SimBuild 2006

Second National IBPSA-USA Conference
Cambridge, MA
August 2-4, 2006

MODELICA VERSUS TRNSYS – A COMPARISON BETWEEN AN EQUATION-BASED AND A PROCEDURAL MODELING LANGUAGE FOR BUILDING ENERGY SIMULATION

Michael Wetter and Christoph Haugstetter
United Technologies Research Center
411 Silver Lane
East Hartford, CT 06108, USA

ABSTRACT

We present the implementation of a multizone building energy simulation model in the equation-based modeling language Modelica and compare it to an implementation with comparable modeling detail in TRNSYS. Our development time comparison indicates a five to ten times faster development through use of the equation-based modeling language Modelica. The development time for TRNSYS was extrapolated using data from BuildOpt and comparison between BuildOpt and TRNSYS since TRNSYS data were not available. We provide a comparative model validation of the Modelica and TRNSYS implementation, and compare their numerical performance. The difference in predicted annual heating and cooling energy usage between our Modelica model and TRNSYS is 5% to 20%, which we primarily attribute to our simplified window model. The computation time of the Modelica model was three to four times longer than TRNSYS, but we believe that dismissing equation-based modeling languages as computationally inefficient is a premature conclusion.

INTRODUCTION

Equation-based object-oriented modeling languages such as Modelica (Mattson and Elmqvist 1997; Fritzson and Engelson 1998) offer significant benefits with regard to model development time, model reuse and hierarchical model construction to manage the complexity of large systems. The building simulation community, however, uses mainly modeling and simulation software that use a flat representation of the building energy system topology, that do not use object-oriented modeling, and that mix modeling the physics with the implementation of numerical solution algorithms. The implications are difficulties in representing large building energy systems in a structured way, high costs for extending existing code to implement new physical phenomena and limited capabilities for analysis apart from time domain simulations. For example, rather ordinary tasks such as rewinding an integrator for solving an optimal control problem in a model predictive control algorithm or performing an input-output linearization for designing a controller is hard because building simulation programs typically do not allow specifying initial conditions for all state variables, nor do they allow the user to control the error of a numerical approximation to a state variable which is critical for approximating derivatives and for constructing optimization algorithms with provable convergence properties (Polak and Wetter 2006).

In 1996, a consortium formed to develop Modelica, an equation-based object-oriented language for modeling of dynamic systems. The goal of the consortium is to combine the benefits of existing modeling languages and to define a new uniform language for model representation by creating a non-causal modeling language for multi-domain physics (Fritzson and Engelson 1998).

Modelica has been applied for some building energy modeling applications. For example, Merz (2002) and Felgner et al. (2002) describe the Modelica library ATPlus which can be used for thermal building simulation. Hoh et al. (2005) expanded the components of the ATPlus library to include room model and heat exchangers that are embedded in wall constructions. Nytsch-Geusen et al. (2005b) developed a hygrothermal building model as part of a Modelica library for multizone building heat and mass transfer analysis. The work is conducted by a consortium of six Fraunhofer Institutes that develops the MOSILAB generic simulation tool that is based on Modelica (Nytsch-Geusen et al. 2005a).

We are interested in how Modelica compares to the TRNSYS (Klein, Duffie, and Beckman 1976) simulation program which is well established in the building simulation community. Of particular interest is the model development time in an equation-based versus a procedural modeling environment because the model development time often dominates the time that is ultimately spent for conducting numerical experiments. We are also interested in how com-

plex models can be dealt with in an equation-based object-oriented modeling environment and how their numerical performance compares.

In the literature, equation-based and procedural modeling languages have been compared by others. Sahlin (1996) reports that developing a simulation model in the equation-based modeling language NMF (Sahlin and Sowell 1989) was about three times faster than it was in the BRIS simulation program (Brown 1990), but the computation time in BRIS was three times faster. Sahlin et al. (2004) compared the computation time of IDA ICE with EnergyPlus. In their numerical experiments, IDA ICE required approximately half the computation time that was required by EnergyPlus for a three zone building with natural ventilation and double the computation time in initial experiments for a 53 zone building without natural ventilation. In the latter experiments, the COMIS airflow program was not included in EnergyPlus. Sowell and Haves (2001) compared the computation time of SPARK and HVACSIM+ for a variable air volume flow system that serves six thermal zones. They report that SPARK computes about 15 to 20 times faster than HVACSIM+ and attribute the decrease in computation time to SPARK's graph decomposition and cut set reduction.

In this paper, we compare the two modeling and simulation environments TRNSYS and Dymola (Brück et al. 2002), which provides a Modelica implementation. To illustrate the complexity of the models used in our comparison, we discuss the relevant physics and provide a comparative model validation. We discuss the time that was required to implement the model equations in Modelica and we benchmark the computation time of the two model implementations.

BUILDING AND HVAC MODEL USED FOR PROGRAM COMPARISON

We will now discuss the building and HVAC model that we implemented in TRNSYS and in Modelica. In both simulation environments, we implemented the same one-story building for a quick service restaurant. The building is modeled using separate thermal zones for the dining area, the kitchen area, the storage area and the bathrooms.

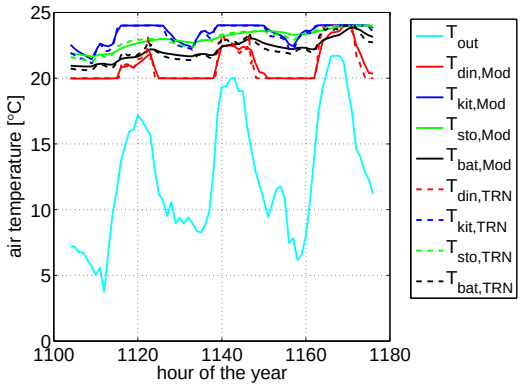
In TRNSYS, we used the multizone building model TYPE56 to model the building. The heating energy consumption and the cooling energy consumption for temperature and humidity control was modeled with the heating and cooling system that is part of TYPE56. The air-side coupling between zones is modeled using a fixed air flow rate and the COUPLING key word of TYPE56's wall model.

In Modelica, we developed a physics-based thermal zone model that is based on Modelica 2.2 (Modelica Association 2005) and Modelica_Fluid (Elmqvist, Tummescheit, and Otter 2003). The thermal zone model can be parameterized to model zones with an arbitrary number of opaque surface constructions and windows. The thermal zone models were connected to each other to construct a multizone building model. Heat conduction in opaque constructions is computed using a finite difference scheme for the spatial discretization of the heat equation. The spatial discretization grid is automatically generated based on material properties and layer thickness, using the same algorithm as is used in BuildOpt and described by Wetter (2004). The air volume of each zone is completely mixed. Long-wave radiative heat transfer between interior surfaces is computed using an area and long-wave emissivity weighted average radiative temperature (Wetter 2004). To reduce the development time required for coding the thermal zone model, our thermal zone model takes as input time-series for the sky temperature, the solar irradiation on the building's outside surfaces and a window's transmitted solar radiation. Those time-series can be precomputed on a simple building model that does not need to have the correct building geometry. In our experiments, we precomputed those time-series in TRNSYS. The window's transmitted solar radiation is used to compute the solar radiation absorbed by the window panes using a simplified optical model that allowed us, at the expense of accuracy, to not having to implement the lengthy algorithm that is based on Windows 4.1 (Finlayson et al. 1993) and used by many building energy simulation programs, including TRNSYS.

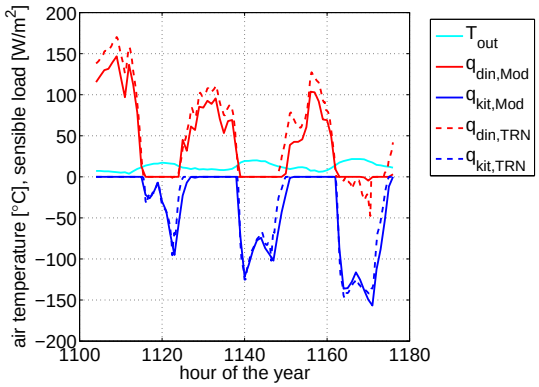
COMPARATIVE MODEL VALIDATION

We will now show results of a comparative model validation for the above described multizone building model. While not as rigorous a validation as the ANSI/ASHRAE Standard 140-2001 (Standard Method of Test for the Evaluation of Building Energy Analysis Computer Programs, see ASHRAE 2001) our comparative model validation shows that the Modelica implementation represent the physical phenomenas with sufficient accuracy to justify a benchmarking with TRNSYS. Measurement data that show the HVAC energy usage or indoor air temperatures did not exist for the analyzed building.

Fig. 1 shows the indoor and outdoor temperatures and the sensible load for a mild day, with good agreement between both models. Comparing the building's annual energy consumption of the two implementations shows that Modelica underpredicts heating by 17%, overpredicts sensible cooling by 1%, and underpredicts latent cooling by 9% and total



(a) Room air temperatures.



(b) Sensible load (heating and cooling). Not shown are the loads for the storage area and bath rooms which are for both zones smaller than $15\,\mathrm{W/m^2}$ in magnitude.

Figure 1: Comparison of room air temperatures and sensible load for a mild day. In the legend, "T" denotes air temperature, "q" denotes sensible load, the subscript "out" denotes outside, "din" denotes dining area, "kit" denotes kitchen, "sto" denotes storage area, "bat" denotes bathroom, "Mod" denotes Modelica and "TRN" denotes TRNSYS.

cooling by 3%. Sensible cooling, however, is overpredicted in the dining area by 7% and underpredicted in the kitchen area by 5%.

COMPARISON OF SIMULATION ENVIRONMENTS

Modelica (Fritzson and Engelson 1998) is an equation-based, non-causal, object oriented modeling language that is designed for component oriented multi-domain modeling of dynamic systems. Models are described by differential equations, algebraic equations and discrete equations. Using standardized interfaces, a model's mathematical relations between its interface variables is encapsulated, and the model can be represented graphically by an icon to facilitate model reuse, model exchange and connecting component models to system models using a graphical editor. Fig. 2 shows how our multizone building model was created by graphically connecting heat and air flow ports of thermal zone models. Each icon contains a model that can consist of other models, thus allowing to create complex models in a hierarchical way. This hierarchical model building facilitates model reuse and testing of submodels before they are assembled to a large system model that may be difficult to debug. To reduce the model development time, the object-oriented model construction in Modelica allows discipline experts, such as an HVAC engineer and a controls engineer, to model their respective process, and latter interface the models, effectively allowing to concurrently as opposed to sequentially build a model. By drawing lines between model ports, mathematical relationships are generated based on the ports' physical quantities.

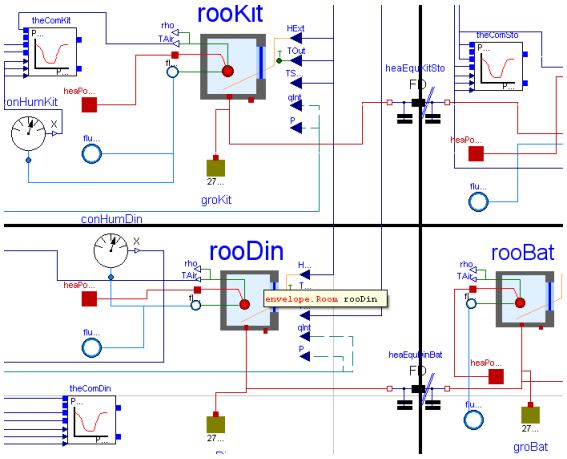


Figure 2: Schematic view of building model in Dymola's graphical editor.

Modelica libraries for multi-domain physics include models for control systems, for thermal systems, for electrical systems and for mechanical systems. Libraries are created using objects that define standard interfaces. For example, in Modelica's thermal library there is a connector called `HeatPort` that defines temperature and heat flow as

```
1 partial connector
2   Modelica.Thermal.
3     HeatTransfer.Interfaces.HeatPort
4     "Thermal port for 1–D heat transfer";
5     SI.Temperature T "Port temperature";
6     flow SI.HeatFlowRate Q_flow
7       "Heat flow rate (positive if
8        flowing into the component)";
9 end HeatPort;
```

On line 6, the type prefix flow declares that all variables connected to `Q_flow` need to sum to zero. For example, if two `HeatPorts` are connected, the relationships $T_1 = T_2$ and the conservation equation

$Q_{flow,1} + Q_{flow,2} = 0$ are generated. This connector can then be used to define the interface for a one-dimensional heat transfer element in the form

```
1  partial model Element1D
2    "Partial heat transfer element with
3     two HeatPort connectors that does
4     not store energy"
5    SI.HeatFlowRate Q_flow
6     "Heat flow rate from port_a->port_b";
7    SI.Temperature dT "port_a.T-port_b.T";
8   public
9    HeatPort port_a;
10   HeatPort port_b;
11  equation
12   dT = port_a.T - port_b.T;
13   port_a.Q_flow = Q_flow;
14   port_b.Q_flow = -Q_flow;
15 end Element1D;
```

This one-dimensional heat transfer element can then be used to define a thermal conductor as

```
1 model ThermalConductor
2   "Lumped thermal element
3    transporting heat without storing it"
4   extends Interfaces.Element1D;
5   parameter SI.ThermalConductance G
6     "Constant thermal conductance";
7 equation
8   Q_flow = G*dT;
9 end ThermalConductor;
```

Note that by replacing the parameter declaration on line 5 and the mathematical model on line 8, the semantics can be changed to represent other one-dimensional heat transfer elements such as a model for long-wave radiation between two surfaces.

Modelica is a language for model representation and cannot be executed directly. To create executable code, it need to be parsed to a programming language that can be compiled. To do so, we used the Dymola 5.3e simulation environment. Dymola performs symbolic manipulations to reduce the dimensionality of the linear and non-linear systems of equations that is defined by the Modelica model. Dymola creates two models, one for computing the initial values, and one for performing the time integration. For the building and HVAC model described in this paper, Tab. 1 shows the dimensionality before and after the symbolic manipulations for those two models. Dymola's implementation of Modelica also performs automatic differentiation. For our problem, it found analytic expressions for all Jacobian matrices, eliminating the need for numerically approximating derivatives. This reduces the computation time and increases the robustness of the numerical solver. Finally, Dymola generates C/C++ code and compiles it. Before running the simulation, a time integrator can be selected from a library with 14 integration algorithms. The library includes implicit integration algorithms with variable step size, which are of particular interest for building energy and controls systems since those systems are often stiff. For controls design, simulating with time steps of less than a minute is often required to properly represent state machines and the rejection of disturbances, and using a variable step size solver allows increasing the time step during times when no mode switching or no fast transients occur.

Because the formulation of the system of equations is separated from its numerical solution algorithm, restarting the integrator, as required if the model is used as a state estimator in a model predictive control algorithm, or linearizing the model, as required for many controls design methods, can easily be performed. Furthermore, the separation between equations and solution procedure also allows automatic design of controllers that are based on nonlinear inverse plant models (see Looye et al. 2005). The steps involved symbolic equation manipulations to obtain an inverse model, automatic differentiation and generation of C code with real-time integration algorithms that has been downloaded to control platform and successfully flight tested on an aircraft. Furthermore, models formulated in Modelica can also be compiled and run from MATLAB/Simulink which allows using Simulink for controls design.

In comparison, TRNSYS uses a procedural language to define its simulation models and discrete time integration algorithm. Models can be encapsulated in so-called TYPEs, which are FORTRAN routines with standardized arguments. Many TYPEs implement their own mathematical solvers for differential and algebraic equations. For example, the multizone building model TYPE56 uses transfer functions by Mitalas for the time integration of the opaque construction's temperature and an iterative solver for the window pane temperatures. The output of TYPEs can be declared as input of TYPEs in a text editor or in a graphical editor such as in the *TRNSYS Simulation Studio* to define a system model. The system model is then solved using either successive substitution with relaxation, or a differential equation solver that converts the system of differential equations to non-linear algebraic equations which it solves using Powell's method, a combination between steepest descent and Newton's method. There is no symbolic manipulations to reduce the size of the system of equations nor is there automatic differentiation performed.

Table 1: Reduction of the number of equations and number of variables to be solved for in Dymola's implementation of Modelica.

		Before manipulation		After manipulation	
		system of equations	variables	system of equations	variables
Initialization problem	linear	1	409	1	77
	nonlinear	6	18	6	6
Differential equation	linear	27	127	1	8
	nonlinear	72	252	72	78

MODEL DEVELOPMENT TIME

To comment on the model development time, we compare the model development time for our Modelica model with the multizone thermal building model BuildOpt (Wetter 2005) because giving a time estimate for the TRNSYS building model is difficult if not impossible since it evolved over many years with contributions of several developers. BuildOpt, however, was developed by one of the authors, has a thermal building model with similar complexity than our Modelica model, and its overhead for data management between the model equations and the program kernel is comparable with the overhead in TRNSYS.

BuildOpt is implemented in C/C++ and took about one year to develop. This estimate does not include the time it took to implement BuildOpt's optical window model, sky temperature model and rather simple daylighting model, which are not implemented in our Modelica model.[1] This compares to one to two months of development time for the Modelica building model by the same author. Thus, using Modelica led to a 5 to 10 times reduction in development time compared to using C/C++. The reduced development time is mainly attributed to the fact that Modelica is an equation-based object-oriented language which eliminates the need for writing routines for data input and output and for managing the large amount of data that is involved in a building simulation. The shorter development time also manifests itself in a four times smaller code size. Specifically, the Modelica model required us to write 6,000 lines of code (0.25 MB), whereas in BuildOpt, 24,000 lines of code (1.0 MB) were required. To make the comparison representative, in Modelica we did in the line count only include the code that we wrote, and not the commercially available libraries Modelica 2.2 and Modelica_Fluid. In BuildOpt, we did not count the daylighting model, the optical window model and the sky temperature model, which account for 3,500 lines of code (0.2 MB), because they are not implemented in our Modelica model. We neither accounted for the size of the commercial solver DASPK. We expect that the comparison would be similar for adding new models in Modelica vs. TRNSYS, since the overhead in each TRNSYS type is comparable to the overhead in each BuildOpt model.

[1] Based on the code size of BuildOpt's implementation of those models, we estimate that implementing those models in Modelica would take about one to two weeks of labor.

NUMERICAL PERFORMANCE

We will now make an attempt to compare the computation time of the TRNSYS and Modelica representation of our multizone building model. As discussed by Sahlin et al. (2004), the creation of a practical and politically acceptable framework for fair comparison of the numerical performance of a simulator with a fixed time step and a variable time step integration algorithm is delicate since the temporal resolutions are very different. In our comparison, because the computation time depends on the solver tolerance and in the case of TRNSYS on the integration step size, we compare the computation times for various settings of those precision parameters. To verify the numerical accuracy of the obtained solutions, we also compare for each experiment the numerical error of the state variables with the state variables computed at the highest tolerance settings. Specifically, for Modelica, let $\varepsilon_m \in \{10^{-k}\}_{k=1}^{9}$ be the solver tolerance and for TRNSYS, let $\varepsilon_t \in \{(10^{-k}, 1/n)\}$, with $k \in \{1,2,\ldots,9\}$ and $n \in \{1,2,\ldots,8\}$ be the vector of solver tolerance and integration step size in hours. We define $\varepsilon_m^* \triangleq 10^{-9}$ and $\varepsilon_t^* \triangleq (10^{-9}, 1/8)$, which corresponds to the highest precision settings used in our experiments.

For a fixed solver tolerance ε, let $u(\varepsilon,\cdot) \in \mathbb{R}^4$ denote the numerical approximation to the state trajectory on the time interval $[0,1]$, which corresponds to one year. The state variable can be for the four thermal zones the air temperature which we denote by $T(\cdot,\cdot)$, the heating energy which we denote by $E_h(\cdot,\cdot)$ or the sensible and latent cooling energy which we denote by $E_c(\cdot,\cdot)$. We will compare for

different solver settings ε the absolute errors

$$e_T(\varepsilon) \triangleq \frac{1}{4}\sum_{i=1}^{4}\int_0^1 |T^i(\varepsilon^*,s) - T^i(\varepsilon,s)|\,ds, \qquad (1)$$

$$e_h(\varepsilon) \triangleq \frac{1}{4}\sum_{i=1}^{4} |E_h^i(\varepsilon^*,1) - E_h^i(\varepsilon,1)|, \qquad (2)$$

$$e_c(\varepsilon) \triangleq \frac{1}{4}\sum_{i=1}^{4} |E_c^i(\varepsilon^*,1) - E_c^i(\varepsilon,1)|, \qquad (3)$$

where ε^* is the highest solver tolerance. The units are Kelvins for $e_T(\cdot)$ and Joules for $e_h(\cdot)$ and $e_c(\cdot)$.

We run the Modelica model from the Dymola simulation environment and from MATLAB/Simulink. In Dymola 5.3e, we used the GCC compiler and the DASSL differential algebraic equation solver with its standard solver parameters. In MATLAB/Simulink 7.0.4.365 (R14) Service Pack 2, we used the GCC compiler and the ode15s solver which is applicable for stiff systems. In TRNSYS 16.00.0038, we used the solver that is based on successive substitution. All numerical experiments were run on a Dell Precision Workstation 670 with a 2.8 GHz dual processor and 4 GB RAM running Windows XP.

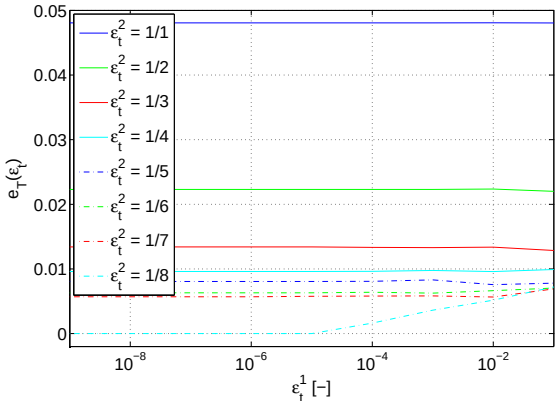


Figure 3: Numerical error $e_T(\cdot)$ as a function of the solver tolerance for the TRNSYS numerical experiments.

Fig. 3 shows that for TRNSYS, the error in room air temperature $e_T(\cdot)$ introduced by changing the number of time steps per hour ε_t^2 from eight to one dominates the error introduced by relaxing the solver tolerance ε_t^1 from 10^{-9} to 10^{-1}. In fact, for $\varepsilon_t^1 \leq 10^{-5}$ and $\varepsilon_t^2 = 1/8$, we obtained $e_T(\varepsilon_t) = e_h(\varepsilon_t) = e_c(\varepsilon_t) = 0$, which explains why no TRNSYS results are shown for these solver tolerance settings in Fig. 4. Also, the TRNSYS computation time is only weakly dependent on the TRNSYS solver tolerance ε_t^1, as is shown in Fig. 5. Based on our experience, however, we expect a stronger dependence of the computation time

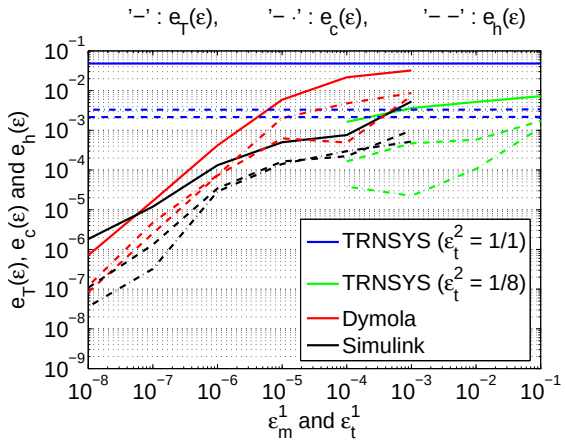


Figure 4: Numerical error as a function of the solver tolerance for TRNSYS, Dymola and Simulink.

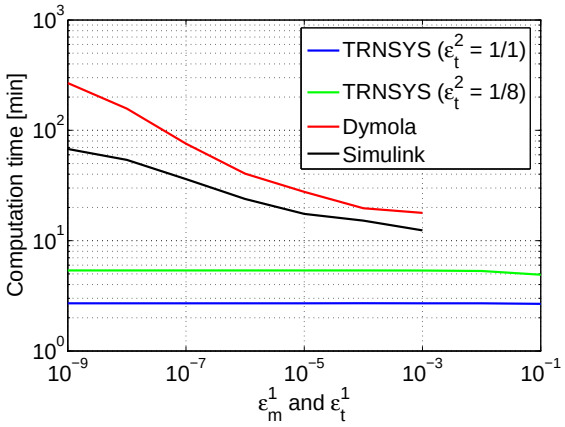


Figure 5: Computation time as a function of the solver tolerance and the number of time step per hour.

on ε_t^1 for models that have a more pronounced nonlinearities than the building model that we used in our experiments.

For Dymola and Simulink, in contrast, we see in Fig. 4 that the numerical error does as expected depend on the solver tolerance because in those simulators, the solver tolerance settings control the integration step size and the number of iterations in the nonlinear equations solvers. For Dymola and Simulink, the solver failed to converge for $\varepsilon_m \in \{10^{-2}, 10^{-1}\}$. For $\varepsilon_m \leq 10^{-3}$, each tenfold increase in solver tolerance increases the computation time on average by a factor of 1.6 for Dymola and 1.35 for Simulink. In Modelica, the computation time is larger than in Simulink, but our experience is that the Dymola solver is more robust.

For TRNSYS, the computation time doubles if

the number of time steps per hours is changed from one to eight. Compared to the TRNSYS simulation with $\varepsilon_t = (10^{-4}, 1/8)$, the Modelica simulation with $\varepsilon_m = 10^{-4}$ is four times slower if run from Dymola and three times slower if run from Simulink. While a time step length of $1/8$ hours seems short for most annual energy analysis, it is not small enough for certain controls analysis.

Conclusively, we can support the statement of Sahlin et al. (2004) that dismissing DAE methods as being inherently inefficient is a premature conclusion.

CONCLUSIONS

We achieved a five- to tenfold reduction in model development time by using the equation-based model representation language Modelica compared to the time that was required to implement a model with similar physics in C/C++. For our situation, this translated in about one year of labor savings. We expect that a similar comparison holds between Modelica and TRNSYS, but we cannot provide data on this comparison because TRNSYS' building model evolved over several years which makes a time estimate difficult if not impossible. Our experience with Modelica is that it is easier in Modelica to construct large models than it is in TRNSYS because Modelica allows models to be built in a hierarchical way that facilitates debugging and reuse of submodels. Furthermore, having a variable step size solver, a symbolic equation manipulator, automatic differentiation and being able to provide initial values for the state variables enables certain controls design and analysis that is not possible with TRNSYS.

We see the main strength of TRNSYS in its large HVAC and building model libraries that passed significant validation tests. However, writing a new model as a TRNSYS TYPE requires significantly more time than it takes to write the same model in Modelica. A further advantage of TRNSYS is its faster computation time, but since we did not explore ways to improve the numerical performance of the Modelica model, and in view of Sahlin et al. (2004) and Sowell and Haves (2001), we do not believe that the longer computation time is an inherent feature of equation-based simulation environments.

ACKNOWLEDGMENTS

This research was supported by the U.S. Department of Commerce, National Institute of Standards and Technology, Advanced Technology Program under the agreement number 70NANB4H3024.

REFERENCES

ASHRAE. 2001. ANSI/ASHRAE Standard 140-2001, Standard Method of Test for the Evaluation of Building Energy Analysis Computer Programs.

Brown, Gösta. 1990. "The BRIS simulation program for thermal design of buildings and their services." *Energy and Buildings* 14 (4): 385–400.

Brück, Dag, Hilding Elmqvist, Sven Erik Mattsson, and Hans Olsson. 2002, March. "Dymola for Multi-Engineering Modeling and Simulation." Edited by Martin Otter, *Proceedings of the 2nd Modelica conference*. Modelica Association and Deutsches Zentrum fur Luft- und Raumfahrt, Oberpfaffenhofen, Germany, 55–1 – 55–8.

Elmqvist, Hilding, Hubertus Tummescheit, and Martin Otter. 2003, November. "Object-Oriented Modeling of Thermo-Fluid Systems." Edited by Peter Fritzson, *Proceedings of the 3rd Modelica conference*. Modelica Association and Institutionen för datavetenskap, Linköpings universitet, Linköping, Sweden, 269–286.

Felgner, F., S. Agustina, R. Cladera Bohigas, R. Merz, and L. Litz. 2002, March. "Simulation of Thermal Building Behaviour in Modelica." Edited by Martin Otter, *Proceedings of the 2nd Modelica conference*. Modelica Association and Deutsches Zentrum fur Luft- und Raumfahrt, Oberpfaffenhofen, Germany, 147–154.

Finlayson, E. U., D. K. Arasteh, C. Huizenga, M. D. Rubin, and M. S. Reilly. 1993, July. "WINDOW 4.0: Documentation of Calculation Procedures." Technical Report LBL-33943, Lawrence Berkeley National Laboratory, Berkeley, CA, USA.

Fritzson, Peter, and Vadim Engelson. 1998. "Modelica – A Unified Object-Oriented Language for System Modeling and Simulation." *Lecture Notes in Computer Science*, vol. 1445.

Hoh, Alexander, Timo Haase, Thomas Tschirner, and Dirk Müller. 2005, March. "A combined thermo-hydraulic approach to simulation of active building components applying Modelica." Edited by Gerhard Schmitz, *Proceedings of the 4th Modelica conference*. Modelica Association and Hamburg University of Technology, Hamburg, Germany. unpublished.

Klein, S. A., J. A. Duffie, and W. A. Beckman. 1976. "TRNSYS – A Transient Simulation Program." *ASHRAE Transactions* 82 (1): 623–633.

Looye, Gertjan, Michael Thümmel, Matthias Kurze, Martin Otter, and Johann Bals. 2005, March. "Nonlinear Inverse Models for Control." Edited by Gerhard Schmitz, *Proceedings of the 4th Modelica conference*. Modelica Association and Hamburg University of Technology, Hamburg, Germany, 267–279.

Mattson, Sven Erik, and Hilding Elmqvist. 1997, April. "Modelica – An international effort to design the next generation modeling language." Edited by L. Boullart, M. Loccufier, and Sven Erik Mattsson, *7th IFAC Symposium on Computer Aided Control Systems Design*. Gent, Belgium.

Merz, Rolf Mathias. 2002, September. "Objektorientierte Modellierung thermischen Gebäudeverhaltens." Ph.D. diss., Universität Kaiserslautern.

Modelica Association. 2005, February. *Modelica – A Unified Object-Oriented Language for Physical Systems Modeling, Language Specification, Version 2.2*. Modelica Association.

Nytsch-Geusen, Christoph, T. Ernst, A. Nordwig, P. Schneider, P. Schwarz, M. Vetter, C. Wittwer, A. Holm, T. Nouidui, J. Leopold, G. Schmidt, U. Doll, and A. Mattes. 2005a, March. "MOSILAB: Development of a Modelica based generic simulation tool supporting model structural dynamics." Edited by Gerhard Schmitz, *Proceedings of the 4th Modelica conference*. Modelica Association and Hamburg University of Technology, Hamburg, Germany, 527–535.

Nytsch-Geusen, Christoph, Thierry Nouidui, Andreas Holm, , and Wolfram Haupt. 2005b, August. "A hygrothermal building model based on the object-oriented modeling language Modelica." Edited by Ian Beausoleil-Morrison and Michel Bernier, *Proceedings of the Ninth International IBPSA Conference*, Volume 1. International Building Performance Simulation Association and Ecole Polytechnique de Montreal, Montreal, Canada, 867–876.

Polak, Elijah, and Michael Wetter. 2006. "Precision Control for Generalized Pattern Search Algorithms with Adaptive Precision Function Evaluations." *SIAM Journal on Optimization* 16 (3): 650–669.

Sahlin, Per. 1996, May. "Modelling and Simulation Methods for Modular Continuous Systems in Buildings." Ph.D. diss., KTH, Stockholm, Sweden.

Sahlin, Per, Lars Eriksson, Pavel Grozman, Hans Johnsson, Alexander Shapovalov, and Mika Vuolle. 2004. "Whole-building simulation with symbolic DAE equations and general purpose solvers." *Building and Environment* 39 (8): 949–958 (August).

Sahlin, Per, and Edward F. Sowell. 1989, June. "A Neutral Format for Building Simulation Models." *Proceedings of the Second International IBPSA Conference*. International Building Performance Simulation Association, Vancouver, BC, Canada, 147–154.

Sowell, Edward F., and Philip Haves. 2001. "Efficient solution strategies for building energy system simulation." *Energy and Buildings* 33 (4): 309–317.

Wetter, Michael. 2004. "Simulation-Based Building Energy Optimization." Ph.D. diss., University of California at Berkeley.

Wetter, Michael. 2005. "BuildOpt – A New Building Energy Simulation Program that is Built on Smooth Models." *Building and Environment* 40 (8): 1085–1092 (August).

NOMENCLATURE

Conventions

1. Vectors elements are denoted by superscripts.
2. $f(\cdot)$ denotes a function where $(\cdot)$ stands for the undesignated variables. $f(x)$ denotes the value of $f(\cdot)$ for the argument x.

Symbols

e	error
E	energy
T	temperature
ε	solver precision parameter
ε^*	highest setting for solver precision parameter
$a \in A$	a is an element of A
$\mathbb{R}$	set of real numbers
$\triangleq$	equal by definition