ON THE EFFICIENCY OF PARTITIONING IN OBJECT-BASED SIMULATION

Jean-Michel Nataf and Rolf Ebert

Groupe Informatique et Systèmes Énergétiques*

Abstract

In this paper we present computational experiments on the efficiency of partitioning of domains on the speed of simulation runs with an object-based solver. This exercise is motivated by the emergence of object-based environments where models are encapsulated in objects and communicate with one another, thus yielding extremely modular simulations. Such an approach is convenient for the user. This article tries to determine whether it can be efficient.

The case considered is the model problem in heat conduction on a square plate. Runs are performed on a object-based environment called Motor, that allows sub-models to solve themselves before they enter the main iteration between the various sub-models of the problem. Twoto fourfold partitionings are considered, with different Dirichlet boundary conditions and different properties. Finally correlations are obtained and some qualitative and quantitative conclusions are derived.

1 Introduction

The recent years have seen the emergence of object based simulation environments like IDA in (Sahlin, 1988), CLIM2000 in (Bonneau *et al.*, 1989), SPARK in (Anderson, September 1986) or (Buhl *et al.*, August 30, 1990), and others. These environments allow for encapsulation of models into objects that can then be reused independently from their implementation. Thus the user only needs to hook together the objects into global simulation specification, and the environment takes care of creating a program simulating the whole problem. The advantages of modularity are ease of simulation generation, modification and extension, thus departing from the usual difficulties pertaining to huge, hard to maintain and to modify simulation programs.

Such object-based approaches entails a partition of the overall problem into submodules. For example for a HVAC system simulation the submodules will be fans, coils, zones, etc., thus following the physical interpretation of the problem and adding to the ease of use. In other case, like discretized PDE problems, partitioning is less obvious.

The present paper presents some experiments about the numerical efficiency of partitioning on problems derived from the discretization of the heat equation on a continuous domain. Thus we want to address the question: how to partition a problem not only to keep the representation close to the physical system but also to gain a better efficiency. Of course that efficiency depends on the underlying numerical solver (iterative, Newton-Raphson or other).

Some theoretical results can be obtained in the 1D case, that is the case where components are hooked together in a single line. By and large, the optimal partitioning tends to be the one suggested by the physical insight that recommends to partition at the boundary between insulators and conductors.

In the 2D case, numerical experiments where conducted with an object based prototype, called Motor. Motor is a simulation environment written in Ada, where models are encapsulated objects communicating with each other via variables and signals. In our problem of interest, it allows the hooking together of zones in a 2D domain and searching for a global solution by solving for the

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

temperatures inside the zones, deriving the heat fluxes at the inter-zone boundaries and modifying the inner zone temperatures by Newton-Raphson iteration until the sum of fluxes is zero at the inter-zone boundaries.

2 Theoretical Considerations: the 1D Case

2.1 Statement of the Problem

We consider $n+1$ resistances of transfer coefficients h_i (i.e. resistances $\frac{1}{h_i}$) placed between potential (i.e. temperature) t_0 and t_{n+1}. The resistance i is placed between temperature t_i and t_{i+1}. The equation satisfied between each pair of adjacent resistances is: $\phi_i = h_{i-1}(t_{i-1}-t_i)+h_i(t_{i+1}-t_i) = 0$. This problem represents thermal conduction between layers at steady state with specified temperatures at both ends of the system, or the flow of electricity thru electrical resistances with specified potential at the boundaries.

It is a classical result of the thermal-electric analogy that the solution of this equations system is: for p from 1 to n, $t_p = \frac{\frac{t_0}{\sum_{i=0}^{p-1}\frac{1}{h_i}}+\frac{t_{n+1}}{\sum_{i=p}^{n}\frac{1}{h_i}}}{\frac{1}{\sum_{i=0}^{p-1}\frac{1}{h_i}}+\frac{1}{\sum_{i=p}^{n}\frac{1}{h_i}}}$

This equation is the response of a sequence of $n+1$ layers to the outside conditions t_0 and t_{n+1}, and can be readily obtained by considering the resistance equivalent to resistances grouped in series.

The problem to tackle is to find an "optimal" partitioning of the system so that convergence properties are enhanced. Physically, for example in the case of the thermal problem, this amounts to divide the system into blocks of layers with their own local temperature and flux solution method, and use these blocks as black boxes communicating only thru their perfect contact flux and temperature boundary conditions. In this part we shall consider as "optimal" a partitioning such that the convergence speed at the top level (that is, the coarsest level of components) be highest. This is not, strictly speaking, a measure of the overall computational cost. But it allows to calculate the overall cost by simply summing the costs of all aggregations and all local solution techniques. And it shows better the point of view of the observer watching the system from the outside at the coarsest partition level.

2.2 Binary Partitionings

We first consider only the case where resistances or blocks of resistances can at most be grouped in pairs. Consequently, the iteration speed at the top level (i.e. between the two super blocks hooked together) is simple to obtain. Let us say that the partitioning is made at temperature t_p, between resistance p and resistance $p+1$. The value of t_p when iterating between super blocks is obtained by the flux condition: $t_p = \frac{h_{p-1}t_{p-1}+h_p t_{p+1}}{h_{p-1}+h_p}$.

Now t_{p-1} and t_{p+1} can be obtained by the general formula presented at the beginning, and will thus be calculated (by any numerical method) by their respective super block, with boundary temperatures t_0 and t_p for t_{p-1}, t_p and t_{n+1} for t_{p+1}. Thus these internal sub-block temperature are: $t_{p-1} = \frac{\frac{t_0}{\sum_{i=0}^{p-2}\frac{1}{h_i}}+\frac{t_p}{\sum_{i=p-1}^{p-1}\frac{1}{h_i}}}{\frac{1}{\sum_{i=0}^{p-2}\frac{1}{h_i}}+\frac{1}{\sum_{i=p-1}^{p-1}\frac{1}{h_i}}}$ and $t_{p+1} = \frac{\frac{t_p}{\sum_{i=p}^{p}\frac{1}{h_i}}+\frac{t_{n+1}}{\sum_{i=p+1}^{n}\frac{1}{h_i}}}{\frac{1}{\sum_{i=p}^{p}\frac{1}{h_i}}+\frac{1}{\sum_{i=p+1}^{n}\frac{1}{h_i}}}$

Combining the flux condition and the two sub-block solutions for the temperaturest_{p-1} and t_{p+1} around the partition point, we obtain n an equation yielding the partition point temperature, t_p, in terms of itself, thus obtaining the basis of the iterative method to calculate t_p. The convergence ratio (sort of spectral radius of the linear function giving the next iterated value in terms of the previous one) is then $\frac{1}{h_{p-1}+h_p}\left[\frac{h_{p-1}}{1+\frac{1/h_{p-1}}{\sum_{i=0}^{p-2}\frac{1}{h_i}}}+\frac{h_p}{1+\frac{1/h_p}{\sum_{i=p+1}^{n}\frac{1}{h_i}}}\right]$ and it is obvious it is positive lower than 1 if all transfer coefficients h_i are positive.

Our goal is to minimize that convergence ratio by a suitable choice of p. Noting $r_i = \frac{1}{h_i}$, on can show that this is equivalent t to maximize $\frac{\frac{1}{\sum_{i=0}^{p-1} r_i}+\frac{1}{\sum_{i=p}^{n} r_i}}{\frac{1}{r_{p-1}}+\frac{1}{r_p}}$

Limit cases illuminate a little that criterion. We shall consider the homogeneous case, the embedded insulator case and the embedded conductor case.

If all h_i are equal to a same value h, then the minimum convergence ratio is obtained at $p=1$ or $p=n$, and is $\frac{n-1}{2n}$. Thus the best partition is a partition at the boundary, reminiscent of the original curse that the best partition is no partition at all.

If all r_i are equal to the same value r, except

one r_p much bigger than the others and equal to R (case of an embedded insulator), then partition must occur at t_{p+1} if $p > n - p$, and at t_p if $p < n - p$. Which means that in all cases, partition should occur at closest neighborhood of an embedded insulator in the case of binary partitioning. This could be expected from common sense: an insulator actually separates two conducting blocks.

If all r_i are equal to the same value r, except r_p equal to ρ much smaller, then the optimal cut is at t_1 or t_n, indifferently, except if $p = 1$ or $p = n$. Then the best cut is at t_n or t_1, respectively.

Heuristically, in the binary case, the partition should occur "close to" large resistances and far from the small ones.

2.3 General Case

In the general case, we consider a m-fold partition at $t_{p_1}, t_{p_2}, \ldots, t_{p_m}$. Then as previously, we have, for each partition point temperature t_{p_k}, the flux conservation equation: $t_{p_k} = \frac{h_{p_k-1} t_{p_k-1} + h_{p_k} t_{p_k+1}}{h_{p_k-1} + h_{p_k}}$. and the value returned by the adjacent modules for the adjacent temperatures: $t_{p_k-1} = \frac{\frac{t_{p_{k-1}}}{\sum_{i=p_{k-1}}^{p_k-2} \frac{1}{h_i}} + \frac{t_{p_k}}{1} \frac{1}{h_{p_k-1}}}{\frac{1}{\sum_{i=p_{k-1}}^{p_k-2} \frac{1}{h_i}} + \frac{1}{h_{p_k-1}}}$ and $t_{p_k+1} = \frac{\frac{t_{p_k}}{h_{p_k}} + \frac{t_{p_k+1}}{\sum_{i=p_k+1}^{p_{k+1}-1} \frac{1}{h_i}}}{\frac{1}{h_{p_k}} + \frac{1}{\sum_{i=p_k+1}^{p_{k+1}-1} \frac{1}{h_i}}}$

One can now see that instead of an easy to calculate scalar relation between one iterated value and the next one, we now have a matrix relation yielding the one iterated value of the array of partition point temperatures, and the next iterated value of the array. The matrix obtained is tridiagonal. After some manipulations one can see that the lower diagonal element belonging to row k is $l_k = \frac{r_{p_k-1} r_{p_k}}{r_{p_k-1} + r_{p_k}} \frac{1}{\sum_{i=p_{k-1}}^{p_k-1} r_i}$, that the upper diagonal element is $u_k = \frac{r_{p_k-1} r_{p_k}}{r_{p_k-1} + r_{p_k}} \frac{1}{\sum_{i=p_k}^{p_{k+1}-1} r_i}$ and that the (positive) diagonal element is $d_k = 1 - \frac{r_{p_k-1} r_{p_k}}{r_{p_k-1} + r_{p_k}} \frac{1}{\sum_{i=p_{k-1}}^{p_k-1} r_i} - \frac{r_{p_k-1} r_{p_k}}{r_{p_k-1} + r_{p_k}} \frac{1}{\sum_{i=p_k}^{p_{k+1}-1} r_i} = 1 - l_k - u_k$

So, since all terms are positive, and since there are rows (the first and the last) that have only two terms, we can say that on all rows the sum of the nonzero elements is positive (and equal to one on the second to penultimate row). Now recalling Gershgorin theorem (see for example (Stoer & Bulirsch, 1980)), that states that the distance of eigenvalues to the diagonal elements is lower than the sum of the absolute value of the nondiagonal row elements, we obtain directly the result that all eigenvalues of the matrix are between 0 and 1 (as the diagonal terms). Thus the spectral radius is lower than 1, and there is convergence with convergence ratio equal to the spectral radius of the above mentioned matrix.

3 The Motor Environment

These partitioning considerations were to be tested practically by computing experiments. A simulation program that permits and even encourages the partitioning of the system description is Motor. In the framework of the research project SYMBOL, supported by the French agency for energy management Motor had been developed, a simulation environment that implements the main ideas of object based simulations. The main goal of *object orientation* is to ease reuse of existing program/simulation units to facilitate and accelerate simulation development. The encapsulation of simulation models into objects keeps all data and rules of a real world in a single computer unit and we get a data abstraction which allows a modular representation of real world phenomena. Re-use (and sharing) of these independent modules can be achieved by module libraries as in other environments.

But it happens quite rarely that one will use exactly the same unit in two different system studies. Thus the aspect of easy modification of the units that widens the possibilities of re-use should be considered in the environment. Motor facilitates the description of new models as being similar to an existing one through the notion of inheritance.

The main differences of Motor with other simulation environments consist in

- 1) a hierarchical partitioning of the system as opposed to a final global system of equations and
- 2) simultaneousness in calculation of the different parts as opposed to serial calculation on traditional computers. Both points are true at simulation time itself.

3.1 Partitioning in Motor

We take the view that the computer representation should stay as close as possible to the physical problem. All data belonging to a real world object is gathered in a computer object together with its own active elements.

The visibility of a module's internal data is defined by so-called *frontiers*. These frontiers enhance the modularity of the simulation system since we can replace a module by another one with the same external appearance (type of frontier). The general advantage of modularity is the ease of simulation generation where the user only has to connect predefined modules selected from a module-library.

Splitting up the system into subsystems and the subsystems into subsubsystemss and so on gives us a hierarchical cut of the system. At every step this leads to more and more detailed description with a tree structure. Here the ramifications represent composite modules or parent units. They are composed of several sub or children units. The buds of the tree are terminal or atomic modules that cannot be split anymore. Generally they are affected to basic components (e.g. a heater, wall). This cutting gives us a structure of dependencies between the modules where the dependencies must be reconstituted in the simulation system. In Motor this hierarchical description of the system is kept even at runtime. Thus, we have computer units (and thus algorithms to treat) not only in the terminal buds of the tree, but also for every ramification; and this on every level of the tree.

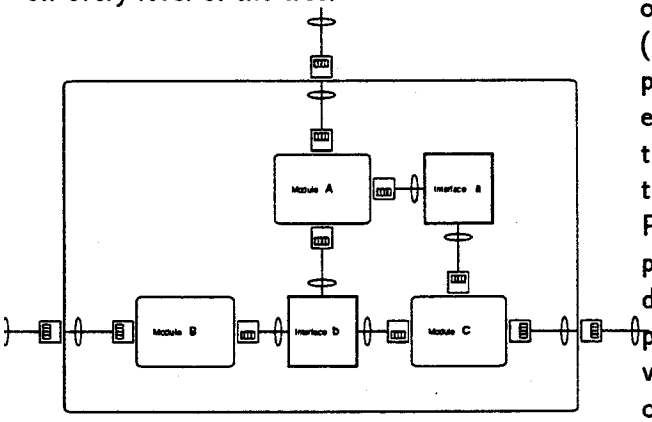

Figure 1: a composite module with two interfaces and three submodules. The interface **a** connects the submodules **A** and **C**. All three submodules **A**, **B**, and **C** are coupled by the interface **b**.

The composite modules mainly consist of what we call interfaces in Motor. Generally several interfaces are within one composite module. These interfaces are communication points which connect two or more submodules by its frontiers. Besides the connection information the interfaces contain the binding equations (e.g. equality of temperatures in all submodules and sum of heat fluxes is zero). For one parent unit with several interfaces the user can choose between different algorithms to solve the resulting systems of equations. Thus the parent unit controls the evolution of its submodules via the interfaces and its own outer frontiers. By this we built a hierarchical system of equation systems. The equations of a top level composite module are in fact equation systems by themselves.

4 Computer Experiments: the 2D Case

The 2D case is not susceptible to a simple analytical statement, as is the 1D case [1]. Therefore one has to resort to computer experiments. As described above, Motor allows to have sub-domains with their own internal solution technique, communicating with their neighbors through a relaxation method on the values of fluxes and temperatures at the inter-domain frontiers (for more details see (Ebert, 1989)). The Motor environment is therefore suitable for investigating the efficiency of domain partitioning.

The approach adopted here is to consider a very simple case, namely the standard model of conduction on a 2D plate (see for example (Varga, 1962) or (Golub & Meurant, 1983)). The plate is square, with 8^2 grid points. Finite differences are used. Various Dirichlet boundary conditions are considered. Admissible partitionings of the plate are generated by a program written in Prolog (derived from (Allaz, 1992)). The Prolog program takes as argument the number of subdomains needed, creates all partitionings of the plate, and, after each partitioning generation, invokes a translator that creates a bitmap picture of these partitioning. These are then parsed by a preprocessor that creates the Motor code simulating that partitioning. Thus are created, for each partitioning, all the elementary computational cell Motor objects, plus the sub-domains

[1] However, in (Padovan & Kwang, 1991), one can find an asymptotic analysis on the overall cost depending on the partitioning as well as guidelines for optimizing the partition

linking these, plus the Motor plate model linking together the sub-domains.

Then the prolog program invokes Motor. The simulation problem is solved, and the machine time needed is saved, along with the partitioning representation, in a log file. Then the prolog program continues and creates a new partitioning.

Eye inspection of the results can be carried out for qualitative results. For more quantitative results, the output log file is then read by another processor that generates a table of the run time, along with the surface of each sub-domain, and the overall length of the inner frontiers. Code for a linear multi-regression of these data is also generated, and then run. So at the end one obtains a correlation for the run time versus symmetric expressions of the sub-domain areas, up to polynomial order 3, and and of the powers of the length of the inner frontier, up to order 3. Due to the influence of the difference between the initial guesses and the final solution, measures of how far off the initial guess is, is also introduced.

Last, due to the inevitable dispersion of the run time for identical cases, the same cases are run several times in each measurement series so as to obtain a better estimation of the run time.

5. Results

Interpretation of the results is not obvious. We first consider the constant conductivity case, alias the "homogeneous case". Then we consider cases with varying conductivity.

5.1 The Homogeneous Case

5.1.1 Influence of the Partition Geometry

Qualitative remarks are, that the time efficiency is better when the sub-domains have comparable sizes. Thus introducing very small domains in a partitioning is likely to introduce inefficiencies. That is sort of confirmed by common sense, since introducing a very small sub-domain hardly changes the complexity of the neighboring sub-domains, but increases the boundary length (and therefore the dimension of the jacobian matrix to be inverted during the Newton Raphson iteration) by a finite amount.

Also, it appears that when one has a partitioning, (imposed by the physical description of the model), it can be worth it to partition further, for the sake of time efficiency. In general, for low number of sub-domains, and unless one departs from no partitioning at all, the higher the number of sub-domains, the better. For example, a simple ternary partitioning is better than the binary partitioning it comes from, and worse than the ensuing 4-partitioning, as shown below:

```
      150.3 user         13.5 sys
+++++++++
+        +
+++++++++
+        +
+        +
+        +
+        +
+        +
+++++++++

      101.9 user          9.4 sys
+++++++++
+        +
+++++++++
+        +
+++++++++
+        +
+        +
+        +
+++++++++

       77.7 user          7.4 sys
+++++++++
+        +
+++++++++
+        +
+++++++++
+        +
+++++++++
+        +
+++++++++
```

The above trend is confirmed by the correlations obtained. These correlation are derived from a limited set of mildly uncertain data (due to the slight variations of the run time as metered), and are also dependent on the particularities of the Motor environment, but they give an idea of what to expect when using similar types of solution techniques. In the sample case with conductivity $10^{-3}W/m^2K$ fixed temperature $5°C$ at top side, $10°C$ at bottom side, $15°C$ at left side and $20°C$ at right side, with initial guesses equal to $19°C$, we consider first the correlations that are independent of the difference between initial guess and final result at the inner sub-domain boundary:

- 2-partitioning: $Cost = 219.5 - 1093.3\sigma_1 + 2300.82\sigma_2 - 1095.9\sigma_3 + 0.4l - 98.3l^2 - 34.1l^3$

- 3-partitioning: $Cost = 215.3 - 130.1\sigma_1 - 1775.3\sigma_2 + 4163.8\sigma_3 + 109.6l - 33.1l^2 - 6.4l^3$

- 4-partitioning: $Cost = 268.3 + 123.4\sigma_1 - 562.4\sigma_2 + 819.3\sigma_3 + 145.8l - 222.4l^2 + 57.7l^3$

where $Cost$ is the time cost for the considered Motor run on the partition considered, σ_i is the scaled sum of the ith powers of the n sub-domain areas, and l the scaled length of the inner frontier. Scaling is performed by dividing lengths by the overall domain length and surfaces by the overall domain surface. The correlations for these formulas are, respectively: 0.79, 0.69, 0.51.

5.1.2 Influence of the Boundary Conditions

The results, and the derived correlations are themselves very dependent on the boundary conditions. It is observed that equivalent partitionings lead to fairly different results when the boundary conditions do not have the symmetry (if any) of the partitionings. That makes sense: if the initial guesses for the solution are farther away from the solution, the convergence will take longer to achieve. To eliminate this bias, three aggregate measures of the distance between the initial guess and the final solution were introduced; these measures are :

$\iota_i = \int_{InnerFrontier} (T_{guess}(s) - T_{sol}(s))^i ds$

with T_{guess} the initial guess at any point s of the inner frontier between sub-domains, T_{sol} the solution at the same point, i an integer taking the values 1,2 and 3, and ds the measure on the inner frontier. Also, the length l of the frontier would be sort of equivalent to the measure ι_0, if there was one.

Introducing this bias, the correlation obtained is more representative and applicable. For the cases examined, we obtain:

- 2-partitioning: $Cost = 408.1 - 806.9\sigma_1 + 1593.9\sigma_2 - 756.6\sigma_3 + 1262.8\iota_1 + 3306.5\iota_2 + 2966.7\iota_3 - 78.1l + 108.7l^2 - 24.4l^3$

- 3-partitioning: $Cost = -213.3 + 49.8\sigma_1 - 2448.6\sigma_2 + 4856.0\sigma_3 - 3680.1\iota_1 - 14984.3\iota_2 - 16699.0\iota_3 + 495.8l - 256.4l^2 + 35.1l^3$

- 4-partitioning: $Cost = 207.6 + 116.1\sigma_1 - 523.3\sigma_2 + 785.5\sigma_3 - 1802.9\iota_1 - 6840.4\iota_2 - 7271.4\iota_3 + 24.6l - 143.3l^2 + 42.1l^3$

where ι_i the i-th measure of the scaled difference between the guess temperature and the solution. Scaling on the temperature is done by dividing the temperature differences by the difference between the highest and the lowest temperature in the overall system. The correlations are respectively here: $0.86, 0.71, 0.52$.

One can see that the coefficient attached to these measures are not negligible, although the improvement in terms of correlation between the data and the formula is not significant.

5.2 Non Homogeneous Case

All the above is valid for constant conductivity (i.e., in the Motor formalism, constant thermal resistances on all four sides of each computational cell). Further experiment were conducted with discontinuous conductivity. In one case, the domain has a vertical straight conductivity discontinuity close to the middle axis, with $K = 0.001$ on the left side and $K = 1.0$ on the right side. We will refer to that case as to the "fixed inhomogeneity" case. In another case, the discontinuity is systematically imposed on the partitioning, for each partitioning automatically generated. We will refer to that case as to the "domain inhomogeneity" case.

5.2.1 Fixed Inhomogeneity Case

It is observed, for a binary partitioning and on the limited sample results obtained, that the introduction of inhomogeneities in the domain lead to higher run times (in general higher by 2 to 10 %) than for the homogeneous case, whatever be the partitioning. The only -notable- exception occurs when the fixed discontinuity is exactly following the partition. That results confirms the theoretical results obtained for the 1D case, and is comforting in the sense that partitioning domains with respect with their physical differences is an efficient way to go.

5.2.2 Domain Inhomogeneity Case

It is observed that in all cases, introducing conductivity differences along the *existing* partitioning decreases the run time by 3% to 12% in general. That fact is consistent with the previous observation that partitioning along an existing discontinuity is efficient. The difference is smaller when the partitioning is regular with thick

sub-domains, and higher when the partitioning is crooked with thin sub-domains.

6 Conclusion

In this article we show that under certain circumstances, domain decomposition in object-based environments is not only a syntactical necessity, but also can yield reward from an efficiency point of view. That is shown analytically in the 1D case, and experimentally on the computer in the 2D case. The experimental guidelines derived from the computational experiments for the model problem are to increase the number of subdomain while not keeping them too small. These guidelines are very dependent of Motor, but they give an idea of the kind of optimization entailed when using certain types of object-based simulation environments.

The work presented here still is fragmentary, and can be extended in several ways. First it will be useful to gather a bigger set of data to build upon, and to obtain more reliable correlations. Also, the partitions considered here are not hierarchical, i.e. there are no nested partitions. Motor allows hierarchical partitioning, and it would be worthwhile to investigate their efficiency. Last, applications of the efficiency of partitioning in real case simulations remains to be demonstrated.

For this last desirable extension, preliminary experiments on a zone limited by walls undergoing convection, convection and radiation show a speedup of 2 (if one neglects the initial input overhead). Therefore even in real problems, partitionning can improve performance.

References

Allaz, Christophe. 1992 (June). *Développement d'un générateur expert de plan masse.* Stage scientifique ENPC.

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.*

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.

Ebert, Rolf. 1989. *New Modular and Structured Approach to the Study of Complex Thermal Systems.* École Nationale des Ponts et Chaussés and École des Mines de Paris.

Golub, Gene H., & Meurant, Gerard A. 1983. *Résolution Numérique des Grands Systèmes Linéaires.* Collection de la Direction des Études et Recherches d'Électricité de France, vol. 49. Eyrolles.

Padovan, Joe, & Kwang, Abel. 1991. Parallelized solvers for heat conduction formulations. *Numerical Heat Transfer, Part B,* **19**, 127–152.

Sahlin, Per. 1988. *IDA, a Modelling and Simulation Environment for Building Applications.* Stockolm, Sweden: Institute of Applied Mathematics. P.O. Box 26300, S-100 41 Stockholm, Sweden.

Stoer, J., & Bulirsch, R. 1980. *Introduction to Numerical Analysis.* New York Heidelberg Berlin: Springer-Verlag.

Varga, R. S. 1962. *Matrix Iterative Analysis.* Series in Automatic Computation.Englewood Cliffs: Prentice Hall.