APPLICATION OF THE SPARK ENVIRONMENT TO 3D AIR FLOW PROBLEMS

Jean-Michel Nataf and Etienne Wurtz

Groupe Informatique et Systèmes Énergétiques*

Abstract

The SPARK simulation environment is an object based simulation environment. Its objects are equations or systems of equations. Creating SPARK objects requires from the user to write SPARK syntax and C code. Hooking objects together requires from the user to specify their common variables. Then SPARK creates a C program that solves for the specified problem.

The first task of creating SPARK objects and C code is automated by use of an interface written in MACSYMA or MAPLE, both well known computer algebra languages. The second task of generating a global simulation specification hooking these objects together is automated by a tool written in C that creates SPARK syntax -hooking together identical cells of arbitrary complexity, taking into count the way the cells' interface have to be connected.

These tools are applied to the problem of 3-D air flow inside a zone, itself subdivided in subzones where mass and energy balances are performed, and between which air flow equations are implemented, as proposed by other authors. Stack effect is taken into count, as well as the perfect gas law. The problem is nonlinear and hard to solve, due to the square root behavior of the mass flow equations between zones.

The implementation of this known nonlinear model into SPARK is fully automated. Results and performance are presented along with the precautions to be taken while specifying the equations (since the solution technique is determined by SPARK itself unless the user constrains the solution techniques by forbidding certain operations). In our case, we order the problem so that the only iteration variables chosen by SPARK are pressures and temperatures, thus avoiding the densities.Various cases are examined. Partition of the zone into 8, 27 and 64 subzones is automatically modeled and simulated. This kind of partitioning, halfway between global lumped formulations and finite differences approaches, combines accur acy with computational tractability.

The advantages of object based approaches to this particular problem are then discussed. Code reusability, ease of modification and full modularity are emphasized.

1 INTRODUCTION

In recent years, code modularity and reusability have been at the forefront of the trends in simulation in buildings. The reason for this evolution was the increasing complexity and intractability of the integrated simulation programs available to the modeler. Although very complete, these programs were and remain hard to modify, correct and maintain.

The answer to that problem was the use of the Object Oriented Paradigm to design new simulation environments. Several so-called object-oriented environments arose (IDA, as described in (Bring, 1992), ZOOM as described in (Joly *et al.*, 1989), CLIM2000 as described in (Bonneau *et al.*, 1989), SPARK as described in (Buhl *et al.*, August 30, 1990)). According to its definition, Object Oriented Programming provides model encapsulation in objects, that can inherit properties among each other and communicate through messages. More practically, this means that model details are localized in the code, and that the user does not have, when designing a model, to worry about the implications of the local model outside of it: thus is the problem of

*GISE-ENPC, La Courtine Cedex, F-93167, Noisy Le Grand, France. Tel:(1)43 04 40 98 X3492, FAX:(1) 43 04 63 64 tentacular code eliminated.

For independent reasons, interest in domain partitioning has risen in the air flow community. Convection in buildings is a very complex and calculations intensive problem, and the need to resort to semi-lumped formulations has started to be felt. This approach is half way between the fully detailed finite difference or finite elements approach (where the domain is finely discretized), and the general aggregate formulation, where only global balance equations are used.

Therefore it is a natural evolution that object-based programming and air flow simulation should meet. This paper presents an implementation of domain partitioned air flow simulation, along with a description of the problems encountered and elaborations upon the results.

2 AGGREGATE FORMULATION OF AIR FLOW MODEL

The problem considered is air flow through subzones of an air zone. It amounts to partition the zone into several subzones called cells, and link these cells together by air flow models.

Previous work in this field is exemplified by (Inard, 1988) or (Inard & Buty, 1991), although the application and the modalities were different: a heat source was present, and the cells were not necessarily rectangular. The model we use is the one presented in (Bouia & Dalicieux, 1991). Only it is generalized to the three-dimensional case, instead of the 2D approach of the cited paper.

The model equations are as follows:

Mass Balance:
$$\sum q_m - q_{source} + q_{sink} = 0$$

Heat Flux Balance:
$$\sum \phi - \phi_{source} + \phi_{sink} = 0$$

Horizontal Fluxes:
$$\phi_{horiz} = q_{m_s} c_p T_s - q_{m_e} c_p T_e$$

Vertical Fluxes:
$$\phi_{vert} = q_{m_{vert}} c_p T_{vert}$$

Neutral Point:
$$z_n = \frac{\Delta P_0}{\Delta \rho g}$$

Perfect Gas Law:
$$P = \rho \frac{R}{M} T$$

Pressure:
$$P = P_0 - \rho g z$$

Horizontal Mass Flow:
$$q_{m_{vert}} = k \rho S (P - P_{top})^n$$

Vertical Mass Flow:
$$\Delta P = \Delta P_0 - \Delta \rho g z = -\Delta \rho g (z - z_n)$$

Integrated Upper Horizontal Mass Flows:
$$q_{m_s} = k l \rho (\Delta \rho g)^n \frac{\bar{h} - h - z_n{}^{n+1}}{n+1}$$

Integrated Lower Horizontal Mass Flows:
$$q_{m_e} = k l \rho (\Delta \rho g)^n \frac{z_n{}^{n+1}}{n+1}$$

Integrated Horizontal Mass Flows:
$$q_m = q_{m_s} - q_{m_e}$$

Balance equations are classical, and are enacted on each cell. Similarly, there is one perfect gas law equation per cell. The flow equations are enacted at each frontier between cells, and at the frontier of the overall domain. The neutral point equation exists only on vertical frontiers.

The parameters are chosen as in (Bouia & Dalicieux, 1991). In particular, the exponent n that determines the air flow between cells ranges between 0.5 for the large openings to 0.67 for the closed doors and windows to 1. for permeable walls.

3 THE SPARK OBJECT-BASED ENVIRONMENT

SPARK is an environment, developed at Lawrence Berkeley Laboratory, that is object-based, and allows the user to hook together components he defined, without having to write a simulation program for the hooked system: SPARK takes in the structure and writes a C program solving for the problem in a dedicated and efficient way.

We present here a short outline of the SPARK environment. More complete descriptions of SPARK and its applications can be found in (Anderson, September 1986), (Sowell *et al.*, 1986), (Sowell & Buhl, 1988), (Sowell *et al.*, June 1989), (Buhl *et al.*, August 30, 1990).

SPARK is an object-based environment. The elementary objects for SPARK are equations. These equations can be piecewise defined, and can involve nonalgebraic terms (like exponentials, logarithms,etc.). The SPARK object specifies which variables the equation can be explicitly solved for, and what user-supplied C function is to be invoked to obtain these variables. Modifying a model in SPARK is usually modifying an equation, and in SPARK the modification takes place only in the relevant object and its attached functions, and nowhere else. Thus is model encapsulation supported in SPARK. An object contains all the methods (functions) that allow to solve for each of its variables in terms of the associated equation, and its connections to the outside worlds are its variables.

As objects, they communicate towards the outside through their variables and parameters. This is the -very limited - way SPARK supports message passing.

Several equations can be grouped together to form a macro object. It suffices to declare in the body of the macro the instances of objets it is composed of. A macro object is therefore a set of equations, and its interface to the outside is the union of the interfaces of its elementary components. That is the -again very limited - way SPARK supports inheritance. Practically, however, it definitely allows to reuse a simple model and complexify it to obtain a more complete model.

A very important case of macro is the ordinary differential equation object, that groups together the ODE itself and an integrator, that derives the dynamic variable from its derivative. Integrators are SPARK object as well, and the modeler/user can design his/her own. It has to be noted that an ordinary differential equation integrated by a simple integrator contains two elementary objets, while resorting to a predictor corrector method leads to three elementary objects, since the predictor-corrector method involves two equations. Last, it is possible In SPARK to merge the ODE with its integrator, in particular when one uses the Runge Kutta method.

Once these objects are available, the modeler links them together in a specification file, specifies there the ones that are unknown and the ones that are inputs, and supplies an input file containing the numerical values of the inputs. It has to be noticed that the input-outputs are specified only at linking time. The objects, themselves, are essentially undirected and have no predefined input or output. They can be used in any way. That is therefore very different from the "modular subroutines" approach, where the model is indeed encapsulated in a routine, but the routine has predefined inputs and outputs. In SPARK, the only thing that counts is the equation. The user can choose whatever input and unknowns he/she wants, provided this leads to a well defined problem.

The originality of SPARK is that it generates, through graph-theoretical means, a "small" set of unknowns (called the "cut set") such that it is possible to derive explicitly from these unknowns all the other unknowns of the problem. The ratio of the overall number of unknowns to the cardinal of the cut set is called the reduction. The interest of the approach is that the system of equations effectively solved is the reduced system of equation, the remaining unknowns being obtained by mere substitution. Nonlinear system resolution usually involves inversion of a Jacobian matrix of size the size of the system. Thus reducing the system yields a r^3 acceleration of the process, r being the reduction.

Several tools are supplied to ease even further the task of the modeler. We have seen that SPARK generates the global simulation program automatically. SPARK also has a symbolic interface that allows the user to enter equations in symbolic form, and the SPARK object and its attached functions is automatically generated. Thus the task of writing C code or SPARK object syntax is eliminated.

4 IMPLEMENTATION OF THE MODEL UNDER SPARK

SPARK is equation-based: an elementary object is an equation. Thus all the above-mentioned equations will appear as elementary objects. The only remaining ambiguity is how to create macro objects, i.e. how to group together elementary objects.

Two approaches are possible and are described below.

4.1 Homogeneous formulation

One approach is to link together identical "cell" objects, the cell containing both the space of the zone subdomain, and the flow equations between subdomains. Thus, such a cell contains, at local level, all the equations presented in the preceding section.

One immediate drawback of this approach is that all frontier equations are calculated twice in the overall simulation. And all cells are forced to ramify into neighboring cells, so as to know their values of state variables, that are needed to calculate flows (based on specific mass difference). The only interest is the bigger ease of the automatic generation of the connection file. It is also interesting as a comparison point to alternate approaches.

4.2 Cells and frontiers

A better approach, more natural, is to create two types of objects: one cell object with the state equation and the balance equations, and several frontier objects containing the flow mass equations and the neutral point equations.

These objects were created automatically by the symbolic interface to SPARK, written in the MACSYMA computer algebra language. More details can be found in (Sowell *et al.*, January 1990) or (Nataf & Winkelmann, 1992). It suffices to say here that the user enters the equations in symbolic form, specifies the name of the object he wants to see created, and the invokation of a suitable function generates automatically the appropriate SPARK object and all its attached C functions.

The global connection file, where the objects are declared and linked together, as well as the input file template, are created automatically by a general specification file generator able to create homogeneous simulations in SPARK as well as "checkered" simulations, in which the objects of one type are linked to their neighbors through another type of objects.

Thus the typing effort of the modeler, for this problem, reduces to the specification of the links between only *one* pair of neighbors, and the equations of the elementary objects. The size of the problem is a parameter. Thus creating a 4x4x4 simulation when one already took the trouble to create a 2x2x2 simulation takes three keystrokes or so -just change 2 2 2 into 4 4 4. Then the rest of the process is entirely automated.

5 RESULTS

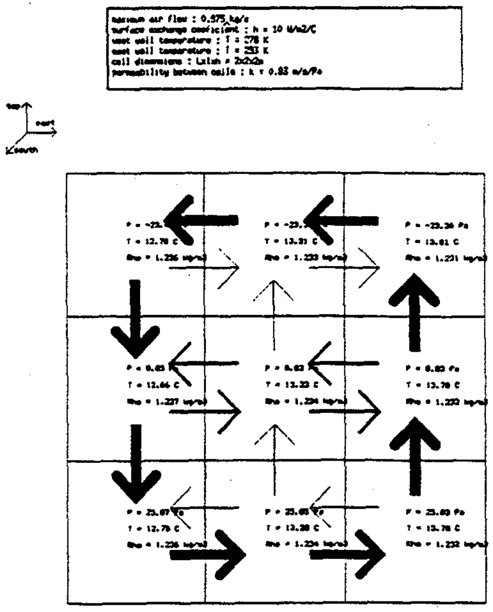
The problem considered was a standard hot and cold wall cubic zone. Runs were performed for a partitioning of this zone in 8, 27 or 64 cells.

For the steady state 8 cells simulation (2x2x2), the SPARK solver ends up with 16 iteration variables, as is natural (for example, two independent state variables per cell): these iteration variables are the pressure and the temperature. This result holds with both the homogeneous approach (all cells identical, one type of object only) and the cell and frontier approach. This results is comforting, in the sense that even in the case of the first approach when there are more equations than needed, the reduction algorithm in SPARK manages to reduce the iterative problem to the size reached from a more efficient approach.

For the steady state 27 cells simulation (3x3x3), the SPARK solver ends up with 54 iteration variables, the middlepoint pressures and the temperatures of all cells. Care must be exercised in the choice of the ordering of the statements in the specification file, however. In the converging case, the ordering is (from beginning to end) the middle point pressures, the vertical mass flows, the top pressures, the densities,... and at the end the temperatures. Now if one had chosen to put the middlepoint pressures that before the temperatures at the end of the specification file, then one would have obtained 71 iteration variables, among which several vertical mass flows, a heat flux and a density; worse, convergence is not attained in that case. That sensitivity of the SPARK solver is due to the two stage process of matching equations with variables first, then reducing the system: if the initial matching (that is ultimately sensitive to the statement ordering) is inappropriate, then the cut set algorithm fares very poorly.

A graphical interface has been developed for a better understanding of the results. The flow intensity is proportional to the thickness of the arrows. In all cases we present, the excitations are identical: walls south and east are cold, north and west are hot, whereas top and bottom temperatures are set to intermediate values. A window is set on the east face and a door on the west face.

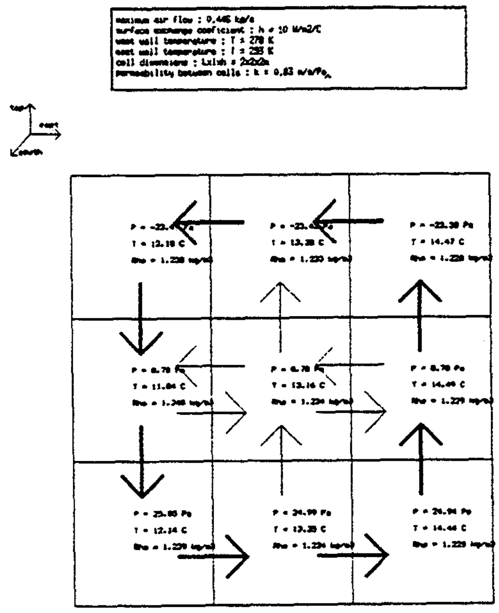
If we cut a cubic volume divided into 3x3x3 cells in the middle (east and west oriented) we obtain the following results:



maximum air flow : 0.575 kg/s
surface exchange coefficient : h = 10 W/m2/C
west wall temperature : T = 278 K
east wall temperature : T = 293 K
cell dimensions : LxlxH = 2x2x2m
permeability between cells : k = 0.83 m/s/Pa
top
east
south

We see that air circulation is consistent with the temperature conditions on the faces of the cubes. The temperature level increases with volume height. And there is lower outgoing and upper incoming air flow.

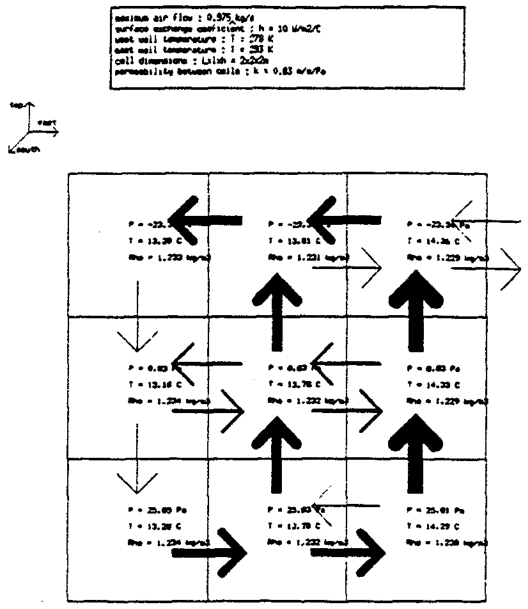
Dependency on n is then investigated: we compare this results with a fictitious case where n is set to 1 in the flow equation $\dot{m} = k * \Delta P^n$ using the same scale for flow representation with the aim of testing non-linearity influence:



maximum air flow : 0.446 kg/s
surface exchange coefficient : h = 10 W/m2/C
west wall temperature : T = 278 K
east wall temperature : T = 293 K
cell dimensions : LxlxH = 2x2x2m
permeability between cells : k = 0.83 m/s/Pa
top
east
south

The flow between interior cells is reduced by half and temperatures are more heterogeneous in this fictitious case. The influence of the value of n cannot be neglected.

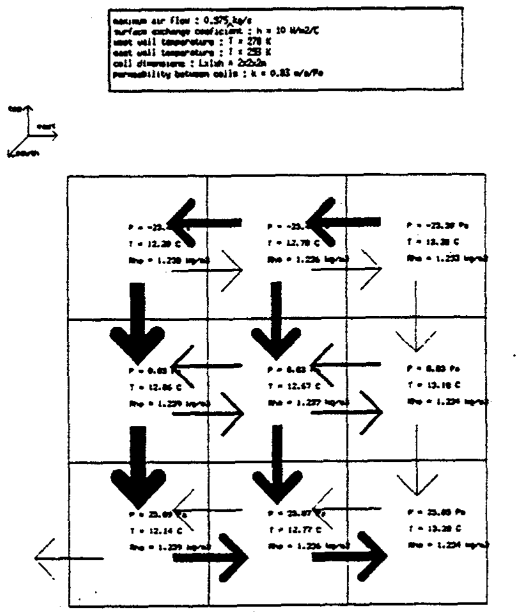
If we consider a new cross section near the hot west wall, that contains the window opening, then we note that the upward air flow is higher than in the previous case, and becomes particularly important near the hot angle:



maximum air flow : 0.575 kg/s
surface exchange coefficient : h = 10 W/m2/C
west wall temperature : T = 278 K
east wall temperature : T = 293 K
cell dimensions : LxlxH = 2x2x2m
permeability between cells : k = 0.83 m/s/Pa
top
east
south

We also see that temperatures globally increase.

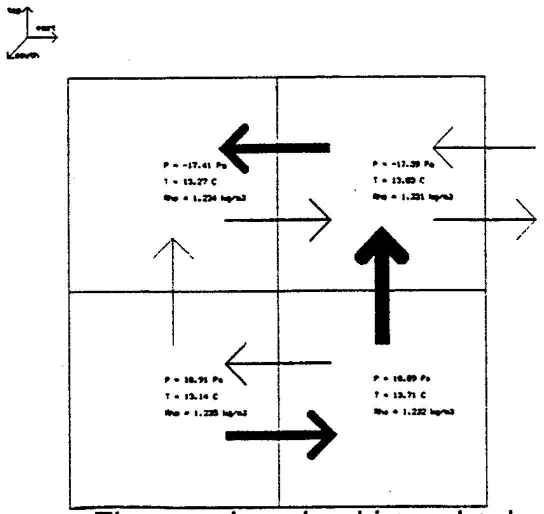
If we consider yet another section of the zone, that contains the door on the hot wall side, then the results are:



maximum air flow : 0.575 kg/s
surface exchange coefficient : h = 10 W/m2/C
west wall temperature : T = 278 K
east wall temperature : T = 293 K
cell dimensions : LxlxH = 2x2x2m
permeability between cells : k = 0.83 m/s/Pa
top
east
south

Last, if we consider a coarser mesh (2x2x2) on the section containing the window, and leave the other conditions unchanged, then we obtain the following results:

```
maximum air flow : 1.77 kg/s
surface exchange coefficient : h = 10 W/m2/C
west wall temperature : T = 278 K
east wall temperature : T = 293 K
cell dimensions : lxlxh = 3x3x3m
permeability between cells : k = 0.83 m/s/Pa
```



The general trend and interpolated values of state variables is preserved.
Results are qualitatively satisfactory. Comparison with Navier Stokes based model in underway. However, the 2D version of the model has already been validated by (Bouia & Dalicieux, 1991) . Since this work is a SPARK 3D implementation of the 2D (Bouia & Dalicieux, 1991) model, the agreement should be satisfactory.

Other tricks to get SPARK to work properly are, while crafting the objects, to be ready to return default expressions for some function outputs when they are fed unphysical arguments during the numerical iteration phase. These defaults are solely used during iteration to allow it to continue instead of returning numerical exceptions, but do not influence the final result. For example, when calculating a temperature from a heat flux equation $\phi = mc_p\Delta T$, one has to be ready to take into count the case were m is zero -in which case t cannot be evaluated. The solution adopted was to return as default value the value of the neighboring temperature. That trick is only a constraint during the iteration phase, to allow the iteration to continue instead of stopping after a division by zero. The final results do not resort to it.

For the steady state 64 cells simulation (4x4x4), the SPARK solver ends up with 128 iteration variables: again, 2 independent variables per cell are automatically chosen by SPARK, the pressure and the temperature. Getting this simulation to work entailed altering some objects so as to make them ready for unphysical arguments during iteration or for some numerical exceptions that might occur due to the choice (during the matching phase) by SPARK of C functions to use. That approach was used in the lower order models also. Here however, the inner tests on whether the top and bottom mass fluxes are zero had to be changed into tests on whether the net interface mass flux was zero.

6 CONCLUSION

In this paper the feasability and suitability of the SPARK object based approach to convection problems is examplified. In particular, the partitioning of the zone into big chunks of space and the linking of subdomain aggregate models is especially well adapted to the object-based methodology, and can be applied in the three dimensional case. The process of writing reusable code and to connect components together is fully automated. And in spite of the lack of local analysis in this coarse meshing, the flow patterns obtained are consistent with the observation.

The presented work deals only with the convective model aspects. In a real simulation, coupling with the conductive walls and introduction of the radiative heat transfer is required. Such an extension will be made easy by the modularity of the SPARK object-based environment. Last, hooking together several zones, themselves partitioned into cells, can be performed in a purely syntactical way.

References

Anderson, Jeffrey L. September 1986. *A Network Definition and Solution of Simulation Problems.* Berkeley, CA 94720: Simulation Research Group, Lawrence Berkeley Laboratory.

Bonneau, D., Covalet, D., Gautier, D., & Rongere, F.X. 1989. *Manuel de Prise en Main, CLIM2000, Version 0.0.*

Bouia, H., & Dalicieux, P. 1991 (August). Simplified Modelling of Air Movements inside Dwelling Room. *In: Proc. of the Building Simulation 91 conference.*

Bring, Axel. 1992 (January). *IDA SOLVER User's documentation.* Kungl Tekniska Hogskolan.

Buhl, Fred, Erdem, Ender, Nataf, Jean-Michel, Winkelmann, Frederick, Moshier, Andrew, & Sowell, Edward. August 30, 1990. *The U.S. EKS: Advances in the SPANK-based Energy Kernel System.* Lawrence Berkeley Laboratory, Report LBL-29419.

Inard, C., & Buty, D. 1991 (August). Simulation of thermal coupling between a radiator and a room with zonal models. *In: Proc. of the Building Simulation '91 conference.*

Inard, Christian. 1988 (July). *Contribution à l'étude du couplage thermique entre un émetteur de chauffage et un local.* Ph.D. thesis, Institut National des Sciences Appliquées de Lyon.

Joly, J.-L. Bonin J.-L., Platel, V., Rigal, M., Granpeix, J.-Y., & Lahellec, A. 1989 (June). Multi-Model Simulation: the T.E.F. approach.

Nataf, Jean-Michel, & Winkelmann, Frederick. 1992 (September). *Applications of Computer Algebra and Compiler-Compilers for Automatic Code Generation in the Simulation Problem Analysis and Research Kernel (SPARK).* LBL Report LBL-32815. Lawrence Berkeley Laboratory.

Sowell, Edward F., & Buhl, Walter F. 1988. Dynamic Extension of the Simulation Problem Analysis Kernel (SPANK). *In: Proceedings of the USER-1 Conference.* Ostend, Belgium: Society for Computer Simulation, La Jolla, CA.

Sowell, Edward F., Buhl, Walter F., Erdem, Ahmet E., & Winkelmann, Frederick C. 1986. A Prototype Object-based System for HVAC Simulation. *In: Proceedings of the Second International Conference on System Simulation in Buildings.* Liege, Belgium: Univ. of Liege, Laboratory of Thermodynamics, B-4000 Liege, Belgium.

Sowell, Edward F., Nataf, Jean-Michel, & Winkelmann, Frederick C. January 1990. *Radiant Transfer due to Lighting: An Example of Symbolic Model Generation for the Simulation Problem Analysis Kernel.* Lawrence Berkeley Laboratory, Report LBL-28273.

Sowell, Edward F., Buhl, Walter F., & Nataf, Jean-Michel. June 1989. Object-oriented Programming, Equation-based Submodels, and System Reduction in SPANK. *Pages 141–146 of: Proceedings of Building Simulation '89.* Vancouver, British Columbia, Canada: International Building Performance Simulation Association, P.O. 282, Orleans, Ontario, K1C 1S7, Canada.