ELSEVIER

Energy and Buildings 33 (2001) 309–317

ENERGY AND BUILDINGS

www.elsevier.com/locate/enbuild

Efficient solution strategies for building energy system simulation

Edward F. Sowell[a,*], Philip Haves[b]

[a]California State University, Fullerton, CA 92834, USA
[b]Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA

Abstract

The efficiencies of methods employed in solution of building simulation models are considered and compared by means of benchmark testing. Direct comparisons between the Simulation Problem Analysis and Research Kernel (SPARK) and the HVACSIM+ programs are presented, as are results for SPARK versus conventional and sparse matrix methods. An indirect comparison between SPARK and the IDA program is carried out by solving one of the benchmark test suite problems using the sparse methods employed in that program. The test suite consisted of two problems chosen to span the range of expected performance advantage. SPARK execution times versus problem size are compared to those obtained with conventional and sparse matrix implementations of these problems. Then, to see if the results of these limiting cases extend to actual problems in building simulation, a detailed control system for a heating, ventilating and air conditioning (HVAC) system is simulated with and without the use of SPARK cut set reduction. Execution times for the reduced and non-reduced SPARK models are compared with those for an HVACSIM+ model of the same system. Results show that the graph-theoretic techniques employed in SPARK offer significant speed advantages over the other methods for significantly reducible problems and that by using sparse methods in combination with graph-theoretic methods even problem portions with little reduction potential can be solved efficiently.

Keywords: Building energy systems; HVACSIM+ models; SPARK models; HVAC simulation; Computational efficiency; Graph theory applications

1. Background

Detailed simulation of building energy systems involves the solution of large sets of non-linear algebraic and differential equations. These equations emerge from component-based simulators such as TRNSYS [1] or HVACSIM+ [2], or equation-based tools such as SPARK [3] or IDA [4]. Since each of these tools employs a different solution strategy, the question arises as to which strategy is most appropriate for the kinds of equations encountered in the building simulation domain.

TRNSYS and HVACSIM+ are both based on subroutines containing algorithmic models of the underlying physics for the represented building system component. TRNSYS, the program with the longest and perhaps most wide spread usage, employs a ''block iterative'' strategy, calling the component subroutines in a sequence largely determined by the order in which they appear in the user's problem definition. Convergence is sought using successive substitution of calculated interface variables into the block inputs on the next iteration. If convergence is indeed obtained, solution is often fast since the number of iteration variables is small and there are no vector-matrix operations. However, the successive substitution method is unreliable in general, so convergence is often slow or not obtained at all. The HVACSIM+ program, which is much like TRNSYS at the problem definition level, assembles a vector of the interface variables throughout the model and employs a Newton-like solution strategy. The advantages sought with this approach are robustness and efficiency, since the information in the Jacobian allows calculation of a better next guess than the previous value alone. Indeed, provided that initial values of the interface variables are within the radius of convergence, the solution is approached quadratically. However, HVACSIM+ often is less efficient than TRNSYS in practice because of the need to calculate the Jacobian and solve the linear equation set that it represents at each iteration. Because no reduction is attempted, the size of this set is the total number of block interface variables, n_i and solving it is $O(n_i^3)$. Consequently, the more rapid and robust convergence can be overwhelmed, resulting in the longer runtimes often experienced relative to an equivalent TRANSYS model.

The IDA and SPARK modeling environments represent a new departure in that they formulate the model and its solution, in terms of equations rather than the algorithmic subroutines employed in TRNSYS and HVACSIM+. One

* Corresponding author.
E-mail address: sowell@fullerton.edu (E.F. Sowell).

0378-7788/01/$ – see front matter
PII: S0378-7788(00)00113-4

Nomenclature

c_i	scalar constant
δ	correction in Newton–Raphson iteration
f	vector of functions being solved in Newton–Raphson iteration
J	Jacobian matrix in Newton–Raphson iteration
LU	lower/upper matrix factorization
n	number of equations and variables
O($f(n)$)	order of notation (the operation in question is bounded from above by $g(n)$, where n is size of data operated on)
PI	proportional–integral control algorithm
$q_{si,j}$	heat source rate per unit surface node in discrete form of Laplace's equation
$T_{i,j}$	temperature of (i, j) node in discrete form of Laplace's equation
U	conductance
$\overline{x}$	solution vector of size n in Newton–Raphson iteration
x_i	scalar variable

advantage of this approach is that the models of individual components are input/output free. That is, the same component model can be used for a variety of different input and output designations. This allows conceptual separation of the *model* from the *problem*, the model is general, and a specific problem is defined only when a specific set of inputs is designated. Although, the two modeling environments are similar in this important respect, the solution methods employed are radically different. In IDA, the equations are formed as residual formulas, e.g. $R = f(x, y, z)$ and R is forced to zero at the solution point. Residual equations comprising math models of individual physical component are grouped into component models, with variables relevant at the system level exposed to the interface. An IDA system model consists of a set of such component models together with set of *coupling equations* that, in effect, equate equivalent interface variables at different component models. The coupling equations are all linear, of the identification form $p_1 - p_2 = 0$ or the conservation form $m_1 + m_2 - m_3 = 0$, but are large in number so that IDA equation sets tend to be quite large and sparse. For example, a simple example used in IDA reports [5] has 26 coupling equations augmenting 12 model equations. An innovative solution strategy employing sparse matrix methods in a Newton-like iterative process is used to solve the resulting large, sparse system. Because the size added by the coupling equation set is an obvious detriment to overall solution efficiency, the solver has a ''compact solution'' option for which the coupling variables and equations are, in some sense, removed. However, the expected theoretical performance improvement is not realized in the implementation, as it appears to in fact *decrease* solution speed [5]. Nonetheless, there is some anecdotal data (from informal discussions with users) to suggest that IDA may be somewhat faster than HVACSIM+ on some problems. Unfortunately, no benchmark testing results have been reported for IDA, so the actual performance remains uncertain.

Like IDA, SPARK [3] is equation-based. However, SPARK relies upon the mathematical graph for model representation and solution rather than the matrix. To support the graph, rather than expressing equations as residuals, they are expressed in the form $x = f(y, z)$, where the functions are symbolic inverses of the user-supplied model equations. This allows graph algorithms to be used to determine a sequence of function evaluations that leads to the solution. This alone is an advantage, since it eliminates the need for coupling equations entirely. Further, it allows the problem to be decomposed into separately solvable (i.e. strongly connected) components. Within each strong component, if no direct sequence is possible, as evidenced by a cyclic problem graph, a small ''cut set'' is determined so as to minimize the number of variables involved in the subsequent Newton-like iteration. As a result of these reductions, the size of the Jacobian matrix, and hence, the linear set that must be solved at each Newton iteration, is reduced, often significantly. Consequently, as will be shown in this paper, solution speed is greatly reduced.

While ideas from graph theory have been used in connection with equation system solving before [6–12], SPARK applies graph methods directly to the non-linear equations. The graph, rather than the matrix, is the primary data structure for storing the problem structure and data and as already noted, graph algorithms are employed to determine a solution sequence that operates directly on the non-linear equations. Another distinctive attribute of the SPARK approach is that the model equations are stored individually, rather than packaged into modules and are treated as equations rather than as formulae with assignment (algorithms). Symbolic methods are employed to find explicit inverses of the equations, when possible, to ensure computational efficiency. In these ways SPARK is unique. However, increasingly, simulation software is employing some of the ideas embodied in SPARK. For example, Klein, in collaboration with F. Alvarado, produced the Engineering Equation Solver [13], which employs decomposition using sparse matrix methods. This is conceptually the same as the strong component decomposition done in SPARK. However, reduction within blocks is not done in this software. In addition, TRNSYS has recently been modified to allow ''reverse solving'' [1]. This is a move toward input/output free (non-algorithmic) modeling, another tenet of SPARK. Also in the building context, Tang has applied graph-theoretic methods to improve matrix-based solution schemes [14,15].

Although, the SPARK methodology is well established, there has been relatively little systematic comparison of solution speed between SPARK and alternative methods available for solving large sets of equations, such as arise in building simulation. In order to begin to fill this gap, a simple benchmarking experiment was designed. Two problems sets were defined: (a) a replicated set of four