Symbolic Processing and Code Generation of Equation Based HVAC&R Simulation Models

Per Sahlin, Ph.D.
Member ASHRAE

Pavel Grozman, Ph.D.

Modeling and simulation are indispensable in current HVAC&R research and engineering, but lack of interoperability between different tools create serious inefficiencies. Standard languages for model expression have recently emerged. They have the same prospects of dramatic efficiency improvements as other software standards. The Neutral Model Format (NMF) is such a standard for equation based expression of building simulation models. In a recent ASHRAE research project, an NMF translator with automatic code generation for the simulation environments TRNSYS and HVACSIM+ was developed. This work is presented along with a general discussion about the technical and economical aspects of improved simulation infrastructure.

INTRODUCTION

The quality and performance of built HVAC&R systems are increasingly depending on the computer tools that were used to design them. A proficient engineer must today not only master the relevant physics, but must also have a more than superficial insight into modeling and simulation technology. This shift is also reflected by the steadily increasing amount of time that researchers spend on computer modeling as a complement to traditional lab work. For HVAC&R research funders, the overall efficiency and reusability of modeling work is crucial. As we will show, the situation today is far from satisfactory. Significant research and development efforts are underutilized. The purpose of this paper is to discuss necessary standards and software infrastructure to remedy this problem. In particular, results from a recent ASHRAE research project (RP839) that deals with automatic translation of equation based simulation models into executable code will be presented.

An Example to Indicate the Magnitude of the Problem

Quantifying the cost of a missing standard is obviously close to impossible. We will nevertheless attempt to provide some indicators by means of an example. A very successful ASHRAE research project was RP629 *A Toolkit for Secondary HVAC Systems*, the first of a series of three similar projects that have been initiated by Technical Committee 4.7. The purpose of this project was to collect, evaluate, package and document component models for secondary HVAC equipment such as cooling coils, fans and various heat exchangers. Since the whole idea of the project was to provide easily accessible model material for simulation, considerable effort was spent on finding a suitable presentation format. A subroutine-based format in FORTRAN 77 was selected and guidelines for the written account of the models were settled. The total cost of the project was $ 65 000, and the main result is a well appreciated ASHRAE publication with accompanying software (Brandemuehl 1992).

Utilizing these results in practical simulations is however still quite laborious. The simplest thing to do is to run a single component, with given input parameters and boundary variables. A small subroutine driver is provided in the package for this case. However, in order to make the toolkit truly useful, arbitrary combinations of toolkit components must be connected in the structure of the air-handling unit to be studied. This is considerably more involved, since some numerical procedures, usually beyond the capability of most HVAC engineers, have to be applied. The most natural thing to do in this situation is to rely on one of several available modular simulation environments (MSE), which are specifically intended for simulation of arbitrarily complex networks of such component models. We will return to a more thorough discussion of MSEs in a subsequent section.

The problem that now arises is that no modular simulation environment can call the toolkit subroutines as they are, since the call structure was set up specifically for the project and not in the required format of any single MSE. A manual conversion must be done, either by rewriting the models in the required format of the target environment or by writing a wrapper model in the target format that internally calls the toolkit models. At least three such conversion projects have been carried out for the ASHRAE secondary toolkit.

TRNSYS	Rewrite of subroutines into TRNSYS format.
SPARK	A very thorough rewrite of the models into the SPARK format (Sowell 1995).
NMF	A mixture of rewrite and wrapping, forming a functional but not optimal NMF edition of the toolkit models (Hyttinen and Vuolle 1997).

Although no exact figures are available, these translation projects have collectively amounted to more than $ 50 000. This additional effort could have been completely spared, had the models originally been cast in a standardized symbolic format. Within ASHRAE, many projects with similar software deliverables as the toolkits are carried out each year. At HVAC&R research laboratories world wide, significant efforts are effectively wasted in the same fashion.

The Neutral Model Format (NMF) - Background

In 1989, a draft standard format for equation based component models, NMF, was proposed to the building simulation community (Sahlin and Sowell 1989). NMF has two main objectives: (1) models can be automatically translated into the local representation of several simulation environments, i.e. the format is program *neutral* and machine readable, and (2) models should be easy to understand and express for non-experts. In addition to the standard component format itself, three things are needed in order to create a truly efficient infrastructure for HVAC&R computing:

1. *A reasonable selection of modular simulation environments that are appropriate for HVAC&R modelling*. These exist, and an overview will be given in the next main section.
2. *Automatic translators*. An objective of this paper is to discuss translator implementation in some detail.

3. *Documented and globally available model repositories*. Significant NMF model libraries have been developed and made available on the World Wide Web (WWW), as will be briefly discussed in a subsequent section.

Establishing a de facto standard can be a long process. Unfortunately, NMF is no exception to this. A year after the first paper, the first formal description of the language was written (first published in Sahlin, Bring and Sowell, 1992) and the first automatic translator (to SPARK) was written shortly thereafter (Nataf 1990). The first significant NMF model library dealt with consumer substations in district heating networks (Kolsaker 1990). In 1992, an ad hoc subcommittee of ASHRAE TC 4.7 was formed to foster ASHRAE usage and to act as an open forum for approval of NMF amendments. An ASHRAE work statement (839) to develop a common NMF parser and code generators for the TRNSYS and HVACSIM+ simulation environments was issued early 1994 and the main deliverables of this project were approved in 1996. The first globally accessible NMF simulation model network (SIMONE, see also Section Other Tools and Models) came on line in 1998.

The main software tools are now in place. The next issue becomes that of building sufficient engineering confidence in these tools, to go from an experimental concept to standard software for everyday use.

NMF models are based on algebraic and ordinary differential equations. Initiated engineers generally raise two main concerns. (1) Can my most complex models really be expressed as equations? (2) Is the automatic conversion from equations to executable algorithms safe and efficient? The answer to both questions is generally, Yes. In this paper, we will concentrate on the second one by explaining some of the internal details of a symbolic NMF translator. The first question is best answered by the existence of truly complex models. Some sources of such models will be pointed out.

Why HVAC&R?

The issue of efficient management of simulation models is clearly not specific to HVAC&R. A domain independent standard would surely be preferable, given the fact that many fields then could contribute to the development of processing tools and model storage structures. Such standardization work is also underway.

Modelica is a unified object-oriented language for physical systems modeling. The ambition is to create a domain independent standard for both continuous and discrete simulation models. A first version, for equation based models in continuous time was issued in September 1997 (Elmqvist et al. 1997). Modelica is based on the most important concepts of a range of previous modeling languages for equation based models. Many of the key features of NMF have found their way into Modelica; others are still missing, but will most likely be present in future editions. Technical Committee 1 of EuroSim presently controls Modelica.

Another recent standardization project is VHDL-AMS (IEEE 1997), an extension to the electronic discrete circuit modeling language VHDL for combined continuous and discrete models. VHDL-AMS is a large and rich modeling language targeted mainly at the application domain of electronics hardware. It is unlikely that VHDL-AMS will be found to be suitable to HVAC&R modeling and simulation.

NMF has a set of features that have been specifically composed to suit the needs of building simulation and to enable development, within reasonable resources, of complete-language translators to commonly used building simulation environments. Development of corresponding Modelica tools is considerably more involved, since the language itself is much more complex in order to cater for the needs of all potential application domains. At the moment, Modelica has excellent momentum and the prospects of success are good. It is reasonable to assume that Modelica, with time, will have sufficient features to enable automatic translation of NMF models into source code quality Modelica.

However, the option of discontinuing NMF library development to await Modelica seems highly irrational. Of paramount importance is to have libraries in *any* sufficiently rich symbolically processable format. Translation to another fundamentally equivalent format is a technical issue of minor consequence in comparison to the investments that are continuously being made in model libraries such as the ASHRAE toolkit series.

The next three sections are devoted to an overview of modular simulation environments, some fundamental associated problems, and an introduction to NMF. The already initiated reader may omit these without loss of continuity. After that, we will present the mathematical problem of converting equation based models into algorithmic form and some details about algorithms used in the ASHRAE NMF Translator that was developed in the Technical Research Project 839. The paper is concluded by an overview of the translator and other available NMF resources.

MODULAR SIMULATION ENVIRONMENTS

Physical systems that are simulated in Modular Simulation Environments (MSEs) are modular in nature, i.e. they naturally decompose into subsystems. Frequently, identical subsystems are repeated a number of times in a model, a fact that is taken advantage of in many tools. Furthermore, the systems should have a basically continuous behavior, meaning that equations used to describe them, as well as forcing functions, will have a limited number of discontinuities. Purely event driven systems are excluded.

If characterized by equations, the physical systems under consideration will require both algebraic and differential equations. Differential equations can be either ordinary (ODE) or partial (PDE), although current tools, and the present NMF, require that PDEs are explicitly discretized in space and thus turned into ODEs. Note that in contrast to many widely used commercial tools, the simulation environments we are concerned with here are not limited to ODEs only. They allow a free mixture of algebraic and ordinary differential equations generally referred to as differential-algebraic systems of equations (DAE).

Furthermore, the simulation tools under discussion are rarely used for applications where a strict formalism for generating governing equations exists. In, e.g., electrical circuit analysis or structural analysis special purpose systems may be more advantageous.

Separation of Modelling and Solving Activities

In contrast to mainstream design tools in building simulation, MSEs separate strictly between the modelling and subsequent system solution activities. A modelling tool is often used for model formulation. This tool generates a system model, generally expressed in a *modelling language*. The model is then treated by a solver. An important benefit of a separate solver is that it may be altered or even exchanged with minimal interference with the modelling environment. Some MSEs rely on regular programming languages as part of their system model description. For these, component models are typically described as subroutines with prescribed structure, while interconnection of pre-programmed component models into system models is described with a dedicated language. Other environments have complete modelling languages, which describe component as well as system behavior.

Key characteristics of the modelling language, such as expressiveness and level of standardization, are critical to the usefulness and development potential of the overall MSE. The Neutral Model Format is part of such a modelling language. NMF can be translated into *subsets* of several complete target modelling languages; it does not cover all constructs in any of the present targets, just a sufficient “common denominator.”

Available and Emerging MSEs

Most of the simulation tools under discussion are intended for quite sophisticated users, who are well versed in mathematical modelling, numerical methods and advanced use of computers. These tools are not directly suited for designers, without special simulation expertise, that use simulation as one of several methods for design evaluation. However, for the expert, they generally provide an efficient environment for model building, simulation and analysis.

Other tools, e.g. EKS and IDA, are primarily intended for efficient design tool production, and the normal end user will rarely interact directly with the underlying MSE techniques.

A few tools and environments with the discussed main characteristics are already matured and available and others are under development. A selection of MSEs, which have some relation to building simulation and/or NMF, is presented in Table 1.

Table 1. A selection of Modular Simulation Environments

FRAGMENTED MODEL REPOSITORIES

Currently, each simulation environment has its own library of models. Porting models between environments is generally a major undertaking and for this reason very few comparative studies of the relative merits of different environments exist. A benchmarking exercise must be based on real-scale models in order to be of interest. Carrying out such an exercise may easily take more than a person-year of effort.

Another problem that hampers construction of re-usable model libraries even within the user community of a single simulation environment is that component models frequently have fixed signal directions, i.e., the input and output signals have been designated already at the component modelling stage.

Input-Output Oriented Models - Pipe Model Example

A simple model of a pipe might contain a single relation between the massflow, *m*, and the pressure at each end of the pipe, say *p1* and *p2*. In an input-output oriented environment such a model must occur in two algorithmic versions in order to be useful in all contexts. One calculates *m*, given *p1* and *p2*. Another calculates *p1*, given *p2* and *m*. In order to connect, e.g., two pipes in series, one pipe of each kind must be utilized. For more complex models, with several ports, e.g. a heat exchanger, even more versions of essentially the same model are required. Examples of such trivial model repetition are common in many model libraries such as the ASHRAE Secondary HVAC Systems Toolkit or the TRNSYS library.

Most of the commercially dominating general-purpose tools for continuous simulation, e.g., SimuLink, SystemBuild, VisSim, and others, are based on an input-output oriented block diagram description of the simulated system. This is one reason for their limited applicability on HVAC&R simulation problems.

ODE Based Models

The majority of commercially dominating general-purpose tools are limited also to only ordinary differential equations. If algebraic equations exist in a system, they must not form loops that go from one component to the next and then back again, a so called *algebraic loop*. This is another inconvenient limitation for HVAC&R modelling.

Let us illustrate this by returning to the pipe example and add a T-piece without pressure loss to the library. The model would add incoming massflows, and make sure all pressures on connected ports are equal (no internal pressure loss). (In an input-output oriented setting, we need more than a single version of this model as well.) We are now in a position to build arbitrary pipe networks and study distribution of flow and pressure throughout the system. However, many interesting systems will contain an abundance of algebraic loops, since they often form closed loop systems. Only the special case of systems completely without internal threads can be treated without creation of algebraic loops.

In order to study a threaded system in an ODE based tool, dynamics have to be introduced, that break each created loop. In our example, this means that a pressure drop in a pipe no longer instantaneously will give rise to a massflow. A special pipe that models the inertia and elasticity of the fluid must be introduced in order to break the algebraic loops. This model will in turn create very fast transients in the, now dynamic, system. These transients will have to be numerically resolved prior to getting to the steady-state situation that we were aiming at.

In conclusion, ODE based simulation environments cannot solve algebraic loops. This makes it necessary to add spurious models with dynamics in order to study threaded systems.

This in turn creates additional (trivial) complexity in the model library and thereby impedes effective model reuse.

Several NMF compatible MSE:s are input-output free, i.e. the designation of given and calculated variables is done as late as possible, at the system modelling stage. None of the present target environments are limited to ODE:s.

THE NEUTRAL MODEL FORMAT - TECHNICAL BASICS

In NMF, internal component model behavior is described by a combination of algebraic and ordinary differential equations. Equations may be written in any order and in the form

<expression> = <expression>;

NMF only *states* equation models, while *solution* of equations is, in some cases, left to the target environment (e.g. IDA, or SPARK), or the NMF translator in others (e.g. TRNSYS, or HVACSIM+).

NMF supports model encapsulation through a link concept, i.e. models may only interact via variables appearing in LINK statements. To enhance and encourage model plug compatibility, links and variables are globally typed. The idea is that a basic list of such types should be included in each revision of the standard, but that users may add to the list as need arise. A selection of such global types is:

Figure 1. NMF code for global declarations

A quantity type includes a physical unit and information about potential (across) or flow (through) type. A link type is simply an ordered list of quantity types. Let us now look at an example of an NMF model, a homogeneous wall (Figure 2) using the heat equation in one dimension.

Figure 2. A homogeneous wall divided into several layers

$$\frac{¶\, T}{¶\, t} = \frac{l}{r\, c_p}\frac{¶\,^{2}T}{¶\, x^{2}} \qquad \text{PDE} \qquad (1)$$

$$Q_a = A\lambda \frac{\partial T}{\partial x}\Big|_{x=0} \qquad \text{BC1} \qquad (2)$$

$$Q_b = -A\lambda \frac{\partial T}{\partial x}\Big|_{x=thick} \qquad \text{BC2} \qquad (3)$$

$$T(x,0) = 0 \qquad \text{IC} \qquad (4)$$

T is the temperature field. *Q* is heat flux across a boundary of area *A*. λ, ρ and c_p are material constants. PDE:s must be converted into ODE:s in order to be expressed with NMF. For the heat equation this means that we must discretize the equation in space but not in time, i.e., we must turn the equation into the following form,

$$\frac{\partial \boldsymbol{T}}{\partial t} = f(\boldsymbol{T}), \qquad (5)$$

where ***T*** is a vector of discrete temperatures across the wall. The space discretized heat equation in this form can be found in the EQUATION section of the NMF code in Figure 3. (The variables Taa and Tbb in the model are so called virtual temperatures that are used to create second order accurate boundary conditions. A more thorough discussion of the discretization process can be found in (Sahlin 1996a)).

Figure 3. NMF code for homogenous wall

To enable direct model translation to input-output oriented environments (e.g. TRNSYS, or HVACSIM+), variable declarations have a role attribute indicating IN for given variables and OUT for calculated ones.

Variables and parameters may be vectors or matrices. A parameter is anything that must remain constant throughout every simulation. Links may also be vectors, thus allowing models with variable number of ports. A special type of parameter, model parameters, governs vector and matrix dimensions. Regular and model parameters are divided into two categories, user supplied and computed; algorithmic computation of the latter category is described in the parameter processing section. Arbitrary foreign functions in FORTRAN 77 or C may be defined, either globally or locally within a model.

Special functions are defined to handle discontinuities, hysteresis, linearization, and errors. A more complete account of NMF is given in the NMF handbook (Sahlin 1996a) and reference report (Sahlin, Bring, and Sowell 1996).

The wall model in the example uses only equations. More general models may also have local assignment of variables and variables with an associated sample and hold function. The latter retain their value between timesteps. Both these variable categories are untouched by the global equation solving mechanism, since they receive their values through assignment directly in each model. Assigned variables are necessary to model certain effects, e.g. the behavior of a thermostat, and to provide sufficient "back doors" for non-standard models. We will refrain from a thorough discussion of these special variables here due to space.

TRANSFORMING EQUATIONS TO EXECUTABLE CODE

The problem of converting NMF models to executable code is, as we have said, handled differently for different MSEs. Many environments, such as IDA, SPARK, ESACAP and others, accept models directly as implicit equations, and the problem of generating code for these is consequently smaller. However, environments such as TRNSYS and HVACSIM+, require that component models are given in state space form, i.e.

$$d\boldsymbol{x}/dt = \boldsymbol{f}^x(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{u}, t, \boldsymbol{p}), \qquad (6a)$$

$$\boldsymbol{y} = \boldsymbol{f}^y(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{u}, t, \boldsymbol{p}), \qquad (6b)$$

where:

t	time, the independent variable.
$\boldsymbol{x}(t)$	variables appearing differentiated.
$\boldsymbol{y}(t)$	algebraic variables.
$\boldsymbol{u}(t)$	known functions of time.
$\boldsymbol{p}$	parameters, i.e., constant variables.
superscripts	function partitioning (not derivatives)

A general NMF model, on the other hand is a completely implicit system of DAEs:

$$0 = \boldsymbol{f}(d\boldsymbol{x}/dt, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{u}, t, \boldsymbol{p}), \qquad (7)$$

where $d\boldsymbol{f}/[d\boldsymbol{x}/dt; \boldsymbol{y}]$ is regular. The conversion between the implicit form (7) and the state space form (6a) and (6b) is nontrivial, and for the general case it cannot be done analytically. Instead the following partitioning is done,

$$0 = \boldsymbol{f}^c(d\boldsymbol{x}^i/dt, \boldsymbol{y}^i, \boldsymbol{x}, \boldsymbol{u}, t, \boldsymbol{p}), \text{ Cutset} \qquad (8a)$$

$$d\boldsymbol{x}^e/dt = \boldsymbol{f}^x(d\boldsymbol{x}^i/dt, \boldsymbol{y}^i, \boldsymbol{x}, \boldsymbol{u}, t, \boldsymbol{p}), \qquad (8b)$$

$$\boldsymbol{y}^e = \boldsymbol{f}^y(d\boldsymbol{x}^i/dt, \boldsymbol{y}^i, \boldsymbol{x}, \boldsymbol{u}, t, \boldsymbol{p}), \qquad (8c)$$

where $d\boldsymbol{f}^c/[d\boldsymbol{x}^i/dt; \boldsymbol{y}^i]$ is regular. The superscript *e* denotes the vector partitions that can be solved for explicitly by analytical methods. Equation (8a), sometimes referred to as the cutset, is solved numerically in each iteration by a call to a general purpose non-linear equation solver. Corresponding variables, with superscript *i*, are known as the cutset variables.

Finding a small cutset, for which $d\boldsymbol{f}^c/[d\boldsymbol{x}^i/dt; \boldsymbol{y}^i]$ remains regular in all cases, is key to the efficiency and robustness of generated code for environments such as TRNSYS and HVACSIM+. In practice, the issue is also complicated by the existence of subroutine calls, FOR loops, vectors, and assigned variables in the NMF code. In the following two sections, the algorithms used in the ASHRAE NMF Translator are presented in some detail.

Symbolic Solver

The symbolic solver can solve an equation with one unknown in two cases:

1. The unknown occurs only once in the equation and any subexpression containing the unknown is invertible.

2. The equation is linear with respect to the unknown or to an invertible expression of the unknown.

Inverses for the following functions are presently included in the symbolic solver: Sqrt, Log, Log10, Exp, Sin, ASin, Cos, ACos, Tan and ATan. In the source code version of the translator, it is straightforward to add new invertible functions.

Examples

$2\,x + b = 10$	- Ok
$c\,/\,x = 10$	- Ok
$a\,x = b$	- Ok if a is always nonzero (as indicated by the limits).
$2\,\exp(x) = a\,\exp(x) - b$	- Ok if never a=2 (as indicated by the limits of a).
$x = \exp(x)$	- unable to solve; x occurs twice.
$\sin(x) = a$	- Ok if the limits of x are inside [-π/2, π/2]
$4\,F(x) = c$	- unable to solve; cannot invert the user-defined function F.

Numerical solver

A public domain solver from the SLATEC library (SNSQ) with slight modifications is used. The solution algorithm is Powell's hybrid Newton-Raphson and gradient method, which has exhibited good performance on typical building simulation problems. It is, e.g., used in HVACSIM+ and TRNSYS 14.0. The control structure of the solver has been changed. The normal call of a user-supplied residual subroutine for equation evaluation has been replaced with a return to the calling routine (Figure 4). The value of a flag parameter tells the calling routine to re-calculate residuals and call the solver again (with unchanged flag). Other values of the flag tell that a solution has been found or signals an error. A code example of the call to the numerical solver can be seen in Figure 5.

With the present structure, all model equations are located in a single subroutine. This enhances readability. Had the cutset equations been placed in a separate subroutine, the numerical solver would still have to be altered, since we normally have several different cutsets in a system simulation. In its original form, SNSQ calls a single user-supplied subroutine with a fixed name.

Figure 4. Flowchart for numerical solver call structure

Cutset Algorithm

The symbolic solver tries to solve systems of equations (that can also include assignments) using the following algorithm:

- Divide the sequence of equations and assignments into subsequences of three classes:

  (A) containing only equations (including FOR loops with equations only);

  (B) containing only assignments;

  (C) compound statements containing both assignments and equations.

- Solve every subsequence separately.

  A. *Solving equations*

  1. Examine in turn the equations from the sequence and do the following:

     - If the equation has a single unknown, try to solve this equation.

       On success store the solution, remove the equation and the unknown and restart loop 1. If failed, continue loop 1 with the next equation.

     - If the equation has no unknowns, convert it to a residual assignment, store the obtained statement in the list of solutions, remove the equation, and continue loop 1 with the next equation.

  2. Select from the remaining set the equation with minimal number of unknowns and try to solve it for an arbitrary unknown.

     On success, add the remaining unknowns to the cutset, store the solutions.

On failure, try another unknown. If none are left, add all unknowns to variable cutset, convert equation to a residual assignment, store the obtained statement in the list of solutions.

Remove the equation and all encountered unknowns, restart loop 1.

Repeat A until the sequence of equations become empty.

B. Solving assignments

Add all unknowns encountered in the assignments to the cutset, store the statements in the list of solutions.

C. Solving compound statements.

Add all unknowns encountered in the assignments to the cutset, convert the equations to the assignment of the residuals, store the modified compound statement in the list of solutions.

- If the cutset is not empty, add call to numerical algorithm to solve the cutset variables iteratively.

Remarks

1. If equations and assignments are mixed, only the sets of consecutive equations (without assignments between them) can be reordered.
2. Arrays of equations (FOR loops) can be solved symbolically in two cases

   (a) generated equations are independent

   (b) the loop contains a single (or several independent) recursive equations

If the solver cannot solve a symbolic array of equations, it will add some of the unknowns to the cutset and then try to solve again.

Examples

```
FOR i=1 TO N
    i*x[i] = (N-i)*c[i]
END_FOR
```

can be solved for unknowns x[1] ... x[N]
(the equations with different `i` are independent)

```
FOR i=2 TO N
  i*x[i] = (N-i*1)*x[i-1]
END_FOR
```

can be solved for x[2] ... x[N]
(a system of recursive equations)

```
FOR i=1 TO N
  x[i] + 1/x[N-i+1] = i
END_FOR
```

the solver cannot find the symbolic solution
(the system is not recursive)

```
FOR i=1 TO N
  SUM j=1 TO N
    a[i,j] * x[j]
  END_SUM = b[i]
END_FOR
```

the solver cannot find the symbolical solution
(the system is not recursive)

```
FOR i=1 TO (N-1)
  y[i+1] + x[i] = i
  x[i+1] = 2*y[i]
END_FOR
```

Can be solved for unknowns x[1] ... x[N].
The unknowns y[1] ... y[N] will be added to the cutset. The equations x[i+1] = 2*y[i] (for i<=N-2) will be used to calculate the residuals.

3. The solver can only process arrays with simple indices: *k*, *i*, *i+k*, *N*, *N-k*, *N-i*, *N-i+k*, where *k* is an integer, *i* is a counter and *N* is an upper bound (limit) of the array. The limits of the iterator *i* must be as *k*, *N*, *N-k*. If an index of an unknown array is too complicated, the whole array will be added to the cutset (or the unknown part of the array).

4. If a FOR loop contains both equations and assignments, it will be split into several loops containing only equations or assignments.

Generated FORTRAN Code for Homogenous Wall Model

Figure 5 shows the generated TRNSYS FORTRAN code for the homogenous wall model (Figure 3). The code is structured in two subroutines, `TYPE 41` and `TQ_HOM_WALL`. The latter is called from the former and has mnemonic variable and parameter names to enhance readability. `TQ_HOM_WALL` calls in turn the subroutine `SOLVE`, indicating that the translator has been unable to find a zero cutset. The reason is that no lower bounds have been given for the computed parameters in the NMF code and there may be a risk of division by zero.

Figure 6 shows a version of `TQ_HOM_WALL` where the translator has been able to find a zero cutset. This code has been generated from a version of the NMF code where lower bounds for the computed parameters have been given. The same result can be obtained by selecting the Maximal rather than the default Safe option for the symbolic solver.

Another group of translator options governs the generation of code for checking of parameter and variable bounds. To save space (in this paper) no such code has been generated in the examples. Furthermore, in the generated code, the computed parameters are calculated for each evaluation of the subroutine. This is the default for simple parameter processing. When the processing section has calls to external subroutines, the computed parameters are stored rather than recalculated in each call.

Figure 5. Generated TRNSYS FORTRAN code for homogenous wall

Figure 6. Generated TRNSYS FORTRAN code for homogenous wall, no numerical solver call due to given lower bound (> 0) for computed parameters.

NMF RESOURCES

ASHRAE NMF Translator

The ASHRAE NMF Translator is the main deliverable of RP839, *Development of a Component Model Translator for the Neutral Model Format (NMF)* (Grozman and Sahlin 1996). It converts NMF files to FORTRAN code in the format required by TRNSYS (v. 13 and 14), HVACSIM+ and IDA Solver. The translator is delivered with a library of supporting FORTRAN routines, such as the modified SNSQ solver. Two translator editions are available, a 32 bit Windows version and a source code version.

The Windows version is equipped with a multiple document graphical user interface (Figure 7). The progress of the translation, and possible errors, is logged in a Message window. After completed translation, any errors reported can be located directly in the NMF source by double clicking on the line with the error report. Under Windows 95 or Windows NT, the program can also be executed in batch mode from a DOS prompt or batch file.

Figure 7. Graphical user interface of ASHRAE NMF Translator.

The source code version has been stripped of all Windows dependent features in order to run in a standard Common Lisp environment on any platform. It is interactively controlled by commands at the Lisp prompt. The source code version has been tested with GNU Common Lisp (GCL), which is available at no cost from the Open Software Foundation (OSF). GCL requires a C compiler for its installation. GNU C compilers (GCC) are also available from OSF for a variety of computer platforms. The translator testing has been done on a Sun SPARC station running Solaris 2.4.

Both translator versions are available from ASHRAE. The source code version may be used freely. It may, e.g., be shipped as a part of other products.

Other Tools and Models

Table 2 gives an overview of other available NMF resources. All of the listed items are accessible from the NMF home page (http://www.brisdata.se/nmf/). NMF users and developers are encouraged to submit relevant material to the maintainers of the home page.

Table 2. Available NMF Tools and Models

CONCLUDING REMARKS

Modular simulation environments in combination with ready-made component model libraries have the potential to drastically improve the quality and applicability of HVAC&R modelling. However, in order to attain this, models must be portable between environments. Portable models will (1) facilitate modelling projects that would otherwise be out of reach and (2) enable benchmarking exercises between simulation environments, and thereby improve their collective capability.

NMF provides a format for creation of portable model libraries. Several previous modelling projects have shown that NMF is capable of expressing full-complexity HVAC models. Large model libraries have already been constructed and made available on the Internet. The present work has shown that automatic translation, to input-output state-space oriented environments such as TRNSYS and HVACSIM+, is both possible and practical. An efficient NMF code development environment has been created and tested.

NMF and associated tools are now mature enough for wide usage. Remaining bottlenecks are not primarily technical but pedagogical. HVAC&R engineers, researchers and funders naturally focus more on the problem at hand than on the mathematical tools that are used to solve it. An example of this is that spreadsheet programs are used far beyond their capabilities. Powerful computer hardware is today accessible to all; the corresponding spread of modern simulation software has yet to materialize.

ACKNOWLEDGEMENTS

The development of the ASHRAE NMF Translator would not have been possible without the aid and support of the following people: Axel Bring, Alexander Shapovalov, Magnus Lindgren, Ed Sowell, Chip Barnaby, John Mitchell, Philip Haves and Les Norford.

REFERENCES

Bonneau, D., F.X. Rongere, D. Covalet, B. Gautier. 1993. CLIM 2000 - Modular Software for Energy Simulation in Buildings. *Conference proc. Building Simulation '93*. IBPSA, Adelaide, Australia, Aug. 1993.

Brandemuehl, M.J. 1992. *HVAC 2 TOOLKIT - A Toolkit for Secondary HVAC System Energy Calculation*. ASHRAE.

Buhl, W.F., E.F. Sowell, J.M. Nataf. 1989. Object Oriented Programming, Equation Based Submodels, and System Reduction in SPANK. *Building Simulation '89 Conference*. Vancouver, Canada, June, 1989.

Buhl, W.F., E. Erdem, F.C. Winkelmann, E.F. Sowell. 1993. Recent Improvements in SPARK: Strong Component Decomposition, Multivalued Objects, and Graphical Interface. *Conference proc. Building Simulation '93*. IBPSA, Adelaide, Australia, Aug. 1993.

Clark, D.R. 1985. *HVACSIM+ Building Systems and Equipment Simulation Program - Reference Manual*. National Institute of Standards and Technology, Washington D.C., 1985.

Clarke, J.A., D.F. Mac Randall. 1993. The Energy Kernel System: Form and Content. *Conference proc. Building Simulation '93*, IBPSA, Adelaide, Australia, Aug. 1993.

Elmqvist, H., F. Boudaud, J. Broenink, D. Brück, T. Ernst, P. Fritzson, A. Jeandel, K. Juslin, M. Klose, S. E. Mattsson, M. Otter, P. Sahlin, H. Tummescheit and H. Vangheluwe. 1997. *Modelica^{TM} - A Unified Object-Oriented Language for Physical Systems Modeling, Version 1*. To be published. (available at http://www.dynasim.se/Modelica).

Grozman, P., P. Sahlin. 1996. *ASHRAE RP-839 NMF Translator - User's Guide*. ASHRAE.

Hyttinen, J. and M. Vuolle. 1997. *http://www.hut.fi/~mvuolle/*

IEEE. 1997. Standard VHDL Analog and Mixed-Signal Extensions. Technical Report IEEE 1076.1, IEEE, March 1997.

Jeandel, A., F. Favret, E. Lariviere. 1993. ALLAN.Simulation - a General Software Tool for Model Description and Simulation. *Conference proc. Building Simulation '93*. IBPSA. Adelaide, Australia, Aug. 1993.

Klein, S. A., W. A. Beckman, J. A. Duffie. 1976. TRNSYS - a transient simulation program. *ASHRAE Trans*, 1976, Vol. 82, Pt. 2.

Kolsaker, K. 1990. DYNAMISK SIMULERING MED IDA - Et praktisk verktöj for bygningssimulering med modellbibliotek for fjernvarmeinstallasjoner. Technical report ISBN-82-595-6091-7, SINTEF, Trondheim, Norway.

Kolsaker, K. 1994. NEUTRAN - A Translator of Models from NMF into IDA and SPARK. *Proceeding of the BEPAC conference,* BEP'94, York, U.K.

Lorenz, F. 1990. Brief Description of the MS1 (Modelling System 1) Project, ACSC Consulting Company, Alleur, Belgium.

Nataf, J.-M. 1995. Translator from NMF to SPARK. *Conference proc. Building Simulation '95*. IBPSA, Madison, WI, USA, Aug. 1995.

Pelletret, R. 1994. Personal communication.

Sahlin, P. and E.F. Sowell. 1989. A Neutral Format for Building Simulation Models. *Proc. Building Simulation '89 Conference,* IBPSA, Vancouver, June 1989.

Sahlin, P., A. Bring, E.F. Sowell. 1992. The Neutral Model Format for Building Simulation. Swedish Institute of Applied Mathematics. report 1992:3. Stockholm, Sweden.

Sahlin, P. 1996a. NMF Handbook - an Introduction to the Neutral Model Format. Research Report. Dept. Building Sciences, KTH, Stockholm, Sweden, Feb., 1996. (available at ftp://urd.ce.kth.se/pub/reports/handbook.ps).

P. Sahlin. 1996b. *Modelling and Simulation Methods for Modular Continuous Systems in Buildings*, Doctoral Dissertation, Building Services Engineering, Bulletin No 39, KTH, Stockholm, May 1996.

Sahlin, P., A.Bring, E.F.Sowell. 1996b. The Neutral Model Format for Building Simulation, Version 3.02. Report. Dept. of Building Sciences, KTH, Stockholm, 1996 (available at ftp://urd.ce.kth.se/pub/reports/nmfre302.ps).

Sowell, E, F. and M. A. Moshier. 1995. HVAC Component Model Libraries for Equation Based Solvers. *Proc. Building Simulation '95 Conference, IBPSA*, Madison, WI, August 1995.

Stangerup, P. 1991. ESACAP: A Simulation Program and Description Language for Interdisciplinary Problems, *proc, 1991 European Simulation Multiconference*, Copenhagen, Denmark, June, 1991.

Yi Jiang , Zhenxi Xu, Feng Chen. 1994. TUHVAC - an object oriented computer software for HVAC design, analysis and simulation, *preprints of the 4th Intnl. Conf. on System Simulation in Buildings*, Dec, 1994, Liege, Belgium.

List of figures:

Figure 1. NMF code for global declarations

Figure 2. A homogeneous wall divided into several layers

Figure 3. NMF code for homogenous wall

Figure 4. Flowchart for numerical solver call structure

Figure 5. Generated TRNSYS FORTRAN code for homogenous wall

Figure 6. Generated TRNSYS FORTRAN code for homogenous wall, no numerical solver call due to given lower bound (> 0) for computed parameters.

Figure 7. Graphical user interface of ASHRAE NMF Translator.

List of tables

Table 1. A selection of Modular Simulation Environments

Table 2. Available NMF Tools and Models

QUANTITY_TYPES

```
/* type name      unit              kind */

   Area           "m2"              CROSS
   Control        "dimless"         CROSS
   HeatCap        "J/(K)"           CROSS
   HeatCapA       "J/(K m2)"        CROSS
   HeatCapM       "J/(kg K)"        CROSS
   HeatCond       "W/(K)"           THRU
   HeatFlux       "W"               THRU
   HeatFlux_k     "kW"              THRU
   Temp           "Deg-C"           CROSS
   Temp_F         "Deg-F"           CROSS
   Temp_K         "K"               CROSS
```

LINK_TYPES

```
/* type name       variable types... */

/* generic        (arbitrary, arbitrary,...) implicitly
defined */
   Q              (HeatFlux)
   T              (Temp)
   TQ             (Temp, HeatFlux)
   PMT            (Pressure, MassFlow, Temp)
   PMTQ           (Pressure, MassFlow, Temp, HeatFlux)
   MoistAir       (Pressure, MassFlow, Temp, HumRatio)
```

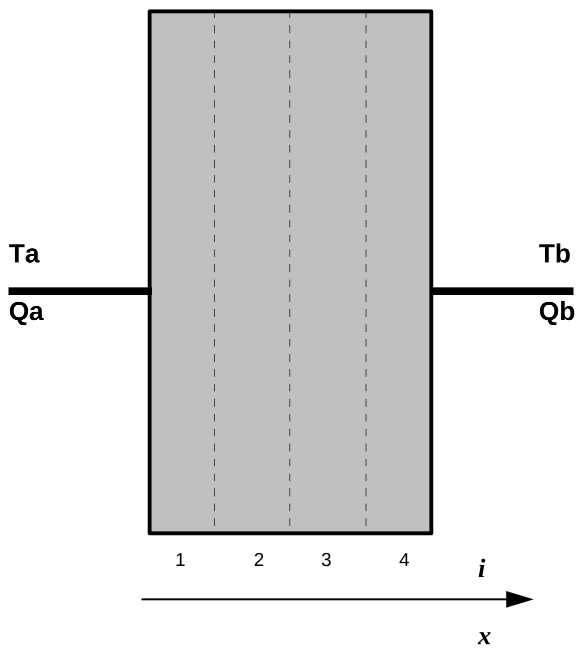


Figure 2. A homogeneous wall divided into several layers

```
CONTINUOUS_MODEL tq_hom_wall
ABSTRACT
   "1D finite difference wall model.
    One homogeneous layer.
    TQ interfaces on both sides."
EQUATIONS
/* space discretized heat equation */

   c_coeff * T'[1] = Taa - 2.*T[1] + T[2]; /*first cell*/
   c_coeff * T'[n] = T[n - 1] - 2. * T[n] + Tbb; /*last cell*/

   FOR i = 2, (n-1) /* loop over internal cells*/
     c_coeff * T'[i] = T[i - 1] - 2. * T[i] + T[i + 1];
   END_FOR ;

/* boundary equations */

   0 = -Ta + .5 * (Taa + T[1]);
   0 = -Tb + .5 * (T[n] + Tbb);
   0 = -Qa + d_coeff * (Taa - T[1]);
   0 = -Qb + d_coeff * (Tbb - T[n]);
LINKS
/*  type           name      variables ....  */

    TQ             a_side    Ta, POS_IN Qa ;
    TQ             b_side    Tb, POS_IN Qb ;
VARIABLES
/* type    name   role  def  min       max    description*/

   Temp     T[n]  OUT   0.   abs_zero  BIG    "temperature profile"
   Temp     Ta    OUT   0.   abs_zero  BIG    "a-side surface temp"
   Temp     Tb    OUT   0.   abs_zero  BIG    "b-side surface temp"
   Temp     Taa   OUT   0.   abs_zero  BIG    "a-side virtual temp"
   Temp     Tbb   OUT   0.   abs_zero  BIG    "b-side virtual temp"
   HeatFlux Qa    IN    0.   -BIG      BIG    "a-side entering heat"
   HeatFlux Qb    IN    0.   -BIG      BIG    "b-side entering heat"
MODEL_PARAMETERS
/* type      name    role  def  min       max    description  */
   INT       n       SMP   3    3         BIGINT "no of temp layers"
PARAMETERS
/* type      name     role def   min   max  description  */

/* easy access parameters */
   Area      a        S_P  10.   SMALL BIG  "wall area"
   Length    thick    S_P  .25   SMALL BIG  "wall total thickness"
   HeatCondL lambda   S_P  0.5   SMALL BIG  "heat transfer coeff"
   Density   rho      S_P  2000. SMALL BIG  "wall density"
   HeatCapM  cp       S_P  900.  SMALL BIG  "wall heat capacity"
/* derived parameters */
   generic   d_coeff  C_P                   "lambda*a/dx"
   Length    dx       C_P                   "layer thickness"
   generic   c_coeff  C_P                   "rho*cp*dx*dx/lambda"
PARAMETER_PROCESSING
dx := thick / n ;
    c_coeff := rho * cp * dx * dx / lambda;
    d_coeff := lambda * a / dx ;

END_MODEL
```

Figure 3. NMF code for homogenous wall

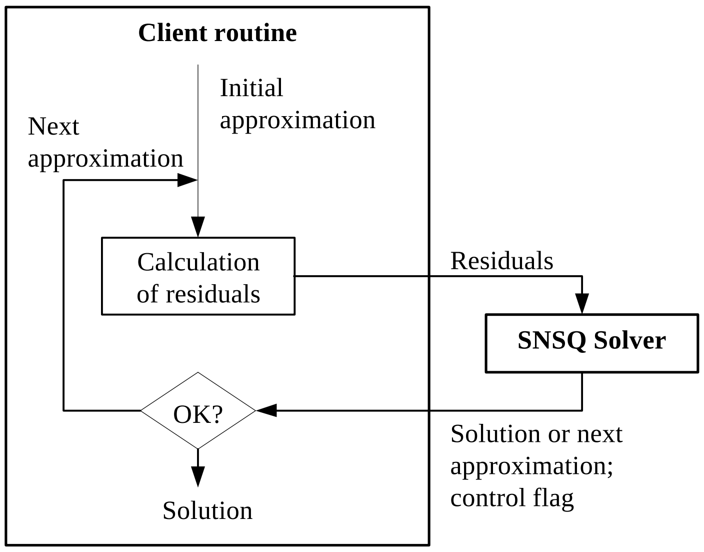


Figure 4. Flowchart for numerical solver call structure

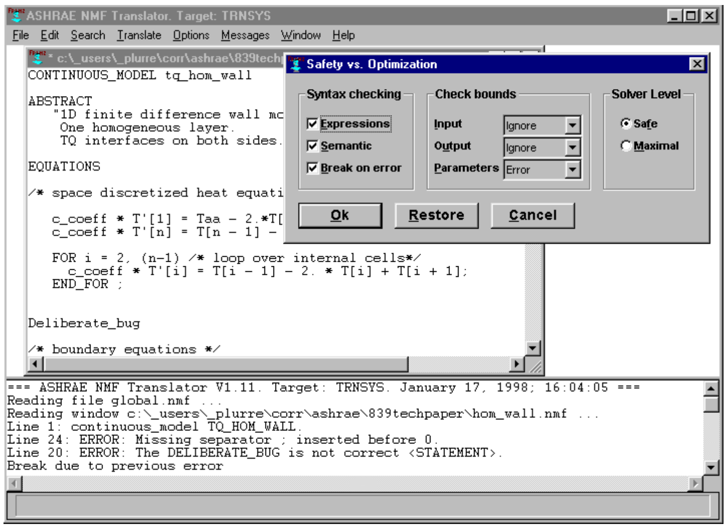
```
* #==================================#
* # TYPE 41  TQ_HOM_WALL             #
* # -------------------------------- #
* # 1D finite difference wall model. #
* # One homogeneous layer.           #
* # TQ interfaces on both sides.     #
* # -------------------------------- #
* # MODEL PARAMETER                  #
* #  1  N  no of temp layers         #
* # PARAMETERS                       #
* #  2  A       wall area            #
* #  3  THICK   wall total thickness #
* #  4  LAMBDA  heat transfer coeff  #
* #  5  RHO     wall density         #
* #  6  CP      wall heat capacity   #
* # INPUT VARIABLES                  #
* #  1  QA  a-side entering heat     #
* #  2  QB  b-side entering heat     #
* # OUTPUT VARIABLES                 #
* #  1    T(N)  temperature profile  #
* #  1+N  TA    a-side surface temp  #
* #  2+N  TB    b-side surface temp  #
* #  3+N  TAA   a-side virtual temp  #
* #  4+N  TBB   b-side virtual temp  #
* # DERIVATIVES                      #
* #  1  DT_T(N)  Derivative of T     #
* # COMPUTED PARAMETERS              #
* #  D_COEFF  lambda*a/dx            #
* #  DX       layer thickness        #
* #  C_COEFF  rho*cp*dx*dx/lambda    #
* # CONSTANT                         #
* #  ABS_ZERO  -273.16  Deg-C        #
* # -------------------------------- #
* # TOTAL NUMBER OF                  #
* #   PARAMETERS        6            #
* #   INPUT VARIABLES   2            #
* #   OUTPUT VARIABLES  4+N          #
* #   DERIVATIVES       N            #
* #==================================#
*
      SUBROUTINE TYPE 41
     &     (TIME, XIN_, OUT_, T_, DTDT_, PAR_,
INFO_, ICNTRL_, *)
      INTEGER INFO_(15), ICNTRL_(*)
      REAL TIME, T_(*), DTDT_(*), PAR_(*)
      DOUBLE PRECISION XIN_(*), OUT_(*)
      COMMON /STORE/ NSTORE_, IAV_, S_
      INTEGER NSTORE_, IAV_, IPOS_, LPOS_
      REAL S_(5000)
      SAVE LPOS_
      COMMON /HEAP/ HEAP_
      REAL HEAP_(1000)
      INTEGER N
      INTEGER I_
      N = PAR_(1)
      NVAR_ = 2+N
* First call of the unit
      IF (INFO_(7) .LT. 0) THEN
         INFO_(6) = 4+N
         IF (6+(17+NVAR_)*NVAR_+N .GT. 2000)
     &          THEN
            PRINT *, 'Out of HEAP Block'
            CALL TYPECK(-2, INFO_, 0, 0, 0)
         END IF
         INFO_(9) = 0
         IPOS_ = IGET_STORE(INFO_, (1+NVAR_)*
     &         NVAR_, LPOS_, 2, 6, N)
      ELSE
         IPOS_ = IFIND_STORE(INFO_(1), LPOS_)
      END IF
      CALL STORE_INFO(INFO_)
* Accept the results of integration
      CALL COPY_ARRAY(HEAP_(3), T_(1), N)
* Call the model subroutine
      CALL COPY_FDARRAY(HEAP_(1), XIN_(1), 2)
      CALL TQ_HOM_WALL (N, PAR_(2),
     &      PAR_(3), PAR_(4), PAR_(5),
     &      PAR_(6), HEAP_(1), HEAP_(2),
     &      HEAP_(3), HEAP_(3+N),
     &      HEAP_(4+N), HEAP_(5+N),
     &      HEAP_(6+N), DTDT_(1), NVAR_,
     &      S_(IPOS_), HEAP_(7+N))
* Convert TIME to hours
      DO 101 I_ = 1, N
         DTDT_(I_) = 3600*DTDT_(I_)
 101     CONTINUE
* Save the output as REAL*8
      CALL COPY_DFARRAY(OUT_(1), HEAP_(3), 4+
     &      N)
      RETURN 1
      END


      SUBROUTINE TQ_HOM_WALL (N, A, THICK,
     &      LAMBDA, RHO, CP, QA, QB, T, TA,
     &      TB, TAA, TBB, DT_T, NVAR_,
     &      XVAR_, FVEC_)
      INTEGER N
      REAL A, THICK, LAMBDA, RHO, CP,
     &      D_COEFF, DX, C_COEFF, T(N), TA,
     &      TB, TAA, TBB, QA, QB, DT_T(N)
      REAL XVAR_(*), FVEC_(*)
      COMMON /INFORM/ INFO_
      INTEGER INFO_(10)
      REAL ABS_ZERO
      PARAMETER (ABS_ZERO=-273.16)
* Counters and auxiliary identifiers
      INTEGER NFLAG_, I
* Parameter processing
      DX = THICK/REAL(N)
      C_COEFF = RHO*CP*DX*DX/LAMBDA
      D_COEFF = LAMBDA*A/DX
* Initial values for numerical solver
      IF (INFO_(7) .LT. 0) THEN
         CALL SET_ARRAY(XVAR_(1), 0.0, 2+N)
      END IF
      NFLAG_ = 0
* Loop for numerical solving
 300     CONTINUE
* The values of the cutset variables
         CALL COPY_ARRAY(DT_T(1), XVAR_(1), N
     &         )
         I_0 = 1+N
         TAA = XVAR_(I_0)
         TBB = XVAR_(1+I_0)
         IF (NFLAG_ .EQ. 1) GOTO 701
* Calculate the residuals for numerical solver
         DO 102 I = 2, (N-1)
            FVEC_(-1+I) = -C_COEFF*DT_T(I)+T(
     &            I-1)-2.0*T(I)+T(I+1)
 102        CONTINUE
         FVEC_(-1+N) = -QA+D_COEFF*(TAA-T(1))
         TA = 0.5*(TAA+T(1))
         FVEC_(N) = -C_COEFF*DT_T(1)+TAA-2.0*
     &         T(1)+T(2)
         FVEC_(1+N) = -QB+D_COEFF*(TBB-T(N))
         TB = 0.5*(TBB+T(N))
         FVEC_(2+N) = -C_COEFF*DT_T(N)+T(N-1)
     &         -2.0*T(N)+TBB
* Numerical solver
         CALL SOLVE(NVAR_, XVAR_, FVEC_,
NFLAG_)
         IF (NFLAG_ .EQ. 1 .OR. NFLAG_ .GT. 10)
GO TO 300
         PRINT *,
     &         'Unable to solve the equation'
         CALL TYPECK(-2, INFO_, 0, 0, 0)
 701  CONTINUE
      RETURN
      END
```

Figure 5. Generated TRNSYS FORTRAN code for homogenous wall

```
SUBROUTINE TQ_HOM_WALL (N, A, THICK,
     &      LAMBDA, RHO, CP, QA, QB, T, TA,
     &      TB, TAA, TBB, DT_T)
      INTEGER N
      REAL A, THICK, LAMBDA, RHO, CP,
     &      D_COEFF, DX, C_COEFF, T(N), TA,
     &      TB, TAA, TBB, QA, QB, DT_T(N)
      COMMON /INFORM/ INFO_
      INTEGER INFO_(10)
      REAL ABS_ZERO
      PARAMETER (ABS_ZERO=-273.16)
* Counters and auxiliary identifiers
      INTEGER I
* Parameter processing
      DX = THICK/REAL(N)
      C_COEFF = RHO*CP*DX*DX/LAMBDA
      D_COEFF = LAMBDA*A/DX
* Calculations
      DO 102 I = 2, (N-1)
         DT_T(I) = (-2.0*T(I)+T(I+1)+T(I-1))/
     &         C_COEFF
 102     CONTINUE
      TAA = QA/D_COEFF+T(1)
      DT_T(1) = (TAA+T(2)-2.0*T(1))/C_COEFF
      TA = 0.5*(TAA+T(1))
      TBB = QB/D_COEFF+T(N)
      DT_T(N) = (TBB-2.0*T(N)+T(N-1))/C_COEFF
      TB = 0.5*(TBB+T(N))
      RETURN
      END
```

Figure 6. Generated TRNSYS FORTRAN code for homogenous wall, no numerical solver call due to given lower bound (> 0) for computed parameters.


ASHRAE NMF Translator. Target: TRNSYS
File Edit Search Translate Options Messages Window Help
* c:_users_plurre\corr\ashrae\839tech
Safety vs. Optimization
Ok
Restore
Cancel

```
CONTINUOUS_MODEL tq_hom_wall

ABSTRACT
   "1D finite difference wall mc
    One homogeneous layer.
    TQ interfaces on both sides.

EQUATIONS

/* space discretized heat equati

   c_coeff * T'[1] = Taa - 2.*T[
   c_coeff * T'[n] = T[n - 1] -

   FOR i = 2, (n-1) /* loop over internal cells*/
     c_coeff * T'[i] = T[i - 1] - 2. * T[i] + T[i + 1];
   END_FOR ;


Deliberate_bug

/* boundary equations */
```

Syntax checking: [✓ Expressions] [✓ Semantic] [✓ Break on error]

Check bounds: Input: Ignore; Output: Ignore; Parameters: Error

Solver Level: [✓ Safe] Maximal

```
=== ASHRAE NMF Translator V1.11. Target: TRNSYS. January 17, 1998; 16:04:05 ===
Reading file global.nmf ...
Reading window c:\_users\_plurre\corr\ashrae\839techpaper\hom_wall.nmf ...
Line 1: continuous_model TQ_HOM_WALL.
Line 24: ERROR: Missing separator ; inserted before 0.
Line 20: ERROR: The DELIBERATE_BUG is not correct <STATEMENT>.
Break due to previous error
```

Figure 7. Graphical user interface of ASHRAE NMF Translator.

Table 1. A selection of Modular Simulation Environments

TRNSYS	was developed during the early seventies at the Solar Energy Lab at the University of Wisconsin. The primary application then was solar energy systems. Several compatible modelling tools have been developed independently, e.g., PRESIM and IISiBat. A fundamental limitation with TRNSYS is the fixed timestep. A simultaneous solver is, since version 14, offered to provide reasonable robustness for many problem types, but to attain acceptable efficiency with such a solver, the possibility to take long timesteps must be utilized when little is going on in the simulated system.	(Klein, Beckman and Duffie 1976) (http://sel.me.wisc. edu/trnsys/)
IDA	is a graphical modelling environment (IDA Modeller), solver (IDA Solver) and NMF translator (IDA NMF Translator). IDA is the work of, among others, the authors of this paper. IDA Solver is a variable timestep and order, input-output free DAE solver with a large selection of numerical methods. IDA Modeller is used for graphical manipulation of large system models. It is also a development environment for end-user applications.	(Sahlin 1996b) Bris Data AB, Stockholm, Sweden. (http://www. brisdata.se)
HVACSIM+	is a solver with similar characteristics as TRNSYS in terms of model format and structure, but more recent numerical techniques than in the original TRNSYS are utilized. It was developed by NIST in Maryland and released in the mid eighties on a Public Domain basis. Unfortunately, the support level is currently very low.	(Clark 1985)
ALLAN. Simulation	is a graphical modeller and solver combination developed by Gaz de France. It was internally released in the mid eighties and have been extensively applied to building simulation.	(Jeandel, Favret and Lariviere 1993).
ESACAP	is an electrical circuit simulator which has been generalized into a full DAE solver. The development of the ESACAP language and solver was initiated in the late seventies at Elektronik Centralen in Denmark. Two prototype NMF translators have been developed (Pelletret 1994), (Lorenz 1994).	(Stangerup 1991) STANSIM, Denmark. (http://www.it.dtu.dk /~el/ecs/esacap.htm)
CLIM 2000	a graphical modelling tool for building applications, has been developed by Electricité de France for internal use. CLIM 2000 has recently changed to the ESACAP solver.	(Bonneau et al. 1993).
MS1	is a graphical multi input language modeller with interfaces to several solvers by Lorenz Simulation in cooperation with Electricité de France. MS1 models can be described with NMF, Bond Graphs, Linear Graphs, and Block Diagrams.	(Lorenz 1990) Lorenz Simulation, Liege, Belgium (http://www.lorsim. be)
THUVAC	a graphical modelling tool and solver for simulation of HVAC related modular systems. It is under development at the Dept. of Thermal Energy, Tsinghua Univ., Beijing, China.	(Yi Jiang et al. 1994)
SPARK	SPARK is a DAE solver under development at Lawrence Berkeley Laboratory, California. SPARK is intended to be a modular complement to the widely used monolithic DOE-2 building energy simulation tool. A library of native models for building secondary mechanical systems is available, as is a prototype NMF translator (Nataf 1995). A graphical model editor has also been developed.	(Buhl et al.1993) (http://eande.lbl.gov/ btp/SPARK.html)
EKS	is a C++ toolkit for development of energy related simulation design tools, by among others the Univ. of Strathclyde, Scotland	(Clarke et al. 1993)

Table 2. Available NMF Tools and Models

IDA NMF Translator	An NMF Translator which is based on the Windows version of the ASHRAE NMF Translator. This version is continually maintained and enhanced. New features include symbolic calculation of Jacobian matrices for IDA Solver, project management, and code generation for TRNSYS 14.2. Platform: Windows 95 or NT. Available without cost.
NMF Handbook and Reference Report	NMF Documentation. The handbook (Sahlin 1996a) gives in-depth advice about NMF usage with many examples. The reference manual (Sahlin, Bring and Sowell 1996) specifies the grammar in detail. Delivered with the ASHRAE and IDA NMF Translators.
NMF to HTML generator	Automatic generation of HTML presentations of NMF models. Links to main code sections. Links between variable usage and declaration. Color coded comments and keywords. Automatic generation of list of all models in library with revision times. Available without cost.
NMF Home Page	A directory for all NMF related tools, simulation environments and model libraries. (http://www.brisdata.se /nmf/)
Simulation Model Network (SIMONE)	A global a web based network of NMF model libraries and developers. Libraries are maintained by individual developers but adhere to a common presentation format. Most libraries are accessible for immediate review, search and download; others are supported commercial products. Developers are encouraged to submit also libraries which are under construction or no longer regularly maintained. Library status is indicated by keywords.
IEA Task 22 - Building Indoor Climate and Energy	NMF models for indoor climate and energy studies in buildings. Detailed and simplified zone models. Simplified models of primary and secondary HVAC equipment. Detailed zone includes thermal straticfication, full non-linear long wave radiation, operating temperature and PPD comfort measures. Bi-directional air flow links for natural ventilation studies. Interchangeable controllers for air and water streams. Reduced and full FD wall models. Detailed external shading algorithms. Available without cost.
ASHRAE HVAC 2 - Secondary HVAC Systems	NMF component models for use in predicting the energy performance of secondary HVAC equipment. Models have been selected from the literature and include computational algorithms for secondary HVAC system equipment, selected properties for working fluids and utility components and controls for modeling a complete system. Static models. Pressure losses in the air part of the system have been added in the NMF conversion process. Available without cost.
Multizone Air Exchange	Bi-directional flow of tempered and contaminated air between zones and in the ventilation system of a building. Static models. Full pressure losses. Available without cost.
IEA Annex 17 - Primary and Secondary HVAC Systems	Air and water transport in the primary and secondary parts of a ventilation system. Ported from TRNSYS types developed in IEA Annex 10 and 17. Static models. No pressure losses. Available without cost.