Moreover, the specification should be used during commissioning to verify that the control contractor implemented the design intent.

2. Controls engineers should be able to extract subsystem models from models used during the building design in order to use them within building control systems for commissioning, model-based controls, fault detection and diagnostics.

3. Urban planners and researchers should be able to combine models of buildings, electrical grids and controls in order to improve the design and operation of such systems that ensure high performance in terms of greenhouse gas emissions or cost, while ensuring power quality of the grid[5].

4. Mechanical engineers should be able to convert design models to a form that allows the efficient and robust solution of optimal control problems as part of MPC[6]. Such models may then be combined with state estimation techniques that adapt the model to the actual building[7].

[5] Baetens et al., 2012; and Bonvini et al., 2014

[6] Širokỳ et al., 2011

[7] Bonvini et al., 2014

The first item requires modeling and simulation of actual control sequences, including proper handling of hybrid systems, i.e., systems in which the state evolves in time based on continuous time semantics that arises from physics, and discrete time and discrete event semantics that arises from digital control. The second item requires extraction of a subsystem model and exporting this model in a self-contained form that can readily be executed as part of a building automation system. The third item requires models of different physical domains and models of control systems to be combined for a dynamic, multi-physics simulation that involves electrical systems, thermal systems, controls and possibly communication systems. The fourth item greatly benefits if model equations are accessible to perform model order reduction and to solve optimal control problems. In this paper, we will focus on the third and fourth items. For the first and second items, seeWetter (2009), Wetter et al. (2014) and Nouidui and Wetter (2014), respectively.

The contributions of this paper are (i) to explain how equation-based languages for multi-physics systems can address needs for

design, operation and dynamic analysis of low energy systems coupled to reneweable energy generation and transmission, (ii) to show how Modelica models for building envelope, generated from OpenStudio input files, can be linked to Modelica models for HVAC and electrical systems to develop a controller that adjusts building temperature and PV inverter reactive power to maintain power quality, and (iii) to show how Modelica models can be used to efficiently solve optimal control problems that minimize energy use subject to comfort constraints.

## 2 *Comparison to State-of-the-art in Building Energy Modeling and Simulation*

Today's whole building simulation programs formulate models using imperative programming languages. Imperative programming languages assign values to functions, declare the sequence of execution of these functions and change the state of the program, as is done for example in C/C++, Fortran or MATLAB/Simulink. In such programs, model equations are tightly intertwined with numerical solution methods, often by making the numerical solution procedure part of the actual model equations. This approach has its origin in the seventies when neither modular software approaches were implemented nor powerful computer algebra tools were available. These programs have been developed for the use case of building energy performance assessment to support building design and energy policy development. Other use cases such as control design and verification, model use in support of operation, and multi-physics dynamic analysis that combines building, HVAC, electrical and control models were no priorities or not even considered[8]. However, they recently gained importance[9].

[8] Crawley et al., 1996

[9] Clarke, 2015

Tight coupling of numerical solution methods with model equations and input/output routines makes it difficult to extend these programs to support new use cases. The reason is that this coupling imposes rules that determine for example where inputs to functions that compute HVAC, building or control equipment are received from the internal data structure of the program, when these inputs are updated, when these functions are evaluated to produce new output, and what output values may be lagged in time to avoid algebraic loops. Such rules have shown to make it increasingly difficult for developers to add new functionalities to software without inadvertently introducing an error in other parts of the program. They also make it difficult for users to understand how component models interact with other parts of the system model, in particular their interaction with, and assumptions of, control sequences. Furthermore, they also

have shown to make it difficult to use such tools for optimization[10].

[10] Wetter and Wright, 2004

The tight coupling of numerical solution methods with model equations makes it also difficult to efficiently simulate models for the various use cases, the reason being that the numerical methods in todays' building energy simulation programs are tailored to the use case of energy analysis during design. However, other use cases such as controls design and verification, coupled modeling of thermal and electrical systems, and model use during operation require different numerical methods. To see why different numerical methods are required, consider these applications:

*Stiff systems:* The simulation of feedback control with time constants of seconds coupled to building energy models with time constants of hours leads to stiff ordinary differential equations. Their efficient numerical solution requires implicit solvers[11].

[11] Hairer and Wanner, 1996

*Non-stiff systems:* In EnergyPlus and in many TRNSYS component models, HVAC equipment and controllers are generally approximated using steady-state models, resulting in algebraic equations. Hence, the resulting system model is not stiff as the only dynamics is from the building model. In this situation, explicit time integration algorithms are generally more efficient[12].

[12] Jorissen et al., 2015

*Hybrid systems:* Hybrid systems require proper simulation of coupled continuous time, discrete time and discrete event dynamics. This in turn requires solution methods with variable time steps and event handling. For example, when a temperature sensor crosses a setpoint or a battery reaches its state of charge, a state event takes place that may switch a controller, necessitating solving for the time instant when the switch happens and reducing accordingly the integration time step. Standard ordinary differential equation solvers require an iteration in time to solve for the time instant of the event, and reinitializing integrators after the event, which both are computationally expensive. A new class of ordinary differential equation solvers called Quantized State System (QSS) integration[13] are promising for the efficient simulation of such systems as they do not require iteration for state event detection. However, their efficient use requires knowledge of the dependency graph of the state equations, which generally is not available in legacy building simulators, but readily available in equation-based languages.

[13] Zeigler and Lee, 1998; Kofman and Junco, 2001; Cellier and Kofman, 2006; Kofman, 2003; and Migoni et al., 2013

It follows from this discussion that for models to be applicable to a wide range of applications, it should be possible to use them with different numerical solvers. Therefore, models for building energy systems and their numerical solution methods should be separated where possible. Exceptions are equations, often arising

from partial differential equations or from light distributions, for which special tailored solution methods and parallel programming patterns allow humans to better exploit the structure of the equations than is currently supported by code generators. Examples include solvers for computational fluid dynamics, heat transfer in borehole heat exchangers[14], and ray-tracing for daylighting. Work however is ongoing to remedy this situation[15].

[14] Picard and Helsen, 2014

[15] Casella, 2015; Schuchart et al., 2015; and Bergero et al., 2015

## 3 *New Technologies for Building Energy Modeling and Simulation*

This section describe new technologies which can be applied in building energy modeling in support of the different use cases.

### *3.1 Equation-based Modeling*

As explained above, the use of imperative programming languages limits the applicability and extensibility of models. Furthermore, in building simulation programs, numerical solution algorithms are often tightly integrated into the models and thereby can mandate the use of supervisory control logic that is far removed from how control sequences are implemented in reality. For example, in EnergyPlus, a cooling coil may request from the supervisory control a certain air mass flow rate in order to meet the load computed in the predictor step of the thermal zone heat balance. In the buildings community, the use of equation-based languages has its origin in the energy simulation program ENET[16], which provided the foundation of the SPANK or SPARK program[17]. In 1989, Sahlin and Sowell[18] introduced an equation-based language called Neutral Model Format (NMF) which is used in the commercial software IDA/ICE[19]. In 1993, Klein introduced the equation-based Engineering Equation Solver EES[20]. In 1997, Mattsson and Elmqvist reported on an international effort to design Modelica, an equation-based, object-oriented modeling language[21], which incorporates many ideas originally presented in the dissertation of Elmqvist (1978). A key difference to imperative programming languages is that equation-based languages do not require to specify the sequence of computer assignments needed to simulate a model. Rather, a model developer can specify the mathematical equations, package them into graphically represented components and store them in a hierarchical library. A model user then assembles these components in a schematic editor to form a system model. A simulation environment analyses these equations, optimizes them using computer algebra, translates them to executable code, typically C, and links them with numerical solvers.

[16] Low and Sowell, 1982; and Sowell et al., 1984

[17] Sowell et al., 1986; Sowell et al., 1989; and Buhl et al., 1993

[18] Sahlin and Sowell, 1989

[19] Björsell et al., 1999

[20] Klein, 1993

[21] Mattsson and Elmqvist, 1997

Specifically, the translation for equations to executable code involves analyzing the system of equations to detect for example algebraic loops and zero-crossing functions for state events, and converting the equations to a form that can be solved efficiently using Block Lower Triangularization and Tearing[22]. The benefit of generating code for specific solvers has been demonstrated by Fernandez and Kofman who showed two orders of magnitude simulation speed improvements when code is generated in a form that is specifically designed for the QSS methods[23]. Symbolic manipulations also allow to partition the model automatically for parallel computing[24].

[22] Cellier and Kofman, 2006; and Elmqvist and Otter, 1994

[23] Fernández and Kofman, 2014

[24] Elmqvist et al., 2014

Loosely speaking, while simulation models implemented using imperative programming languages require numerical solvers to select numerical inputs and compare the function values for these inputs to infer what equations they solve, equation-based modeling languages such as Modelica allow to understand the structure of the equations and make use of it to generate efficient code for computation. Examples of structures are what variables are connected to each other through algebraic constraints or through a differential equation, what equations can be differentiated, what equations can be inverted, and what equations trigger an event that can instantly change a control signal. As a detailed discussion of these methods are beyond the scope of this paper, we refer for more background to Elmqvist (1978), Cellier and Kofman (2006), Elmqvist and Otter (1994) and Elmqvist et al. (1995). To make these technologies accessible to a wide range of users in building simulation, research and development is required and ongoing to advance translators and solvers so they can better handle large models[25].

[25] Wetter, 2009; Zimmer, 2013; Wetter et al., 2014; Jorissen et al., 2015; Casella, 2015; Schuchart et al., 2015; and Bergero et al., 2015

A promising aspect of Modelica is that it is an open-source language that is supported internationally by various industries. In the buildings industry, in 2012, an international project was started by Lawrence Berkeley National Laboratory and RWTH Aachen under the umbrella of the International Energy Agency's Energy in Buildings and Communities Programme (IEA EBC) called Annex 60[26]. Annex 60 develops free open-source, new generation computational tools for building and community energy systems based on the Modelica, Functional Mockup Interface and Building Information Modeling standards. It is currently comprised of 41 institutes from 16 countries.

[26] Wetter and van Treeck, 2012

### *3.2 Optimization*

Equation-based modeling languages allow code generators to convert model equations to a form that is well suited to solve large scale nonlinear optimization problems[27]. This section describes a state of

[27] Åkesson et al., 2010

the art method that converts an infinite dimensional optimal control problem into a finite dimensional approximation that standard nonlinear programming (NLP) solvers can solve. Equation-based modeling languages allow automating this conversion.

Equation-based modeling languages allow to describe systems of differential algebraic equations (DAE) in the general form

$$F(t,\, \dot{x}(t),\, x(t),\, u(t),\, y(t),\, \Theta) = 0, \tag{1a}$$

$$Y(t,\, x(t),\, u(t),\, y(t),\, \Theta) = 0, \tag{1b}$$

$$F_0(\dot{x}(t_0), x(t_0), u(t_0), y(t_0), \Theta) = 0, \tag{1c}$$

where $F(\cdot,\cdot,\cdot,\cdot,\cdot,\cdot)$ describes the time rate of change, $Y(\cdot,\cdot,\cdot,\cdot,\cdot)$ are algebraic constraints, $F_0(\cdot,\cdot,\cdot,\cdot,\cdot,\cdot)$ implicitly defines initial conditions, $t \in [t_0,\, t_f]$ is time for some initial and final time $t_0$ and $t_f$, $x\colon \mathbb{R} \to \mathbb{R}^{n_x}$ is the state vector, $u\colon \mathbb{R} \to \mathbb{R}^{n_u}$ is the control function, $y\colon \mathbb{R} \to \mathbb{R}^{n_y}$ is the vector of algebraic variables, and $\Theta \in \mathbb{R}^p$ is the vector of parameters. Such a DAE system can be used to model a building, its HVAC systems and controllers. Necessary and sufficient conditions for existence, uniqueness and differentiability of a solution to (1) can be found in Wetter (2005).

Once the model is available, we can add constraints and a cost function to define an optimal control problem that minimizes energy consumption or cost. An example optimal control problem for (1) is

$$\underset{u(\cdot)\in\mathcal{U},\,\Theta\in\mathbb{R}^p}{\text{minimize}} \quad f(x(t),\, u(t),\, y(t),\, \Theta), \tag{2a}$$

$$\text{subject to} \quad F(t,\, \dot{x}(t),\, x(t),\, u(t),\, y(t),\, \Theta) = 0, \tag{2b}$$

$$Y(t,\, x(t),\, u(t),\, y(t),\, \Theta) = 0, \tag{2c}$$

$$F_0(\dot{x}(t_0),\, x(t_0),\, u(t_0),\, y(t_0),\, \Theta) = 0, \tag{2d}$$

$$H(t,\, \dot{x}(t),\, x(t),\, u(t),\, y(t),\, \Theta) = 0, \tag{2e}$$

$$G(t,\, \dot{x}(t),\, x(t),\, u(t),\, y(t),\, \Theta) \leq 0, \tag{2f}$$

for all $t \in [t_0,\, t_f]$, where $f(\cdot,\cdot,\cdot,\cdot)$ is the cost function and $\mathcal{U}$ is the set of admissible control functions. The solution to (2) is the optimal control function and the optimal design parameter that minimizes $f(\cdot,\cdot,\cdot,\cdot)$ while satisfying the system dynamics (2b) and (2c), the initial conditions (2d) and the constraints (2e) and (2f). For generality, we assume (2a)–(2f) to be nonlinear and twice continuously differentiable[28].

[28] Polak, 1997

The problem (2) is infinite dimensional because its solution is a functional that has to be valid for all $t \in [t_0,\, t_f]$. Directly solving an infinite dimensional optimal control problem for a general nonlinear system is not possible and it therefore needs to be converted into a finite dimensional approximation[29]. Biegler (2010) presents multiple

[29] Polak, 1997

methods for such a conversion into the form

$$\begin{aligned} \underset{z \in \mathbb{R}^{n_z}}{\text{minimize}} \quad & c(z), \\ \text{subject to} \quad & z^l \leq z \leq z^u, \\ & g(z) = 0, \\ & h(z) \leq 0, \end{aligned} \tag{3}$$

where $z$ is the finite dimensional optimization variable, $z^l$ and $z^u$ are the lower and upper bounds, $c(\cdot)$ is the cost function, and $g(\cdot)$ and $h(\cdot)$ are the equality and inequality constraints.

Among the available techniques, we describe direct collocation methods because they are well suited for equation-based modeling languages[30]. Direct collocation methods use polynomials to approximate the trajectories of the variables of a DAE system. The polynomials are defined on a finite number of support points that are called collocation points. Hence, they convert the infinite to a finite dimensional optimization problem, which can be solved by a NLP solver such as IPOPT[31].

[30] Åkesson et al., 2010

[31] Wächter and Biegler, 2006

The method starts by dividing the time horizon $[t_0, t_f]$ into $n_e$ elements, each element containing the same number of collocation points $n_c$. The Modelica tool JModelica[32], which we used, uses the Radau collocation method to place these points, but other methods exist as well. The Radau collocation method places a collocation point at the start and end of each element to ensure continuity of the state trajectories, and places the others to maximize accuracy. In each element, time is normalized as $\tilde{t}_i(\tau) = t_{i-1} + h_i\,(t_f - t_0)\,\tau$, for $\tau \in [0, 1]$ and $i \in \{1, \ldots, n_e\}$, where $t_i$ is the time at the end of element $i$, $\tau \in [0, 1]$ is the normalized time within the element, and $h_i$ is the length of element $i$. The time dependent variables $\dot{x}(\cdot)$, $x(\cdot)$, $u(\cdot)$, and $y(\cdot)$ are approximated using collocation polynomials in each element. The collocation polynomials use the Lagrange basis polynomials, and they use the collocation points as the interpolation points. The collocation polynomials are

[32] Åkesson et al., 2009

$$x_i(\tau) = \sum_{k=0}^{n_c} x_{i,k}\,\tilde{l}_k(\tau), \tag{4a}$$

$$u_i(\tau) = \sum_{k=1}^{n_c} u_{i,k}\,l_k(\tau), \tag{4b}$$

$$y_i(\tau) = \sum_{k=1}^{n_c} y_{i,k}\,l_k(\tau), \tag{4c}$$

where $x_{i,k}$, $u_{i,k}$, and $y_{i,k}$ are the values of the variable $x(\cdot)$, $u(\cdot)$ and $y(\cdot)$ at the collocation point $k$ in element $i$, $l_k(\cdot)$ is the Lagrange basis polynomial and $\tilde{l}_k(\cdot)$ is the Lagrange basis polynomial that includes

the first point to ensure continuity of the state variables. The Lagrange bases are, with $i \in \{1, \ldots, n_e\}$,

$$\tilde{l}_k(\tau) = \prod_{j \in \{0, \ldots, n_c\} \setminus \{k\}} \frac{\tau - \tau_j}{\tau_k - \tau_j}, \tag{5a}$$

$$l_k(\tau) = \prod_{j \in \{1, \ldots, n_c\} \setminus \{k\}} \frac{\tau - \tau_j}{\tau_k - \tau_j}. \tag{5b}$$

As $\tau$ is normalized, the basis polynomials are the same for all elements. The polynomial approximation of the derivative $\dot{x}_i(\cdot)$ of (4a) is

$$\dot{x}_i(\tau) = \frac{1}{h_i \, (t_f - t_0)} \sum_{k=0}^{n_c} x_{i,k} \frac{d\tilde{l}_k(\tau)}{d\tau}. \tag{6}$$

The collocation method defines the approximations (4) and (6) of the variables in (2). Equation-based modeling languages allow accessing the model equations, thereby allowing to automatically generate the finite dimensional approximations defined by the collocation methods in (4) and (6).

JModelica employs a collocation method to transcribe the problem (2) into an NLP problem. A local optimum to the finite dimensional approximation of (2) will be found by solving the first-order Karush-Kuhn-Tucker (KKT) conditions, using iterative techniques based on Newton's method. This requires first- and second-order derivatives of the cost and constraint functions with respect to the NLP variables. JModelica uses CasADi[33], a software for automatic differentiation that is tailored for dynamic optimization. Equation-based modeling languages allow to automatically provide the information required by CasADi to build a symbolic representation of the optimization problem. Using the symbolic representation of the NLP problem, CasADi can efficiently compute the required derivatives and exploit the sparsity pattern of the problem. NLP solvers such as IPOPT are then used to find a piecewise polynomial approximation of the solution to the original problem (2). The number of variables in the approximated problem is $n_z = (1 + n_e \, n_c)(2n_x + n_u + n_y) + (n_e - 1)n_x + n_p + 2$. For a more detailed overview see Magnusson and Åkesson (2012).

[33] Andersson, 2013

In summary, equation-based modeling languages provide three main advantages for optimization: First, they support the automatic conversion of simulation models into optimization problems, reducing engineering costs and time. Second, they can provide analytic expressions for gradients to be used by NLP solvers. Third, they allow to automatically generate the finite dimensional approximations defined by the collocation methods. In Section 4.2 we present how this improves computing performance relative to simulation-based optimization.

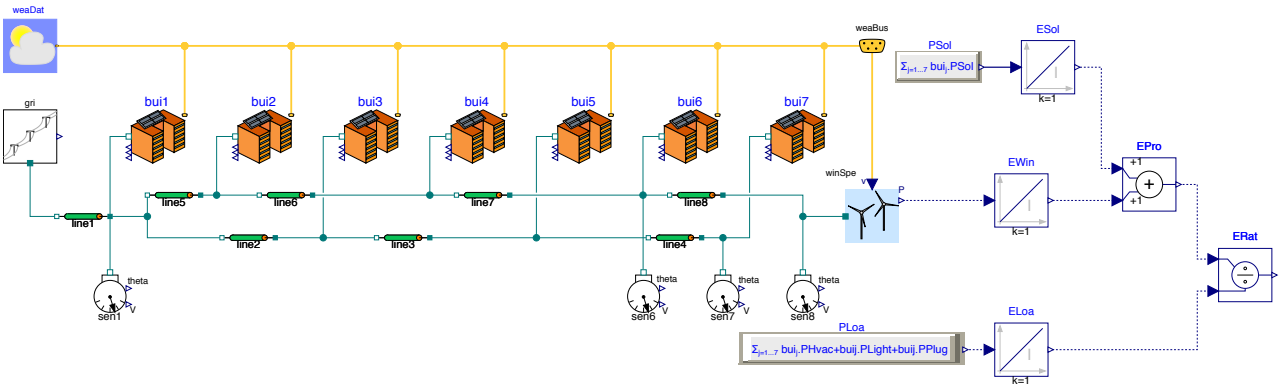


Figure 1: Neighborhood model with renewable energy sources. The yellow lines are weather data, green lines are electrical lines, and dashed blue lines are for post-processing.

## 4 *Applications*

We will now present two applications. First, we demonstrate the development of a building HVAC and PV inverter controller that maintains voltage constraints in the electrical distribution. Second, we demonstrate the reduction in computing time for solving a non-linear optimal control problem with and without the use of the above described computer algebra.

### 4.1 *Combined building temperature and PV inverter control to satisfy voltage constraints in electrical distribution grid*

This example demonstrates how equation-based modeling languages allow to couple models of different physical domains. The example analyzes both the thermal and electrical dynamics of a small neighborhood with a high PV penetration. It requires the coupling of models for building energy simulation, electrical system simulation and feedback control loops. The models interact with each other through the electrical load imposed by the building on the grid, and the feedback control that adjusts the building room temperature set point and that increases the reactive power of the PV inverter in case of violation of the power quality.

Figure 1 shows a net zero energy (NZE) neighborhood, implemented using components from the Buildings library[34], an open-source free Modelica library with more than 500 models for the simulation of buildings, HVAC components, room and interzonal airflow, electrical systems and controls. By coupling models of buildings, HVAC, electrical systems and controls we can assess the effect of building load onto the electrical grid and assess the efficacy of control measures. The neighborhood contains seven small office buildings. Each building represents a small office that is part of the

[34] Wetter et al., 2014

EnergyPlus commercial reference building models[35]. Each model has one floor and is divided into five thermal zones plus one attic. The floor area of each building is about $500\,\mathrm{m}^2$.

[35] Deru et al., 2011

The building model comprises four different components, the thermal part, the schedules, the HVAC system and the electrical models. The thermal part accounts for the heat transfer through the envelope and energy storage in building constructions. The schedules represent the building internal loads due to occupants, as well as plug loads and lighting systems. The HVAC model represents the mechanical system and the control loops that maintain the thermal comfort in the building. The electrical models represent the interaction between the building and the electric network. They include an inductive load model that represents the electric load of the building and PV panels. We implemented all models in Modelica[36]. The thermal model was available as an EnergyPlus model. To integrate it with the rest of the building model, it was automatically converted to Modelica. The automatic conversion program leverages the OpenStudio API to identify the thermal zones and the components of the building fabrics. The conversion program converts the models by instantiating and connecting components of the Modelica Buildings library.

[36] Wetter et al., 2014; and Bonvini et al., 2014

The building electricity consumption is modeled using the inductive load

$$P_{bui}(t) = P_{hvac}(t) + P_{plug}(t) + P_{light}(t), \tag{7a}$$

$$Q_{bui}(t) = P_{bui}(t)\tan(\phi), \tag{7b}$$

where $P_{bui}(\cdot)$ is the total active power, $Q_{bui}(\cdot)$ is the total reactive power, $\phi$ is the phase angle of the apparent power for the power factor $p_f$, with $\phi = \arccos(p_f)$, $P_{hvac}(\cdot)$ is the power consumed by the HVAC system, $P_{plug}(\cdot)$ are the plug loads and $P_{light}(\cdot)$ is the lighting power. In addition, the neighborhood has a wind turbine that supplements the energy provided by the PVs. TMY3 weather data for San Francisco, CA, were used. For more information about the electrical models and their implementations see Bonvini et al. (2014). The nominal voltage of the neighborhood is $V_{nom} = 1.2\,kV$ and the nominal load of each building is $P_{nom}^{load} = 18.6\,kW$. The total nominal power of the PVs installed in the neighborhood is $P_{nom}^{PV} = 130\,kW$ and hence twice the sum of the nominal load of the office units. The PVs are unevenly distributed among the different buildings. The nominal power of the wind turbine is $P_{nom}^{wind} = 93\,kW$ and therefore it is five times the nominal load of an office building. The buildings are connected through annealed aluminum cables of size AWG 1/0 that are 300 m long. The neighborhood produces annually about 25% more energy than it consumes.

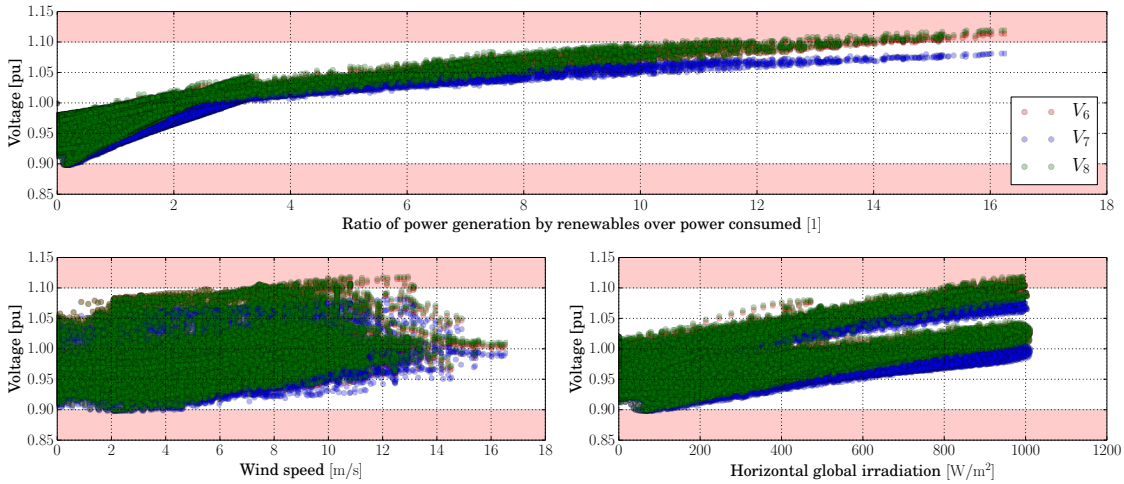


Figure 2: Voltage levels in three different nodes of the neighborhood as a function of power generated by renewables, wind speed and horizontal global irradiation without feedback control.

We assumed the electrical system to be balanced because the analysis does not focus on possible asymmetries caused by the connection of the loads and sources on different phases, but rather on their impact on the voltage quality. In particular, we aim to keep the voltage within an admissible region of $V = [0.9, 1.1]$ pu. To obtain diversity, the PV efficiency, orientation and tilt angles have been varied. Also, the power factor of the inductive load varies among the buildings between 0.8 and 0.95.

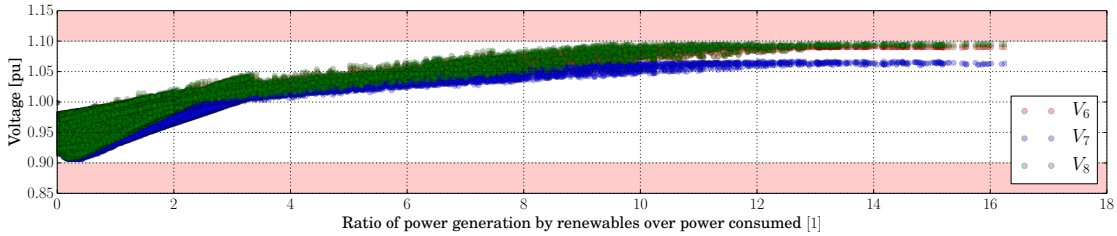


Figure 3: Voltage levels in three different nodes of the neighborhood as a function of power generated by renewables with feedback control that adjust room air temperature setback and reactive power control at all PV inverters.

We will now analyze the simulation results for the case where there is no control to ensure that the voltage remains within the admissible region. Figure 2 shows how the voltages in three different nodes of the neighborhood vary with respect to the power generated by the renewables over the power consumed. The lower two plots show the voltage at the three nodes as a function of wind speed and horizontal global irradiation. As expected, the voltage increases as the power generated by the renewables increases. The highest voltage is $V_8$, measured by the sensor `sen8` in Figure 1, which is at the end of the line where the concentration of PVs is higher and where the wind turbine is located. During 17 hours of the year, the voltage is above or below the admissible region of $V = [0.9, 1.1]$ pu.

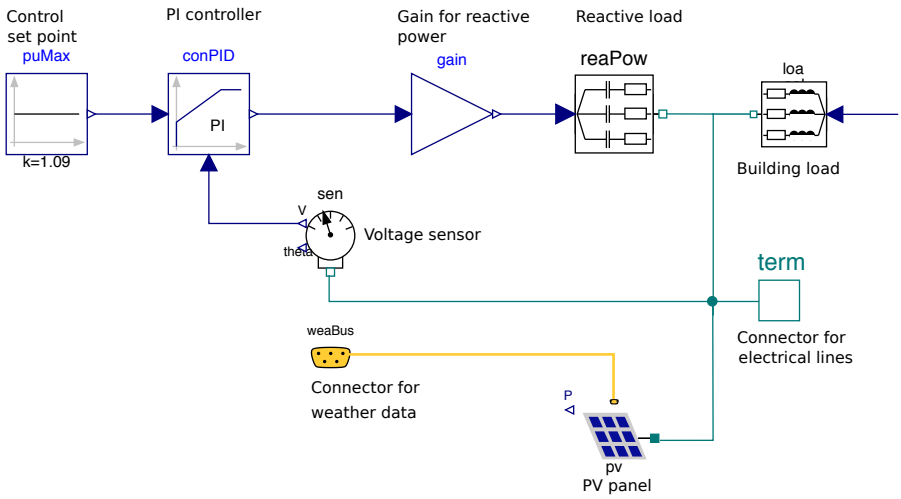


Figure 4: Electrical circuit with reactive power control at the PV inverter.

To keep voltages within the admissible region, we will now add two control measures to the above example. During low voltages, we increase the set point temperature of the buildings by 2 Kelvin. During high voltages, we add reactive power at the PV converters[37]. Figure 4 shows the section of the Modelica model that comprises of the building load, the PV, and a reactive load. Based on the voltage at the PV connection, a feedback controller injects reactive load in order to not exceed a voltage set point of 1.09 pu. Figure 3 shows that this control measure keeps the voltage within the admissible region.

[37] Turitsyn et al., 2011

### 4.2 *Optimal control of a thermal zone*

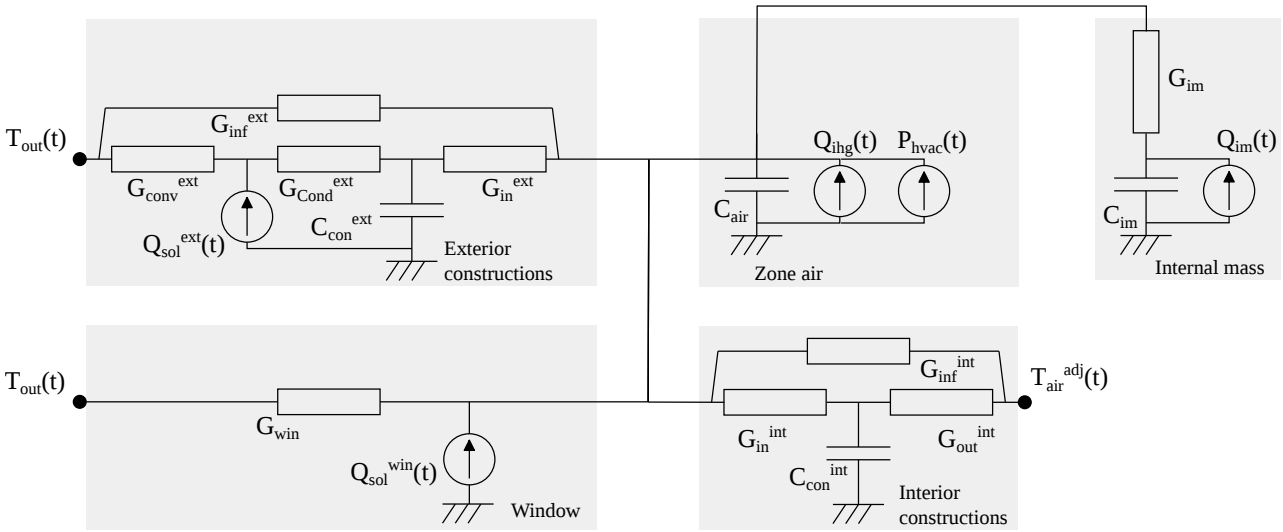


Figure 5: Simplified model of a thermal zone.

This example demonstrates the use of collocation methods to solve a constrained nonlinear optimal control problem, and compares its computing performance to a gradient free optimization method. The example minimizes sensible cooling, heating and fan energy demand for a thermal zone of a variable air volume flow (VAV) system by adjusting the time profiles of the supply air mass flow rate, the shading device control signal and the reheat power at the terminal box, subject to comfort constraints. Our application requirements are that the optimizations of indidual zones are decoupled from each other, and that the central HVAC system can have its own control. For our example, the central HVAC system will supply air at 18°C and at a relative humidity required for humidity control. Figure 5 shows the simplified thermal model of the zone. Separate models exist for exterior constructions, partitions to adjacent zones and internal mass. These constructions are characterized by thermal capacitors and thermal conductances that account for heat conduction and convection. There is also a thermal conductor for infiltration.

We modeled the windows with a thermal conductance $G_{win}$ and a power source $Q_{sol}^{win}(t) = S_{rad}(t)\, A_{win}\, SHGC\, \mathcal{F}(u_{win}(t))$, where $S_{rad}(t)$ is the total solar radiation per unit area towards the window normal direction, $A_{win}$ is the glass area, $SHGC$ is the solar heat gain coefficient of the window, $u_{win}(t) \in [0, 1]$ is the position of the blind, and $\mathcal{F}\colon \mathbb{R} \to \mathbb{R}$ is a function that models the impact of the blinds on the fraction of solar radiation that enters the zone. The contribution due to solar radiation on the external constructions is

$Q^{ext}_{sol}(t) = S_{rad}(t)\, A_{ext}\, \alpha$, where $A_{ext}$ is its area, and $\alpha$ is the solar absorption coefficient of the exterior surface. The thermal conductance $G_{im}$ models the heat transfer between the air and the internal mass. The power sources $Q_{ihg}(t)$ and $Q_{im}(t)$, respectively, model the internal heat gains that contribute to the zone air and internal mass. The internal heat gain is defined as $Q_{ihg}(t) = P_{occ}(t) + P_{lights}(t) + P_{plug}(t)$, where $P_{occ}(t)$ is the internal heat gain due to occupancy, $P_{lights}(t)$ is the internal heat gains due to lights, and $P_{plug}(t)$ is the internal heat gains due to the plug loads. The power delivered by the HVAC system is $P_{hvac}(t) = P_{hea}(t) + P_{sup}(t) + P_{fan}(t)$, where $P_{hea}(t)$ is the power released by the reheating coil in the VAV box, $P_{fan}(t)$ is the fan power, and $P_{sup}(t)$ is the cooling power provided by the supply air from the central HVAC system through the VAV box.

The latter is $P_{sup}(t) = \dot{m}_{air}(t)\, c_p\, (T_{sup}(t) - T_{air}(t))$, where $\dot{m}_{air}(t)$ is the mass flow rate of air passing through the VAV box, $c_p$ is the air specific heat capacity, $T_{sup}(t)$ is the temperature of the supply air entering the VAV box and $T_{air}(t)$ is the temperature of the air in the zone.[38] The power of the fan is $P_{fan}(t) = P^{nom}_{fan}(\dot{m}_{air}(t)/\dot{m}^{nom}_{air})^3$, where $\dot{m}^{nom}_{air}$ is the nominal supply air mass flow rate, and $P^{nom}_{fan}$ is the fan power required to supply $\dot{m}^{nom}_{air}$ to the zone. Weather data for Sacramento, CA, have been used.

We implemented the model using Modelica. The model can be described as an initial-value ordinary differential equation. Thus, it is a special case of the generalized DAE system (1) in which the algebraic constraints $Y(\cdot,\cdot,\cdot,\cdot,\cdot)$ are absent. Therefore, the state variables, control functions and parameter are

$$x(t) = [T_{air}(t),\, T_{im}(t),\, T^{int}_{con}(t),\, T^{ext}_{con}(t)], \tag{8a}$$

$$u(t) = [\dot{m}_{air}(t),\, u_{win}(t),\, P_{hea}(t)], \tag{8b}$$

$$\Theta = [T_{air}(t_0)]. \tag{8c}$$

The energy consumption of the HVAC system is

$$E(t_f) = \int_{t_0}^{t_f} (-P_{sup}(t)\, \eta_{coo} + P_{hea}(t)\, \eta_{hea} + P_{fan}(t))\, dt, \tag{9}$$

where $\eta_{coo}$ and $\eta_{hea}$ are coefficients that repesents the efficiency of the HVAC system to provide cooling and heating. These typically include the coefficient of performance of chillers or heat pumps, or the efficiency of the furnace, as well as the effect of free-cooling by the economizer. The optimization variables are the time profiles for the control signals of the supply air mass flow rate $\dot{m}_{air}(\cdot)$, the blind position $u_{win}(\cdot)$, the reheat power $P_{hea}(\cdot)$, as well as the initial zone temperature $T_{air}(t_0)$. We modified equation (9) to balance the weights of all its terms, obtaining

$$E_\gamma(t_f) = \int_{t_0}^{t_f} (-\gamma^1 P_{sup}(t)\eta_{coo} + \gamma^2 P_{hea}(t)\eta_{hea} + \gamma^3 P_{fan}(t))\, dt, \tag{10}$$

[38] Note that the computation of $P_{sup}(t)$ is approximate to decouple the individual optimizations of multiple zones from each other. This was done to reduce the dimensionality of the optimization and to allow them to be solved in a distributed way. The actual sensible cooling provided by a cooling coil in a system with economizer is $\bar{P}_c(t) = \dot{m}_{air}(t)\, c_p\, (T_{sup}(t) - (T_{mix}(t) + \Delta T_{fan}(t)))$, where $T_{mix}(t)$ is the mixed air temperature after the economizer and $\Delta T_{fan}(t)$ is the temperature raise over the fan. If $y_{out}(t) \in [0,1]$ is the outside air fraction, this becomes $\bar{P}_c(t) = \dot{m}_{air}(t)\, c_p\, (T_{sup}(t) - T_{air}(t) + y_{out}(t)\, (T_{air}(t) - T_{out}(t)) - \Delta T_{fan}(t))$. As $y_{out}(t)$ and $\Delta T_{fan}(t)$ (through the fan efficiency) depends on the mass flow rates and return air temperatures of other zones, these terms have been neglected in order to decouple the individual zone-level optimizations.

where $\gamma \in \mathbb{R}^3$, with $\gamma^i > 0$ for all $i \in \{1, 2, 3\}$, are constants.

We formulated the optimal control problem as

$$\begin{aligned}
\underset{u(\cdot)\in\mathcal{U}}{\text{minimize}} \quad & E_\gamma(t_f) + \int_{t_0}^{t_f} \kappa\, u_{win}(t)^2\, dt, && \text{(11a)}\\
\text{subject to} \quad & F(t, \dot{x}(t), x(t), u(t), y(t), \Theta) = 0, && \text{(11b)}\\
& F_0(\dot{x}(t_0), x(t_0), u(t_0), y(t_0), \Theta) = 0, && \text{(11c)}\\
& T^l_{air}(t) \le T_{air}(t) \le T^u_{air}(t), && \text{(11d)}\\
& 0 \le u_{win}(t) \le u^u_{win}(t), && \text{(11e)}\\
& \dot{m}^l_{air} \le \dot{m}_{air}(t) \le \dot{m}^u_{air}, && \text{(11f)}\\
& P_{hea}(t) \ge 0, && \text{(11g)}\\
& T_{air}(t_0) = T_{air}(t_f), && \text{(11h)}
\end{aligned}$$

for all $t \in [t_0, t_f]$, where $\mathcal{U}$ is the set of admissible control functions $u(\cdot)$ defined in (8b), $\kappa \in \mathbb{R}^+$ is a constant, $T^l_{air}(t)$ and $T^u_{air}(t)$ are the lower and upper bounds of the comfort region of the air temperature, $u^u_{win}(t)$ is the maximum admissible value for the control signal of the blind, and $\dot{m}^l_{air}$ and $\dot{m}^u_{air}$ are the minimum and maximum supply air mass flow rates for the zone. The cost function (11a) includes an additional term that penalizes excessive control actions of the windows. For example it penalizes deploying the blinds at night. The equality constraint (11h) imposes that the temperature of the air at the start and at the end of the optimization period are the same.

We solved the optimization problem using JModelica[39], using the collocation method described in Section 3.2 and solved the finite dimensional NLP problem with IPOPT version 3.11.9, and with the linear solver mumps. For the collocation, we used $n_e = 48$ elements, each being 30 minutes long. For each element, three collocation points were used. The optimization is run for a summer day. The red dotted line in Figure 6(a) shows the outside air dry bulb temperature.

[39] Åkesson et al., 2009

Figure 6 shows the results of the collocation method. The optimal cooling power delivered by the HVAC system maintains the temperature of the zone inside the thermal comfort zone, The constraint (11f) on the minimum air change causes the temperature to not quite reach the upper comfort bound at night. The optimal blind position follows the maximum admissible value in order to reduce the solar heat gain during the afternoon, when the supply mass flow rate reaches its maximum capacity of seven air changes per hour. Figure 6(c) shows that the additional term in (11a) that penalizes excessive control actions causes the blinds to be closed at night. The red line in Figure 6(b) shows that the VAV box does not reheat the air to maintain comfort in the zone. This result is consistent with the high outside air temperatures.

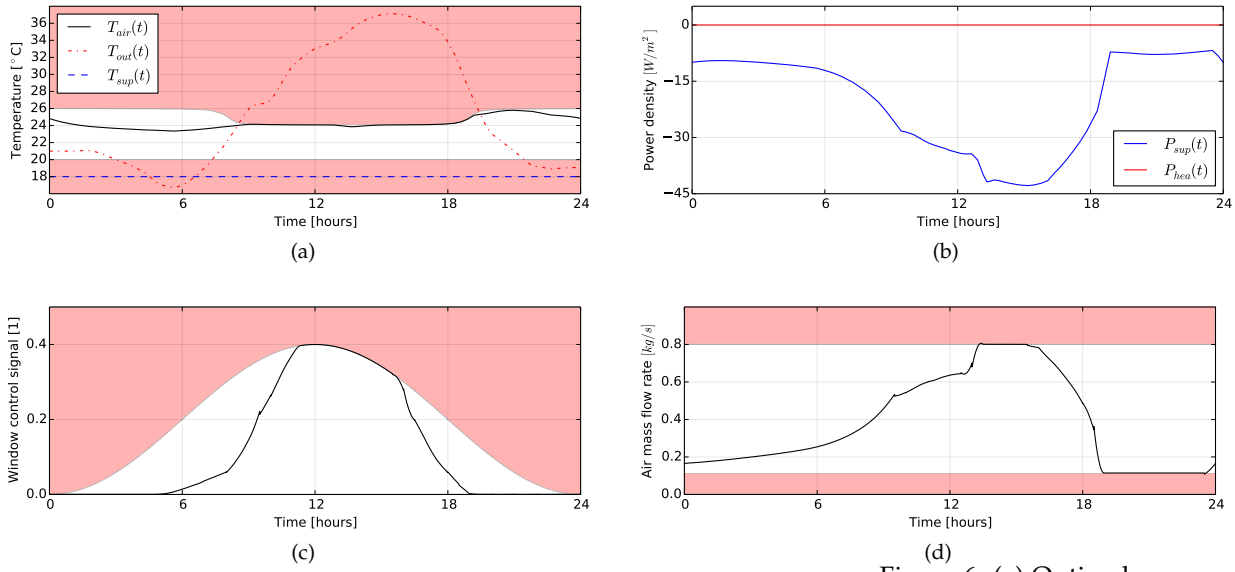


Figure 6: (a) Optimal room air temperature (black line), thermal discomfort zone (red area), outside temperature (red dotted) and supply air temperature (blue dashed). (b) Optimal cooling and heating power provided by the HVAC system through the VAV box. (c) Optimal control signal for the blind (black line). (d) Optimal supply air mass flow rate. Infeasible area for the control signal (red area).

*Simulation-based optimization* To assess the performances of the collocation-based method we now compare it with a simulation-based optimization approach. For the latter, we used the JModelica implementation of the Nelder-Mead (NM) algorithm. The NM algorithm can be applied to nonlinear optimization problems where derivatives are not available, as is the case with conventional building simulation programs. See Wetter and Wright (2004) for a comparison with other derivative free methods.

The optimization variables in the NM optimization problem are the initial temperature of the air in the zone, the control signal for the blinds, and the supply air mass flow rate. To reduce the size of the optimization problem, we simplified it for the NM optimization as follows: We set the heating power to $P_{hea}(t) = 0$ for all $t \in [t_0, t_f]$, and we discretized both the supply air mass flow rate $\dot{m}_{air}(t)$ and the blind control signal $u_{win}(t)$ using one-hour rather than 30 minutes intervals. The decision to choose no heating is appropriate for this hot day. Therefore, the optimization variables are

$$\begin{aligned} z = [&T_{air}(t_0),\, \dot{m}_{hvac}(t_0), \dot{m}_{hvac}(t_0 + \Delta t), \ldots, \dot{m}_{hvac}(t_0 + 24\,\Delta t), \\ &u_{win}(t_0),\, u_{win}(t_0 + \Delta t), \ldots, u_{win}(t_0 + 24\,\Delta t)], \end{aligned} \tag{12}$$

and the total number of variables to optimize is $n_z = 51$. Let $\tau = \{t_0 + i\,\Delta t\}_{i=0}^{24}$. The cost function minimized by the NM algorithm is

$$f_{NM}(z) = E_\gamma(t_f) + \mu_j\,(\Delta T_{err} + \Delta T_{init})\,, \tag{13a}$$

$$\Delta T_{err} = \max_{t \in \tau}(0,\, T_{air}(t) - T^u_{air}(t),\, T^l_{air}(t) - T_{air}(t)), \tag{13b}$$

$$\Delta T_{init} = (T_{air}(t_0) - T_{air}(t_f))^2, \tag{13c}$$

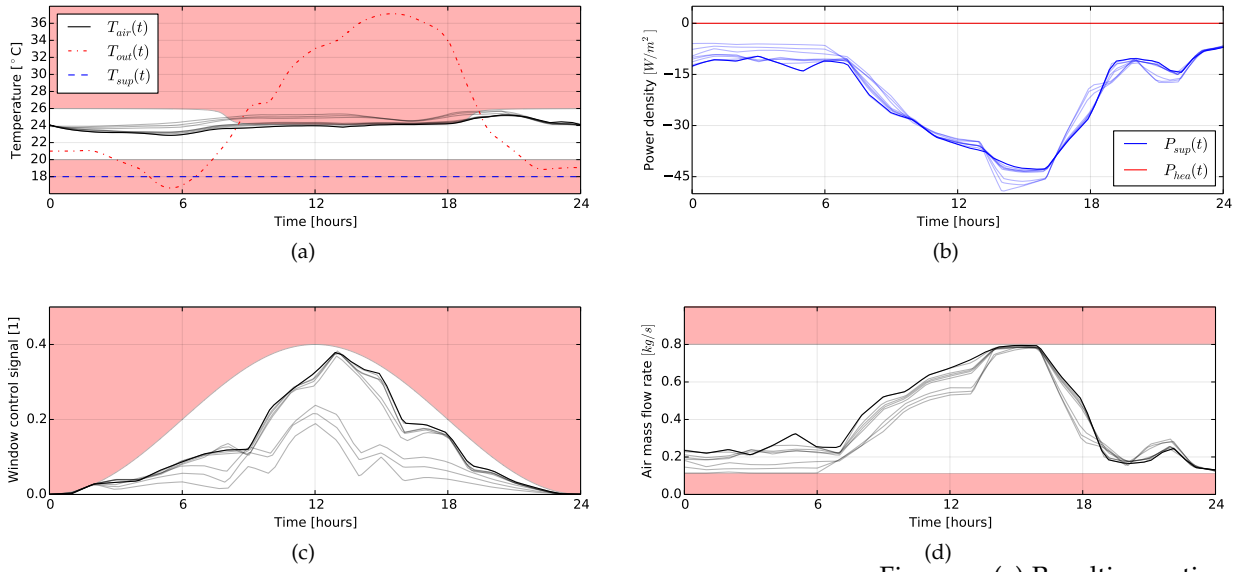


Figure 7: (a) Resulting optimal zone air temperature (black line) and its successive approximations for different values of the penalty function multiplier $\mu_i$ (gray lines). (b) Optimal cooling power provided by the HVAC system through the VAV box (blue line) and its successive approximations for different values of $\mu_i$ (light blue lines), and optimal heating power provided by the VAV box (red line). (c) Optimal control signal for the blind (black line) and its successive approximations for different values of $\mu_i$ (gray lines). (d) Optimal supply air mass flow rate (black line) and its successive approximations for different values of $\mu_i$ (gray lines). The red areas indicate the infeasible regions.

where $\mu_j = 1.5^j/10$, for the iteration counter $j \in \mathbb{N}$, are penalty function multipliers, $\Delta T_{err}$ penalizes thermal comfort constraint violations, and $\Delta T_{init}$ forces initial and final zone air temperature to be equal. The NM algorithm accepts upper and lower limits for the optimization variables. We used the limits $T_{air}(t_0) \in [20, 26]$°C, and $\dot{m}_{air}(t) \in [\dot{m}^l_{air}, \dot{m}^u_{air}]$ and $u_{win}(t) \in [0, u^u_{win}(t)]$ for all $t \in \tau$. These constraints are equivalent to the ones of problem (11).

The blue lines in Figure 7(b) show the optimal cooling load computed during successive iterations of the NM algorithm. Each line corresponds to a different penalty function multiplier $\mu_i$, with the bold line being the final optimal solution. Figure 7(a) shows a similar plot for the resulting optimal zone air temperature. As $\mu_i$ increases, the solutions converge towards an optimal trajectory that satisfy the constraints.

*Comparison* This section compares the results of the two optimization algorithms. Both algorithms were run on Ubuntu 14.04 64 bits hosted on a Virtual Box virtual machine with 2 GB of memory and four processors.

We initialized both algorithms with a sub-optimal feasible solution computed by a PI controller that maintains the temperature of the zone at a fixed value of 24°C. The selection of initial conditions of the optimization problem play an important role, in particular for collocation based methods[40] because the trajectories of the state, input, output and algebraic variables are used to create the initial polynomial approximations and to place the collocation points. Hence, the initial conditions have to be feasible.

[40] Magnusson and Åkesson, 2012

Both algorithms converge to solutions that minimize the energy consumption with a similar strategy. The strategy is to maintain the zone temperature close to the upper boundary of the thermal comfort zone during the peak hours. Both algorithms compute an optimal solution that reaches a maximum cooling power density of approximately $45\,\mathrm{W/m^2}$ around 4 PM. The trajectories computed by the two algorithms are different because of the different discretization mechanisms, implementations of the cost and constraint functions, and the numerical methods. The optimized total energy consumption $E(t_f)$ is 15.721 kWh/day and 16.543 kWh/day for the optimization with the collocation and NM method, respectively. Hence, the collocation methods reduces the cost by an additional 5.2%.

Tables 1 and 2 show the statistics of the two optimization methods. Despite the fact that the optimization problem solved with the collocation method is larger and has a finer temporal resolution, it was solved approximately 2,200 times faster than the problem solved using Nelder-Mead.

| | |
|---|---|
| Number of variables | 10385 |
| Number of equality constraints | 9953 |
| Number of inequality constraints | 1160 |
| Number of Iterations | 72 |
| Initialization time | 0.93 s |
| Solution time | 6.79 s |
| Post-processing time | 0.04 s |
| Total computing time | 7.75 s |

Table 1: Statistics of the collocation-based optimization method.

| $\mu_i$ | Computing time [s] | Function evaluations | Iterations |
|---|---|---|---|
| 0.150 | 2673 | 25033 | 447 |
| 0.225 | 1461 | 13605 | 242 |
| 0.337 | 1877 | 16853 | 300 |
| 0.506 | 2829 | 25033 | 447 |
| 0.759 | 2276 | 20773 | 370 |
| 1.139 | 1768 | 16517 | 294 |
| 1.709 | 1498 | 13829 | 246 |
| 2.563 | 1793 | 15957 | 284 |
| 3.844 | 1222 | 10805 | 192 |
| Total | 17401 | 158405 | 2822 |

Table 2: Statistics of the simulation-based optimization method.

## 5 Conclusions

The use of equation-based languages allows symbolically manipulating the model equations. We listed in Section 2 numerical methods for simulation that benefit from this capability. We also discussed in Section 3 how having access to the model equations aids in automatically discovering mathematical properties that are important if the models are used to simulate stiff systems, hybrid systems or used within a model predictive controller. Finally, we presented two applications. First, we showed how equation-based languages can be used to model a multi-physics problem to assess physical limits and control strategies for building to electrical distribution grid integration. For this purpose, we combined in a schematic graphical editor models of electrical distribution networks, PV and wind turbines, PV inverter with reactive power control, and prototypical buildings to develop a controller that maintains the voltage within the admissible region. Second, we showed how equation-based languages can be combined with optimal control methods that use computer algebra to speed up the solution by a factor of 2,200 compared to using a conventional gradient-free optimization method.

Related work[41] shows other aspects of these technologies and identifies research and development needs in order to advance it to a state where it can become an integral part of building energy simulation.

[41] Wetter, 2009; Zimmer, 2013; Wetter et al., 2014; Jorissen et al., 2015; Casella, 2015; Schuchart et al., 2015; and Bergero et al., 2015

## 6 Acknowledgments

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.

This work emerged from the Annex 60 project, an international project conducted under the umbrella of the International Energy Agency (IEA) within the Energy in Buildings and Communities (EBC) Programme. Annex 60 will develop and demonstrate new generation computational tools for building and community energy systems based on Modelica, Functional Mockup Interface and BIM standards.

The authors thank Juan Van Roy from KU Leuven, Belgium, and Spyros Chatzivasileiadis, Ciaran Roberts and Emma Stewart, all from Lawrence Berkeley National Laboratory, for their feedback on the electrical models.

## 7 Nomenclature

| Symbol | Description |
|---|---|
| $\mathbb{R}^n$ | Euclidean space of $n$-tuples of real numbers |
| $f(\cdot)$ | function, with the dot standing for the undesignated variable |
| $f(x)$ | value of $f(\cdot)$ evaluted for the variable $x$ |
| $f\colon A \to B$ | function with domain in space $A$ and range in space $B$ |
| $x \in A$ | $x$ is an element of space $A$ |
| $\dot{x}(\cdot)$ | time derivative of variable $x$ |

## 8 References

Åkesson, J., K.-E. Årzén, M. Gäfvert, T. Bergdahl, and H. Tummescheit (2010, January). Modeling and optimization with Optimica and JModelica.org—languages and tools for solving large-scale dynamic optimization problem. *Computers and Chemical Engineering 34*(11), 1737–1749.

Åkesson, J., M. Gäfvert, and H. Tummescheit (2009, feb). JModelica – an open source platform for optimization of modelica models. In *Proceedings of MATHMOD 2009 - 6th Vienna International Conference on Mathematical Modelling*, Vienna, Austria. TU Wien.

Andersson, J. (2013). *A general-purpose software framework for dynamic optimization*. Ph. D. thesis, PhD thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10, 3001-Heverlee, Belgium.

Baetens, R., R. De Coninck, J. Van Roy, B. Verbruggen, J. Driesen, L. Helsen, and D. Saelens (2012). Assessing electrical bottlenecks at feeder level for residential net zero-energy buildings by integrated system simulation. *Applied Energy 96*(Special issue on Smart Grids, Renewable Energy Integration, and Climate Change Mitigation - Future Electric Energy Systems), 74–83.

Bergero, F., M. Botta, E. Campostrini, and E. Kofman (2015, September). Efficient compilation of large scale dynamical systems. In P. Fritzson and H. Elmqvist (Eds.), *11-th International Modelica Conference*, Paris, France, pp. 449–458. Modelica Association.

Biegler, L. T. (2010). *Nonlinear programming: concepts, algorithms, and applications to chemical processes*, Volume 10. SIAM.

Björsell, N., A. Bring, L. Eriksson, P. Grozman, M. Lindgren, P. Sahlin, A. Shapovalov, and M. Vuolle (1999, September). IDA

indoor climate and energy. In N. Nakahara, H. Yoshida, M. Udagawa, and J. Hensen (Eds.), *Proc. of the 6-th IBPSA Conference*, Kyoto, Japan, pp. 1035–1042.

Bonvini, M., M. D. Sohn, J. Granderson, M. Wetter, and M. A. Piette (2014). Robust on-line fault detection diagnosis for HVAC components based on nonlinear state estimation techniques. *Applied Energy 124*(0), 156 – 166.

Bonvini, M., M. Wetter, and T. S. Noudui (2014, September). A Modelica package for buildings-to-electrical grid integration. In *Proc. of Fifth German-Austrian IBPSA Conference*, Volume 1, Aachen, Germany, pp. 6–13. BauSIM.

Buhl, W., A. Erdem, F. Winkelmann, and E. Sowell (1993, August). Recent improvements in SPARK: Strong component decomposition, multivalued objects,and graphical interface. In *Proc. of the 3-rd IBPSA Conference*, Adelaide, Australia, pp. 283–291.

Casella, F. (2015, September). Simulation of large-scale models in modelica: State of the art and future perspectives. In P. Fritzson and H. Elmqvist (Eds.), *11-th International Modelica Conference*, Paris, France, pp. 459–468. Modelica Association.

Cellier, F. E. and E. Kofman (2006). *Continuous System Simulation*. Springer.

Clarke, J. (2015). A vision for building performance simulation: a position paper prepared on behalf of the IBPSA Board. *Journal of Building Performance Simulation 8*(2), 39–43.

Crawley, D. B., L. K. Lawrie, F. C. Winkelmann, C. Pedersen, R. Liesen, and D. Fisher (1996, August). Next-generation building energy simulation tools – a vision of the future. In *Proceedings of the ACEEE 1996 Summer Study on Energy Efficiency in Buildings*, Volume 4 of *Commercial Buildings: Technologies, Design, and Performance Analysis*, Asilomar, Pacific Grove, CA, pp. 4.69–4.75. ACEEE.

Deru, M., K. Field, D. Studer, K. Benne, B. Griffith, P. Torcellini, B. Liu, M. Halverson, D. Winiarski, M. Rosenberg, M. Yazdanian, J. Huang, and D. Crawley (2011, February). U.S. Department of Energy commercial reference building models of the national building stock. Technical Report NREL/TP-5500-46861, National Renewables Energy Laboratory, 1617 Cole Boulevard, Golden, Colorado 80401.

Elmqvist, H. (1978). *A structured model language for large continuous systems*. Ph. D. thesis, Lund Institute of Technology, Lund, Sweden.

Elmqvist, H., S. E. Mattsson, and H. Olsson (2014, March). Parallel model execution on many cores. In *10-th International Modelica Conference*, Lund, Sweden, pp. 363–370. Modelica Association.

Elmqvist, H. and M. Otter (1994, June). Methods for tearing systems of equations in object-oriented modelling. In *European Simulation Multiconference*, Barcelona, Spain, pp. 326–332.

Elmqvist, H., M. Otter, and F. Cellier (1995, June). Inline integration: A new mixed symbolic/numeric approach for solving differential–algebraic equation systems. In *Keynote Address, Proc. ESM'95*, Prague, Czech Republic, pp. xxiii–xxxiv". European Simulation Multiconference.

Fernández, J. and E. Kofman (2014). A stand-alone quantized state system solver for continuous system simulation. *SIMULATION 90*(7), 782–799.

Hairer, E. and G. Wanner (1996). *Solving ordinary differential equations. II* (2nd ed.). Springer series in computational mathematics. Berlin: Springer-Verlag.

Jorissen, F., M. Wetter, and L. Helsen (2015, September). Simulation speed analysis and improvements of Modelica models for building energy simulation. In P. Fritzson and H. Elmqvist (Eds.), *11-th International Modelica Conference*, Paris, France, pp. 59–69. Modelica Association.

Klein, S. A. (1993). Development and integration of an equation-solving program for engineering thermodynamics courses. *Computer Applications in Engineering Education 1*(3), 265–275.

Kofman, E. (2003). Quantization-based simulation of differential algebraic equation systems. *SIMULATION 79*(7), 363–376.

Kofman, E. and S. Junco (2001). Quantized-state systems: A DEVS approach for continuous system simulation. *Transactions of The Society for Modeling and Simulation International 18*(1), 2–8.

Low, D. and E. Sowell (1982, September). ENET, a PC-based building energy simulation system. In *Energy Programs Conference*, Austin, Texas, pp. 2–7. IBM Real Estate and Construction Division.

Magnusson, F. and J. Åkesson (2012). Collocation methods for optimization in a modelica environment. In *Proceedings of the 9th International Modelica Conference*, pp. 649–658.

Mattsson, S. E. and H. Elmqvist (1997, April). Modelica – An international effort to design the next generation modeling language. In

L. Boullart, M. Loccufier, and S. E. Mattsson (Eds.), *7th IFAC Symposium on Computer Aided Control Systems Design*, Gent, Belgium, pp. 1–5.

Migoni, G., M. Bortolotto, E. Kofman, and F. E. Cellier (2013). Linearly implicit quantization-based integration methods for stiff ordinary differential equations. *Simulation Modelling Practice and Theory 35*(0), 118 – 136.

Nouidui, T. S. and M. Wetter (2014, September). Linking simulation programs, advanced control and FDD algorithms with a building management system based on the Functional Mock-Up Interface and the building automation Java architecture standards. In *ASHRAE/IBPSA-USA Building Simulation Conference*, Atlanta, GA, pp. 49–55. IBPSA-USA.

Pang, X., M. Wetter, P. Bhattacharya, and P. Haves (2011, August). A framework for simulation-based real-time whole building performance assessment. *Building and Environment 54*(54), 100–108.

Picard, D. and L. Helsen (2014). A new hybrid model for borefield heat exchangers performance evaluation. In *ASHRAE Annual Conference*, Volume 120, Seattle, WA, pp. 1–8. ASHRAE.

Polak, E. (1997). *Optimization, Algorithms and Consistent Approximations*, Volume 124 of *Applied Mathematical Sciences*. Springer Verlag.

Sahlin, P. and E. F. Sowell (1989, June). A neutral format for building simulation models. In *Proceedings of the Second International IBPSA Conference*, Vancouver, BC, Canada, pp. 147–154.

Schuchart, J., V. Waurich, M. Flehmig, M. Walther, W. E. Nagel, and I. Gubsch (2015, September). Exploiting repeated structures and vectorization in modelica. In P. Fritzson and H. Elmqvist (Eds.), *11-th International Modelica Conference*, Paris, France, pp. 265–272. Modelica Association.

Širokỳ, J., F. Oldewurtel, J. Cigler, and S. Prívara (2011). Experimental analysis of model predictive control for an energy efficient building heating system. *Applied Energy 88*(9), 3079–3087.

Sowell, E., K. Taghavi, H. Levy, and D. Low (1984). Generation of building energy system models. *ASHRAE Transactions 90*(1B), 573–586.

Sowell, E. F., W. F. Buhl, A. E. Erdem, and F. C. Winkelmann (1986, September). A prototype object-based system for HVAC simulation. Technical Report LBL-22106, Lawrence Berkeley National Laboratory.

Sowell, E. F., W. F. Buhl, and J.-M. Nataf (1989, June). Object-oriented programming, equation-based submodels, and system reduction in SPANK. In *Proceedings of the Second International IBPSA Conference*, Vancouver, BC, Canada, pp. 141–146.

Turitsyn, K., P. Sulc, S. Backhaus, and M. Chertkov (2011, June). Options for control of reactive power by distributed photovoltaic generators. *Proceedings of the IEEE 99*(6), 1063–1073.

Wächter, A. and L. T. Biegler (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming 106*(1), 25–57.

Wetter, M. (2005, August). BuildOpt – a new building energy simulation program that is built on smooth models. *Building and Environment 40*(8), 1085–1092.

Wetter, M. (2009, June). 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.

Wetter, M. (2011). The future of building system modeling and simulation. In J. L. M. Hensen and R. Lamberts (Eds.), *Building Performance Simulation for Design and Automation*, pp. 481–504. Oxon, OX: Taylor & Francis.

Wetter, M. and C. van Treeck (2012). New generation computational tools for building and community energy systems.

Wetter, M. and J. Wright (2004, August). A comparison of deterministic and probabilistic optimization algorithms for nonsmooth simulation-based optimization. *Building and Environment 39*(8), 989–999.

Wetter, M., W. Zuo, T. S. Nouidui, and X. Pang (2014). Modelica Buildings library. *Journal of Building Performance Simulation 7*(4), 253–270.

Zeigler, B. and J. S. Lee (1998). Theory of quantized systems: Formal basis for DEVS/HLA distributed simulation environment. In *SPIE Conference on Enabling Technology for Simulation Science*, Volume SPIE Vol. 3369, Orlando, pp. 49–58.

Zimmer, D. (2013). Using artificial states in modeling dynamic systems: Turning malpractice into good practice. In *Proceedings of the 5th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, pp. 77–85.