A New Approach to Developing Building Energy System Simulation Programs Suitable for Both Design and Optimal Operation

Jian Sun, PhD, PE  T. Agami Reddy, PhD, PE
Student Member ASHRAE  Member ASHRAE

ABSTRACT

Having building energy simulation programs with the inherent capability to efficiently perform optimization either during the conceptual design phase or during the operational stage would provide professionals with a tool to reduce energy use in building systems. This paper starts by reviewing two popular simulation techniques for building energy systems—sequential component method (SCM) and equation-oriented method (EOM)—in terms of their advantages and disadvantages. Next, a hybrid simulation strategy, successive approximation method (SAM), is proposed. This strategy is inherently capable of dealing with both design and optimal operation while combining desirable features of both SCM and EOM. SAM is a more coherent strategy that involves two aspects: (1) formulation of the equations following SAM and (2) using the automatic differentiation (AD) technique to compute the Jacobian matrix necessary to solve the equations. Finally, an illustrative example is presented to demonstrate the feasibility of this approach and how its results compare to those of a traditional simulation method.

INTRODUCTION

Buildings consume more energy than any other sector of the US economy (DOE 2003), including transportation and industry. While space heating and cooling account for 46% of all energy consumed in US residential buildings, water heating accounts for an additional 14%. Annual energy consumption for space conditioning and refrigeration in commercial buildings is 5.1 quads. Fortunately, the opportunities to reduce building heating and cooling energy use are significant. One way of reducing the energy consumption in building energy systems is to optimize system design during the conceptual phase as well as to optimally operate building systems once completed. Hence, structuring building energy system simulation programs to simultaneously allow optimization to be performed would be a desirable capability.

Over the last three decades, various building energy simulation programs, such as DOE, TRNSYS, BLAST, Energy-Plus, SPARK, HVACSIM+, HVACToolKit, etc., have been developed. Numerous studies abound in the published literature based on the above programs. Metcalf et al. (1995) combined the zone load model of IBLAST and the modular simulation approach of HVACSIM+ to create a powerful design tool. Madjidi and Bauer (1995) provided application examples to show that architectural design, HVAC preplanning and planning, HVAC component sizing, and building commissioning processes can significantly benefit from an analysis based on system simulation. Bourdouxhe (1997) used TRNSYS to simulate the cooling plant in different operating modes for reducing energy consumption. Fisher et al. (1999) categorized existing building energy simulation programs as either system based, component based, or equation based. Lebrun (2001) used an engineering equation solver, which solved the equations simultaneously, to model and simulate an HVAC system. Sowell and Moshier (1995) discussed the development of a library of equation-based models for building energy system simulation. Houte and Bulck (1994) developed a simultaneous equation-solving procedure for modeling chiller performance and discussed certain issues of partitioning, tearing, and decomposition.

Most of the simulation techniques used in building energy systems can be categorized into two broad types: sequential component method (SCM) and equation-oriented method (EOM). This paper starts by systematically reviewing these two

popular system simulation techniques. SCM allows very efficient and robust computational algorithms on both the component level and the system level. However, multiple nested iteration loops, especially when dealing with controlled constraints, often counteract its calculating efficiency. Although EOM avoids the nested iteration loops by solving the model equations, connection equations, constraint equations, and physical property equations simultaneously, its reliability is relatively poor; this is due to certain computation difficulties, such as the need to provide good starting values, the convergence property, and the large storage requirement. Additionally, both of these approaches are primarily meant for design purposes, and the few studies that have adopted them for optimal operation of an existing building have done so in a disjointed manner by artificially coupling an existing optimization code to the building simulation program.

Combining some of the beneficial features of the above methods, a hybrid simulation approach called successive approximation method (SAM) is proposed in this paper. The mathematical basis of this approach is developed, and some issues surrounding the Jacobian evaluation necessary to solve the set of equations are also discussed in detail. Finally, an example problem is presented to illustrate the feasibility, efficiency, and robustness of this approach.

SOME IMPORTANT CONCEPTS OF SIMULATION

Materials and Information Flow

A process in a real operating system involves materials flow, which includes transfer of material, such as mass and energy, among various pieces of equipment in the system. For a given system, the materials flow is unique. Figure 1 illustrates the water and air mass flow rate (\(m\)), air specific humidities (\(w\)), and temperatures (\(T\)) for the two streams of a cooling coil.

An information flow diagram, on the other hand, is a concept used during system simulation that indicates the computational sequence of variables. An information flow diagram is essentially a block diagram in which the block implies that its output can be calculated as soon as the inputs are known. One needs to distinguish between information flow and material flow diagrams since the flows may be different for the same system. Information flow diagrams are non-unique for a given system since they depend on how the simulation is performed. For example, Figure 2 depicts two different information flow diagrams for the cooling coil shown in Figure 1; the diagrams differ depending on whether or not the water flow rate is known.

Definition of Component

A typical component in the simulation program can be defined as in Figure 3, where input variable vector \(\textbf{I} = (\textbf{u}, \textbf{x})^T\) and output variable vector \(\textbf{O} = (\textbf{d}, \textbf{y})^T\). Note that \(\textbf{x}\) is the input connection variable vector between components, \(\textbf{u}\) is the input control variable vector, \(\textbf{y}\) is the output connection variable vector, \(\textbf{d}\) is the output dependent variable vector, and \(\textbf{p}\) is the parameter vector.

The mathematical description of the above component is given by:

\[
\begin{align*}
\textbf{y} &= f(\textbf{x}, \textbf{u}, \textbf{p}) \\
\textbf{d} &= d(\textbf{x}, \textbf{u}, \textbf{p}) \quad (1)
\end{align*}
\]

Information Flow Diagram of System

Any system consists of a set of components (or modules) that connect with each other. In general, there are two typical system topologies: acyclic (or sequential) system and recyclic (or feedback) system, as shown in Figures 4 and 5, respectively.

![Figure 1](image1.jpg)  
**Figure 1** Materials flow diagram of a cooling coil.

![Figure 2](image2.jpg)  
**Figure 2** Two examples of information flow diagrams of the cooling coil shown in Figure 1.

![Figure 3](image3.jpg)  
**Figure 3** A typical component.
In essence, any system simulation procedure involves solving a set of linear and/or nonlinear equations that represent the materials, energy, and thermodynamic balance in the operating process, such as mass flow, enthalpy, temperature, humidity, pressure balance, etc., of the system following a prespecified topology. In a building energy system, the set of equations usually consists of the following types (Figures 4 and 5):

1. **Model equations** that indicate the thermodynamic, heat transfer, energy, and mass balance relations.
   \[ y_i = f(x_i, u_i, p_i) \]
   \[ d_i = d_i(x_i, u_i, p_i) \quad i = 1, 2, ..., N \quad (2a) \]

2. **Connection equations** that capture the interconnected topology of all the equipment in the system.
   \[ x_i = y_j \quad i, j \in \{1, 2, ..., N\} \quad (2b) \]

3. **Constraint equations**.
   \[ h_i(x, u, p) = 0 \quad i \in \{1, 2, ..., N\} \quad (2c) \]

   Two kinds of constraints, namely, design constraints and control constraints, are often used in building energy simulations. Generally, most simulation constraints are expressed simply as
   \[ x_j = const \quad \text{and/or} \quad u_j = const \quad j \in \{1, 2, ..., N\} \quad (2d) \]

4. **Physical property equations**, which need no elaboration.

What differentiates one system simulation method from another is its specific strategy for solving the above equations. The two most popular numerical approaches used to solve the same set of model equations characterizing a building energy system simulation are sequential component method (SCM) and equation-oriented method (EOM), which will each be discussed next.
**SEQUENTIAL COMPONENT METHOD (SCM)**

Most commercial building energy system simulation programs to date are based on the SCM approach. A general building energy system consists of a series of equipment connected by fluid flow, such as water, air, refrigerant, etc. SCM treats each piece of equipment as a component and models it by a set of equations that indicate the thermodynamic and heat transfer relationships and the energy and mass balances of that equipment. Most of the components are usually modeled on standard information flow, which means that the information flow of component models closely resembles the fluid flow in the system. Therefore, the output state can be obtained by solving the model equations given the input state and the equipment parameters.

In essence, SCM uses a two-level computation scheme for simulating the system: component-level computation and system-level computation. During component-level computation, the model equations of each component, as well as physical property equations, are solved separately to obtain the performance of a single piece of equipment. System-level computation couples the component-level computation with connection equations and constraint equations to solve the performance of the whole system. Different SCM methods vary in the specific strategy or algorithm used to solve the model equations, connection equations, constraint equations, and property equations.

Component models for the various HVAC&R equipment used in buildings have been proposed by a number of researchers and have been used to create a component-level library for whole system simulation. In general, the component-level calculations can be very efficient and robust. At system-level computation, SCM breaks the recycle loop by selecting a set of “tearing” variables and handling the connection equations implicitly. For example, the numerical value for \( x_1^1 = y_1^1 \) in Figure 5. However, this cannot be determined following a sequential computation algorithm unless one knows \( x_2^1, y_2^1, x_2^1 \), and \( y_1^1 \) beforehand. An iterative technique, for example, direct substitution or accelerated substitution, can be used to converge the information flow.

Most building energy systems are simulated following the multi-loop recycle system shown in Figure 5. Generally, at least one torn variable vector is selected for each recycle loop. For complex systems, an optimal tear strategy for finding the optimal torn variable set is necessary but is not straightforward. Gundersen and Hertzberg (1982) proposed some criteria for making such a choice; these criteria have been explained further by Houte and Bulck (1994). Barkley and Motard (1972), Upadhye and Grens (1975), and Genna and Motard (1975) also proposed several tear criteria. The problem is relatively simple for most building cooling energy systems that consist of three recycle loops: condenser water loop, chiller water loop, and air distribution loop. The torn variable vector can often be determined by inspection in such simple cases with one torn variable vector for each loop.

**EQUATION-ORIENTED METHOD (EOM)**

EOM, or simultaneous simulation method, treats all the equations, including the model equations, connection equations, constraint equations, and physical property equations, as one large system of equations and solves them simultaneously. Model equations used in EOM are formulated differently from those used in SCM. The independent variables of model equations in SCM are just the input-output state variables (external variables), while in EOM, all variables, including internal and external variables, are independent variables. Physical property equations are often treated as a specific kind of model equation or calculated by additional subroutines.

Compared to SCM, EOM avoids the nested iteration loops, therefore reducing computation cost. Another attractive characteristic of EOM is that connection equations and constraint equations become simple equations within the larger equation system. There is no need to tear some connection variable or add additional iteration loops for constraint equations. This offers greater flexibility and ease for system optimization (Sun and Reddy 2005) since equations can be used as constraints in a generalized, nonlinear programming problem.

Theoretically, EOM is a very promising approach in system simulation because any nonlinear equation solution algorithm can be used to solve the equation system. However, only a few building energy system simulation programs are based on this method or its variants. The main reason for this is that model equations of certain components in building energy systems are so complicated and strongly nonlinear that the convergence of EOM may be very poor. Therefore, EOM is usually applied to simpler models. In addition, unless good initial iteration variables are provided, the reliability of this method may be suspect. Another problem is that much larger computer memory storage is required for EOM compared to SCM due to the larger number of independent variables and equations being solved simultaneously.

**HYBRID APPROACH: SUCCESSIVE APPROXIMATION METHOD (SAM)**

As discussed previously, SCM allows very efficient and robust algorithms on both component-level and system-level computation. However, multiple nested iteration loops, especially when dealing with the constraints, often counteract the calculation efficiency. EOM avoids the problem by solving all the equations simultaneously, which offers much greater speed and flexibility, especially when dealing with complex system and design simulation. Additionally, from the standpoint of system optimization, SCM would require adding another outer iteration loop, further exacerbating the numerical efficiency problem discussed above. Thus, greater computational expense would be incurred were SCM to be used with system optimization. These seems to be merit in an approach that combines the desirable features of both EOM and SCM; such an approach, successive approximation method (SAM), is described in the following paragraphs.
Mathematical Background

The concept behind SAM involves solving a sequence of simplified system models consisting of linear and/or nonlinear approximation that approximates the original system model. Similar to SCM, SAM involves two levels of computation compared to the one level of computation in EOM. Further, SAM solves all equations simultaneously, as EOM does, instead of solving them sequentially as SCM does. However, the dimension of the equation system in SAM is much smaller than that in EOM since internal variables in the former approach no longer appear as system equations. In the component-level computation, rigorous models are often used to generate the coefficients for building the approximation model. The detailed mathematical description follows.

Consider Equation 2a. First-order Taylor series expansion of \( y_i \) around the current iterative point \( x_i^k \) yields

\[
y_i = f(x_i^k, u_i, p_i) + \frac{\partial f}{\partial x_i} |_{x_i=x_i^k} \Delta x_i + 0(|\Delta x|^2, |\Delta u|^2)
\]

where \( \Delta x = x_i - x_i^k \).

Then, linear approximation models for each component around the current iterative point \( x_i^k \) are given by

\[
y_i \approx A_i x_i + C_i
\]

where \( A_i = \left[ \frac{\partial f}{\partial x_i} |_{x_i=x_i^k} \right] \) and \( C_i = f(x_i^k, u_i, p_i) - \frac{\partial f}{\partial x_i} |_{x_i=x_i^k} x_i^k \).

Combining all the component models yields the linear approximation model for the entire system around the current iterative point.

\[
y \approx Ax + C
\]

where diagonal matrices \( A = \begin{bmatrix} A_1 & \cdots & A_N \end{bmatrix} \) and \( C = \begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_N \end{bmatrix} \),

where \( A_1, \ldots, A_N \) and \( C_1, \ldots, C_N \) are themselves matrices of individual components.

Obviously, the constraint and connection equations can be handled easily by combining all the simplified model equations (the linearized equations), connection equations, and constraint equations into one large equation system.

From Equation 2b we define a relevant binary (either 1 or 0) connection matrix \( R \) (Sun and Reddy 2005) between \( x \) and \( y \) for the system as

\[
y = Rx.
\]

If no constraint equations exist, solving Equations 5 and 6 yields

\[
x = [(R - A)^T (R - A)]^{-1} (R - A)^T C.
\]

If constraint equations given by Equation 2c are present, we use the approximation

\[
\Delta x + Eu_c + F \approx 0 \text{ with } u = \{u_p, u_c\}
\]

where \( D = \left[ \frac{\partial h}{\partial x} \right]_{x=x^p}, E = \left[ \frac{\partial h}{\partial u_p} \right]_{u_p=u_p^c}, F = h(x^k, u^c) - \left[ \frac{\partial h}{\partial x} \right]_{x=x^k} x^k + \left[ \frac{\partial h}{\partial u_p} \right]_{u_p=u_p^c} u^c
\]

and \( u_c \) are controlled input variables whose values are given and \( u_p \) are unknown controlled input variables.

We rewrite the model equations as

\[
y_i = f(x_i, u_i, p_i) \quad i = 1, 2, ..., N
\]

where \( u_i = \{u_{i,j}, u_{i,k}\} \), \( i_c = 1, 2, \ldots, N_c, \quad i_s = 1, 2, \ldots, N_s \), \( N_c + N_s = N \), \( N_c \) is the number of unknown controlled input variables, and \( N_s \) is the number of known controlled input variables.

Next, we define \( z = [x, u_i]^T \). From a first-order Taylor series expansion, the approximation model given by Equation 9 is now

\[
y_i = f(x_i, u_i, p_i) + \frac{\partial f}{\partial x_i} \Delta x_i + \frac{\partial f}{\partial u_i} \Delta u_i + 0(|\Delta x|^2, |\Delta u|^2)
\]

or

\[
y_i \approx A_i z + C_i
\]

where

\[
A_i = \begin{bmatrix} \frac{\partial f}{\partial x_i} |_{x_i=x_i^k} \\ \frac{\partial f}{\partial u_i} |_{u_i=u_i^c} \end{bmatrix},
C_i = f(x_i^k, u_i^c, p_i) - \frac{\partial f}{\partial x_i} |_{x_i=x_i^k} x_i^k - \frac{\partial f}{\partial u_i} |_{u_i=u_i^c} u_i^c
\]

Finally, the entire system for the case with constraint equations present can be expressed as:

\[
y = Az + C
\]

\[
y = Rx
\]

\[
Dx + Eu_c + F \approx 0
\]

The above linear equation system can be solved by any Newton type algorithm that yields \( x \) and \( u_c \).

Jacobian Evaluation

One of the most critical factors that affects the performance of SAM is the accuracy with which the Jacobian matrix is evaluated. A poor approximation of Jacobian evaluation may result in the SAM being numerically inefficient. The easi-
Then the doublet of a derivative is based on the fact that the derivative of a function defines the best local linear approximation to the function near a given point. One of the disadvantages of using the finite difference estimation of a derivative is the difficulty in selecting a reasonable finite difference interval \( \eta \) to produce acceptable approximations. Inappropriate choices often lead to poor optimization results. Though no general procedure can guarantee satisfactory results, a standard choice is to select the interval that can minimize the total errors (Gill et al. 1981; Bartholomew-Biggs 1998). Another more serious disadvantage of finite difference estimation of each variable requires at least two evaluations of the function. If the simulation plant has highly nonlinear models and complicated functions, the accuracy and computing cost will present difficulties under the finite difference approach. This is why recent optimization codes prefer to use another promising differential technique, automatic differentiation.

The automatic differentiation (AD) technique is based on the mathematical concept of differential algebra, where the original function is decomposed into simpler, elementary functions whose gradient vector and Hessian matrix are easy to derive by applying first-order and second-order derivatives (Bartholomew-Biggs et al. 2000). The implementation is the repeated use of the chain rule in elementary calculus, but applied to floating point numerical values rather than to symbolic expressions. A new data type, doublet (Bartholomew-Biggs 1998), is introduced and defined as an ordered pair,

\[
\hat{u} = \{u, Vu\}.
\]  

Then the elementary mathematical operations (such as addition, subtraction, multiplication, division, logarithm...) can be extended to doublet. For example, multiplication and logarithm functions can be written as:

\[
\hat{u} \times \hat{v} = \{uv, u\nabla v + v\nabla u\}
\]

\[
\log u = \{\log u, \nabla \log u/u\}
\]

Assuming \( y \) to be a variable depending on \( n \) independent variables \( x_i \),

\[
y = f(x).
\]

Then the doublet of \( y \) is \( \hat{y} \) given by

\[
\hat{y} = \{y, \nabla f\}.
\]

The independent variables \( x_i \) (where \( i = 1, 2, ..., n \)) also form a doublet element:

\[
\hat{x}_i = \{x_i, e_i\}
\]

where \( e_i \) is the \( i \)th column of the \( n \times n \) identity matrix.

There are two primary approaches to AD: forward mode and reverse mode. Forward mode is a straightforward implementation of the chain rule as applied to the original function. The initial doublet of independent variables is given by Equation 20. The procedure to decompose the original function is

\[
x_i = f(x_j) \quad j \in J_i \quad J_i \subset \{1,2,...,i-1\} \quad i = n + 1, n + 2, ..., m
\]

\[
y = x_m.
\]

The derivatives are accumulated as the program proceeds:

\[
\forall x_j = e_j \quad i = 1, 2, ..., n
\]

\[
\forall x_i = \sum_{j \in J_i} \frac{\partial f}{\partial x_j} \forall x_j \quad J_i \subset \{1,2,...,i-1\} \quad i = n + 1, n + 2, ..., m
\]

\[
\forall f = \forall x_m.
\]

The doublet of auxiliary variables are given by

\[
\hat{x}_i = \{x_i, \nabla x_i\} \quad i = n + 1, n + 2, ..., m.
\]

In most simulation processes, the independent and dependent variables are often implicit functions. It is necessary to apply the automatic differentiation technique to implicit functions (Bartholomew-Biggs 1998). The idea of solving an implicit function by constructing an iterative equation for the dependent variables can also be extended to the doublet. Suppose the independent and dependent variable vectors are given by

\[
x = [x_1, x_2, ..., x_n]^T
\]

\[
y = [y_1, y_2, ..., y_p]^T.
\]

The implicit equation is

\[
f(u, y) = 0
\]

where \( u = [u_1(x), u_2(x), ..., u_m(x)]^T \).

In order to solve the above nonlinear system, the equation for the \( k \)th iteration can be defined as

\[
f(u^k, y^k) - h(u^k, y^k) = 0
\]

or

\[
y^{k+1} = h(u^k, y^k)
\]

where \( h(u, y) \) depends on the iterative algorithm.

Performing the doublet arithmetic on Equation 29 yields a sequence,

\[
y^{k+1} = h(u^k, y^k),
\]

with

\[
\hat{u}^k = \{u^k, u^k_x\}
\]

\[
\hat{y}^k = \{y^k, y^k_x\}
\]

\[
h(u^k, y^k) = \{h(u^k, y^k), h(u^k, y^k u^k_x) + h(u^k, y^k) y^k_x\}.
\]

Thus, the iterative equation for the derivative is

\[
y^{k+1} = h(u^k, y^k)u^k_x + h_y(u^k, y^k) y^k_x.
\]
VERIFICATION OF NUMERICAL SOLUTION METHODOLOGY

In order to demonstrate the numerical accuracy of the proposed methodology (SAM) a typical two-zone building (zone 1: 5000 ft$^2$ and zone 2: 4000 ft$^2$) from Knebel (1983) is selected as the reference building. The assumed diurnal variation of the sensible loads and latent loads of each zone, as well as that of the ambient temperature, are shown in Figure 6. A constant air volume terminal reheat system, which consists of a central air-handling unit with fan, cooling coil, and mixing box, is used to supply cooling to the two zones of the building. The air leaving the cooling coil is to be controlled at a fixed setpoint. The air leaving the unit is delivered to the zone reheat coils, which heat the zone supply as needed.

We shall compare the results calculated by the proposed SAM method with the benchmark, HVAC$^2$Toolkit (Brande-muehl 1993), for the constant air volume terminal reheat system so as to evaluate its accuracy, efficiency, and robustness. The information flow diagram used for the SAM program is shown in Figure 7. Some important state points are indicated: $S_1$ and $S_2$ denote the supply air condition, $Z_1$ and $Z_2$ the zone air, $R$ the return air, $O$ the outdoor air, $M$ the mixing air, $C$ the cooling/heating air, and $F$ the fan outlet air conditions. Table 1 lists the input variables with numerical values used in this illustrative example.

The relevant equations of the selected constant air volume terminal reheat system follow.

**a. Model Equations**

\[ y_i = f(x_i, u_i, p_i) \]
\[ d_i = d(x_i, u_i, p_i) \quad i = 1, 2, ..., 7 \]

where $x_1 = [T_{air}, w_{air}]$
\[ x_2 = [T_{ce}, w_{ce}] \]
\[ x_3 = [T_{fe}, w_{fe}] \]
\[ x_4 = [T_{rehe1}, w_{rehe1}] \]
\[ x_5 = [T_{rehe2}, w_{rehe2}] \]
\[ x_6 = [T_{z1}, w_{z1}] \]
\[ x_7 = [T_{z2}, w_{z2}] \]
\[ y_1 = [T_{am}, w_{am}] \]
\[ y_2 = [T_{cl}, w_{cl}] \]
\[ y_3 = [T_{fr}, w_{fr}] \]
\[ y_4 = [T_{rhe1}, w_{rhe1}] \]
\[ y_5 = [T_{rhe2}, w_{rhe2}] \]
\[ y_6 = [T_{z1}, w_{z1}] \]
\[ y_7 = [T_{z2}, w_{z2}] \]
\[ p_1 = [R_{ao}, T_{ao}, w_{ao}] \]
\[ p_2 = [T_{io}, U_{A_{in}}, U_{A_{ext}}] \]
\[ d_2 = [Q_c, T_u] \]
\[ u_2 = [m_1] \]
\[ p_3 = [f_{ce}, P_{ce}, c_{me}, f_{ml}] \]
\[ d_3 = [P_3] \]
\[ p_4 = [T_{rhe1}, U_{A_{rhe1}}] \]
\[ d_4 = [Q_{rhe1}, T_{rhe1}] \]
\[ u_4 = [m_{rhe1}] \]
\[ p_5 = [T_{rhe2}, U_{A_{rhe2}}] \]
\[ d_5 = [Q_{rhe2}, T_{rhe2}] \]
\[ u_5 = [m_{rhe2}] \]
\[ p_6 = [U_{A_{r1}}, R_{rl1}, T_{ao}, w_{ao}] \]
\[ p_7 = [U_{A_{r2}}, R_{rl2}, T_{ao}, w_{ao}] \]

**b. Connection Equations**

\[ y_1 = x_2 \]
\[ y_2 = x_3 \]
\[ y_3 = x_4 = x_5 \]
\[ y_4 = x_6 \]
\[ y_5 = x_7 \]
\[ y_1 = f(y_6, y_7) \]

**c. Constraint Equations**

\[ T_{arl} = T_{z1, set1} \]
\[ T_{ar2} = T_{z2, set2} \]
\[ T_{fr} = T_{c, set} \]

Note that vectors $x = [x_1, x_2, ..., x_7]^T$ and $y = [y_1, y_2, ..., y_7]^T$ have 14 elements, 7 of which are air temperature and 7 are air humidity, for the state points shown in Figure 7. Control input variable $u_c = [u_2, u_4, u_5]^T$ and $z = [x^T, u_c^T]^T$. The relative binary connection matrix $R$ ($14 \times 14$) of this system is given by Table 2. From the connection equation $y_2 = x_3$, we note that $y_2$ have two elements $[T_{cl}, w_{cl}]$, which are the third and forth elements of vector $y$; and $x_3$ have two elements $[T_{fe}, w_{fe}]$, which are the fifth and sixth elements of vector $x$. Thus the elements (3, 5) and (4, 6) of $R$ are unity.
Table 1. List of Input Variables with Numerical Values

<table>
<thead>
<tr>
<th>Input Parameter Variables</th>
<th>Unit</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>type of air-side system</td>
<td>Multi-zone constant air</td>
<td></td>
</tr>
<tr>
<td>(two zone)</td>
<td>volume plus reheat</td>
<td></td>
</tr>
<tr>
<td>economizer control</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>cooling coil overall</td>
<td>$U_{A_{int}}$ W/$^\circ$C</td>
<td>1.6656e+004</td>
</tr>
<tr>
<td>internal UA/total external area</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cooling coil overall</td>
<td>$U_{A_{ext}}$ W/$^\circ$C</td>
<td>8.7440e+003</td>
</tr>
<tr>
<td>external dry UA/total external area</td>
<td></td>
<td></td>
</tr>
<tr>
<td>reheat coil 1 UA/total area</td>
<td>$U_{A_{r1}}$ W/$^\circ$C</td>
<td>5.7338e+003</td>
</tr>
<tr>
<td>reheat coil 2 UA/total area</td>
<td>$U_{A_{r2}}$ W/$^\circ$C</td>
<td>5.7338e+003</td>
</tr>
<tr>
<td>rated volumetric flow rate</td>
<td>$f_r$ m$/^3$/s</td>
<td>5.664</td>
</tr>
<tr>
<td>rated shaft power</td>
<td>$P_{r}$ W</td>
<td>17700</td>
</tr>
<tr>
<td>motor drive efficiency</td>
<td>$e_m$</td>
<td>0.85</td>
</tr>
<tr>
<td>fraction of motor heat loss to fluid stream</td>
<td>$f_{ml}$</td>
<td>1.0</td>
</tr>
<tr>
<td>setpoint temp. for zone 1</td>
<td>$T_{z, set1}$ $^\circ$C</td>
<td>24</td>
</tr>
<tr>
<td>setpoint temp. for zone 2</td>
<td>$T_{z, set2}$ $^\circ$C</td>
<td>24</td>
</tr>
<tr>
<td>return air ratio</td>
<td>$r$</td>
<td>0.8</td>
</tr>
<tr>
<td>entering liquid temp.</td>
<td>$T_{le}$ $^\circ$C</td>
<td>4.4</td>
</tr>
<tr>
<td>reheat coil 1 liquid</td>
<td>$T_{rhl1}$ $^\circ$C</td>
<td>30.0</td>
</tr>
<tr>
<td>entering temp.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>reheat coil 2 liquid</td>
<td>$T_{rhl2}$ $^\circ$C</td>
<td>30.0</td>
</tr>
<tr>
<td>entering temp.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cold deck temp. setpoint</td>
<td>$T_{c, set}$ $^\circ$C</td>
<td>12.8</td>
</tr>
<tr>
<td>zone 2 supply air mass flow rate</td>
<td>$m_{z2}$ kg/s</td>
<td>1.4</td>
</tr>
<tr>
<td>zone 2 supply air mass flow rate</td>
<td>$m_{z2}$ kg/s</td>
<td>2.7</td>
</tr>
</tbody>
</table>

SAM simulates the reference system of equations by approximating them to a linear system at each iterative point and calculating matrix $A$ and vector $C$. The numerical values of $A$ and $C$ for the last hour of simulation are listed in Table 3. The approximated linear system is solved to yield a new set of variables that will be used to update the system coefficient matrix until a predefined convergence criterion is satisfied.

The results of one day’s simulation are shown in Figures 8a–8d, which present comparison between the SAM codes and the benchmark, HVACToolkit. These include a comparison of cooling coil total heat transfer, coil sensible heat transfer, coil leaving air dry-bulb temperature, liquid (cooling water) mass flow rate, mixed air temperature, and liquid (cooling water) leaving temperature. Differences between simulation results predicted by SAM and the benchmark are within 2% of relative errors.

SUMMARY

We started by reviewing two simulation approaches, the sequential component method and equation-oriented method. The former is computationally efficient and robust at both the component level and the system level, but it is unfortunately inflexible in dealing with a multiple-loop system. The equation-oriented simulation method is more flexible and better suited for system optimization, while its disadvantages lie in poor convergence property, sensitivity to initial guessed values, and large storage requirement. A method that combines the best features of both methods, the successive approximation method, is proposed in this paper to serve as the basis for simulating building energy systems with the inherent capability of efficiently performing optimization either during the conceptual design phase or during the operational stage of the building system. An efficient solution algorithm to evaluate the Jacobian matrix, the automatic differentiation technique, used in other engineering disciplines, is also discussed in detail. Finally, an example illustrates SAM and compares its results with those from a widely accepted traditional simulation program. The methodology can also be extended to deal with more complete and complex energy systems as well as dynamic system simulation.

NOMENCLATURE

$e_m$ motor drive efficiency
$f_{ml}$ fraction of motor heat loss to fluid stream
$f_r$ rated volumetric flow rate, m$/^3$/s
$m_1$ liquid mass flow rate, kg/s
$m_{rhl}$ reheat coil liquid mass flow rate kg/s
$P_r$ rated shaft power, W
$P_f$ fan Power, W
$R_{ar}$ return air Ratio
Table 3. Matrix A and Vector C

<table>
<thead>
<tr>
<th>Matrix A</th>
<th>Vector C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8026 -4.3093 50.5714 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>2.5632 0 6.0778 0 7.8041 0.0004 4.8013 0.0004 4.6966 0.0024 2.0932 0.0034 1.7359 0 12.8 12.8 24</td>
</tr>
</tbody>
</table>

- **Matrix A**
- **Vector C**

(a) Cooling coil total heat transfer.

(b) Cooling coil sensible heat transfer.

(c) Liquid (cooling water) mass flow rate.

(d) Liquid (cooling water) leaving temperature.

Figure 8 Simulation accuracy of SAM compared with the benchmark, HVAC2Toolkit.
REFERENCES


