NASA Technical Memorandum 86448

NASA-TM-86448 19850024509

CONCURRENT IMPLEMENTATION OF THE CRANK-NICOLSON METHOD FOR HEAT TRANSFER ANALYSIS

JONATHAN B. RANSOM AND ROBERT E. FULTON

JUNE 1985

Corrected Copy

NASA
National Aeronautics and Space Administration

Langley Research Center
Hampton, Virginia 23665

CONCURRENT IMPLEMENTATION OF THE CRANK-NICOLSON METHOD FOR HEAT TRANSFER ANALYSIS

Jonathan B. Ransom
NASA Langley Research Center
Hampton, Virginia

Robert E. Fulton
George Washington University
NASA Langley Research Center
Hampton, Virginia

SUMMARY

Equations of large-order structures problems are often intractable on current sequential computers due to memory and execution time limitations. The introduction of a new generation of Multiple Instruction Multiple Data (MIMD) computers provides an opportunity for significant gains in computing speed and can make tractable the solution to large-order problems. One such problem class is heat transfer analysis which is important to many aerospace applications. To exploit this opportunity, concurrent methods for solution to heat transfer analysis problems need to be investigated. This paper describes two alternate implicit methods for time integration of the heat transfer equations on a concurrent processing computer; the first is an iterative technique and the second is a direct technique. The paper also discusses how these methods were implemented on a specific experimental MIMD computer, and gives timing and response results for the solution to an example transient test problem. Execution times of the direct technique are approximately one half

those of the iterative technique. However, results indicate that as the number of iterations per time increment decreases, it becomes more attractive to use the iterative solution technique. This decrease in execution time per time increment implies that, in a concurrent environment, an analyst may want to solve a heat transfer problem initially using a direct method and subsequently, switch to an iterative technique when the temperature gradients are more predictable.

INTRODUCTION

The solution of the system of equations resulting from discretization of large-order heat transfer problems is often very demanding in both execution time and memory on sequential computers. The introduction of a new generation of computers based on concurrent processing offers an alternative that promises to alleviate these limitations. A concurrent processing computer contains multiple processors which may operate simultaneously to share the computational load and the memory requirements. Development of appropriate algorithms for these computers can reduce computation time and can make tractable the solution of large complex problems. The near-term introduction of a new series of multi-processor computers termed Multiple Instruction Multiple Data (MIMD) computers promises to provide a significant advantage in high-speed computational scientific capability. Simple implementation of existing sequential algorithms will not take full advantage of the capability provided by MIMD computers. To utilize this opportunity for heat transfer, concurrent computational methods appropriate to heat transfer analysis problems need to be investigated. There are a number of solution techniques for solving such problems and the proper choice of technique can be critical for the efficient solution on MIMD computers. This paper explores two alternate implicit methods

for the solution of the heat transfer equation, one based on an iterative method and the other on a direct method. The paper discusses their implementation and gives results for the solution of an example transient test problem. This study was carried out using a research MIMD computer called the Finite Element Machine (FEM) at NASA Langley Research Center.

DESCRIPTION OF TEST PROBLEM

The test problem is to determine the subsequent temperature values throughout a uniform rod when given the initial temperature distribution and the history of the end conditions. The rod and boundary conditions are shown in Figure 1. Distance along the rod is denoted by x, time by t, and the temperature of the rod by $T(x,t)$. The rod is insulated on the left end and initially has a uniform unit temperature along the length except at the right end where the temperature has been instantaneously reduced to zero. At the right end, the temperature is taken to be one half initially and zero afterwards. The temperature gradient at the left end is zero.

DESCRIPTION OF NUMERICAL METHOD

The general one dimensional heat conduction equation is

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2} \tag{1}$$

where $\alpha = \frac{K}{\rho c_p}$

and K is the thermal conductivity, ρ is the mass density, c_p is the specific heat, and α is the thermal diffusivity. Finite difference approximations are written for this expression. The rectangular finite difference grid

is shown in Figure 2. The spacing in the x direction is h and that in the t direction is k. The mesh numbers in the x and t directions are i and j, respectively. The first derivative with respect to time is approximated by central half station differences and the second derivative with respect to x by central whole station differences. The implicit Crank-Nicolson method (Ref. 1) was chosen in this study because it is stable, has error of order h^2 and k^2, and appears to be well-adapted to concurrent processing. The method uses an average of approximations in the j and $j+1$ rows and results in the solution of tridiagonal sets of simultaneous equations. Using the notation of Figure 2, the heat-conduction equation for point $(i, j + 1/2)$ reduces to

$$T_{i,j+1} = T_{i,j} + \frac{k\alpha}{2} \left[T_{xx_j} + T_{xx_{j+1}}\right] \tag{2}$$

or

$$T_{i,j+1} = T_{i,j} + \beta \left[\{T_{i-1,j} - 2T_{i,j} + T_{i+1,j}\} + \{T_{i-1,j+1} - 2T_{i,j+1} + T_{i+1,j+1}\}\right] \tag{3}$$

where $\beta = \frac{k\alpha}{2h^2}$

and $T_{i,j}$ is the temperature of the ith degree of freedom at the jth time increment and subscript x denotes differentation with respect to the coordinate x.

Two methods for determining the second derivative approximation in the $j+1$ row are: 1) by assuming it initially and iterating; or 2) by moving the term to the left hand side of the equation and solving directly. These two methods are denoted "iterative" and "direct" in the text. For the iterative scheme, equation (2) can be written with superscripts to denote the iteration number, r, as follows

$$T^{r+1}_{i,j+1} = T_{i,j} + \frac{k\alpha}{2}\left[T_{xx_j} + T^{r}_{xx_{j+1}}\right] \tag{4}$$

For the first iteration, $T_{xx_{j+1}}$ is assumed to be equal to T_{xx_j} and computation proceeds until $T^{r+1}_{xx_{j+1}}$ and $T^{r}_{xx_{j+1}}$ are within a given convergence criterion. For the direct scheme, equation (3) can be written as

$$\begin{aligned} -\beta T_{i-1,\,j+1} + \{1+2\beta\}T_{i,\,j+1} - \beta T_{i+1,\,j+1} = \\ \beta T_{i-1,\,j} + \{1-2\beta\}T_{i,j} + \beta T_{i+1,j} \end{aligned} \tag{5}$$

Equation (5) results in a set of tridiagonal equations required to be solved in moving from station j to $j+1$ as follows

$$\begin{bmatrix} 1+2\beta & -2\beta & & \\ -\beta & 1+2\beta & -\beta & \\ & \ddots & \ddots & \ddots \\ & & -\beta & 1+2\beta \end{bmatrix} \begin{Bmatrix} T_{1,j+1} \\ T_{2,j+1} \\ \vdots \\ T_{n,j+1} \end{Bmatrix} = \begin{bmatrix} 1-2\beta & +2\beta & & \\ +\beta & 1-2\beta & +\beta & \\ & \ddots & \ddots & \ddots \\ & & +\beta & 1-2\beta \end{bmatrix} \begin{Bmatrix} T_{1,j} \\ T_{2,j} \\ \vdots \\ T_{n,j} \end{Bmatrix} \tag{6}$$

In this case all the unknowns at $j+1$ can be obtained through solution of the tridiagonal system of equations.

CONCURRENT PROCESSING IMPLEMENTATION

On a concurrent computer (Refs. 2-5), one may distribute, over different processors, the solution of specific nodal point equations. Communication between the processors is accomplished by the interprocessor communications capability of the concurrent computer. For the MIMD computer used for these studies, the user can invoke either direct nearest-neighbor communications or global bus communications for distant processors. In addition, on concurrent processors, one must consider issues such as how much of the computation can be performed concurrently, how this computation should be distributed, how many processors should be used, and how much communication should occur between processors. At some point, communication may become too time consuming, in which case a trade-off must be made between sequential and concurrent computing. These implementation issues are dependent on the architecture of each concurrent computer; a brief summary of the FEM hardware and software is given below.

Concurrent Processing Hardware

The concurrent processing hardware used in this study is the NASA, Langley Finite Element Machine (FEM) shown schematically in Figure 3. FEM consists of an array of 16 processors[1] which can communicate with each other over local links or global bus paths. The controller uses the global bus as indicated in Figure 3. FEM's flexible communication structure provides twelve nearest-neighbor links of which eight are shown in Figure 3. The global communications bus allows communication from one processor to any or all other processors. The

[1] When this research was carried out, FEM had 12 processors.

hardware contains a global flag network that can be used to signal the completion of a process.

Software

The controller is the user interface to the Finite Element Machine. The software on the controller for FEM is an extended version of the menu-driven minicomputer operating system. In addition, software termed PASLIB (Ref. 6) was written to support a set of companion commands residing on each processor of the FEM. A user constructs a concurrent algorithm on the controller and invokes a command to transfer the program and associated data to each processor or a subset of processors. Communication is accomplished by including the appropriate calls to PASLIB procedures (SEND and RECEIVE) in the concurrent algorithm.

CONCURRENT ITERATIVE TECHNIQUE APPROACH

For this concurrent implementation, m thermal equations are distributed and solved on n processors. A typical distribution of computation is shown below for the solution of n equations on n/2 processors.

$$T_{1,j+1} = T_{1,j} + \frac{k\alpha}{2}\{T_{xx_{1,j}} + T_{xx_{1,j+1}}\}$$

$$T_{2,j+1} = T_{2,j} + \frac{k\alpha}{2}\{T_{xx_{2,j}} + T_{xx_{2,j+1}}\}$$

> Processor 1 (7)

$$T_{3,j+1} = T_{3,j} + \frac{k\alpha}{2}\{T_{xx_{3,j}} + T_{xx_{3,j+1}}\}$$

$$T_{4,j+1} = T_{4,j} + \frac{k\alpha}{2}\{T_{xx_{4,j}} + T_{xx_{4,j+1}}\}$$

> Processor 2

$$T_{n-1,j+1} = T_{n-1,j} + \frac{k\alpha}{2}\{T_{xx_{n-1,j}} + T_{xx_{n-1,j+1}}\}$$

$$T_{n,j+1} = T_{n,j} + \frac{k\alpha}{2}\{T_{xx_{n,j}} + T_{xx_{n,j+1}}\}$$

> Processor n/2

All processors perform the same functions. The computation flow for one degree of freedom per processor is shown in Figure 5. Each column of the figure represents the computation flow for the assigned degree of freedom. The procedure can be readily extended to multiple degrees of freedom per processor. Moving from time j to $j+1$ begins with an assumed second derivative $T_{xx_{j+1}}$ for each assigned degree of freedom, the corresponding temperatures are calculated using equation (4). The computation is interrupted so each processor can communicate results for its assigned degrees of freedom to its neighboring processors. The second derivative is then computed by standard central difference approximations and compared to the assumed second derivatives. If a given convergence criterion for the degrees of freedom is met, local convergence is achieved. The FEM flag network is used to check for convergence of all processors (i.e., global convergence). If either the local or the global convergence test fails, each processor uses the current calculated values of the second derivatives as the assumed values and repeats the computation. When global convergence is achieved, all processors simultaneously proceed to the next time step.

CONCURRENT DIRECT TECHNIQUE APPROACH

Approximation of the second derivatives by central differences results in a tridiagonal system of equations. Therefore, a tridiagonal equation

solver was chosen as the direct solution technique. The cyclic reduction method was chosen because it appeared to lend itself best to concurrent computations (Ref. 7). The resulting tridiagonal system of equations is distributed evenly over n processors. A typical distribution of the n rows of the tridiagonal matrix on $n/2$ processors is as follows

$$\begin{bmatrix} b & c & & & & & & \\ a & b & c & & & & & \\ & a & b & c & & & & \\ & & a & b & c & & & \\ & & & & \ddots & & & \\ & & & & & a & b & c \\ & & & & & & a & b \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \text{Processor 1} \\ \text{Processor 2} \\ \\ \\ \\ \text{Processor n/2} \end{matrix} \qquad (8)$$

where a, b, and c are coefficients of the $j+1$ matrix shown in equation (6). Equation (6) is used to solve for the unknowns at the $j+1$ increment in time directly. Computation begins with the coefficients a, b, and c of the original system of equations stored in an array denoted P shown in Figure 6. Each column of the figure represents the computation flow for the assigned rows of the matrix of coefficients in equation (8) where q is the cycle number in the cyclic reduction algorithm. Computation is interrupted for each processor to communicate these coefficients to its neighboring processors. New coefficients are then calculated by row operations on the matrix to eliminate variables. If there have been enough cycles to sufficiently reduce the original matrix, the temperatures are calculated. Otherwise, each processor repeats the computation until the matrix has been reduced sufficiently, after which the temperatures are calculated. Each

processor then simultaneously proceeds to the next time step and communicates its temperature results in order to calculate a new right hand side of equation (6). The cyclic reduction is redone with each time increment which is required if α is a function of time, t, or space, x.

RESULTS

Results were obtained for a transient thermal test problem by an iterative method and a direct method for solving tridiagonal systems of equations. Temperature versus time results are shown in Figure 7.

The primary results are the computational times needed to calculate the temperature distribution on a varied number of processors. The computational speedup (the computation time required to calculate results on one processor divided by the computation time required to calculate the same results on n processors) derived from the concurrent approach can then be computed. The computational speedup versus the number of processors for the iterative and direct techniques are shown in Figures 8 and 9, respectively. The theoretical maximum speedups would be the speedups if there were no overhead for concurrent processing and therefore, are equal to the number of processors used. For a system of 24 equations, the speedup for the iterative technique is 6.9 on 12 processors. The speedup for the same system of equations using the direct technique is 5.6 on 12 processors. The speedup values show that the potential for decreasing the computational speed of the iterative technique is greater than that of the direct method. However, the execution times for the direct method are approximately half those of the iterative technique. The speedups fall short of the theoretical limit because the amount of communication required is relatively large compared to the amount of computation for this problem size.

The speedup of the iterative technique relative to the direct technique is called the relative speedup. It is defined as the execution time of the direct technique for one time increment on one processor divided by the execution times of the iterative technique for one time increment on n processors. This ratio may be used to compare the execution times of the direct and iterative techniques by assuming convergence of the iterative technique after a given number of iterations. The relative speedups for a varied number a iterations are shown in Table 1. The corresponding curves for the relative speedups in the table are shown in Figure 10. The solid curve is the relative speedups of the direct technique for one time increment and the dashed curves are the relative speedups of the iterative technique for one time increment. The iterative technique is faster than the direct technique when its relative speedups are greater than those of the direct technique.

The timing results for the example problem indicate that when the number of iterations per time increment is less than five, the iterative technique is faster than the direct technique as shown in Figure 10. For the example problem, the number of iterations per time increment varied from 14 iterations to 4 iterations. The decrease in execution time per time increment implies that in a concurrent environment an analyst may want to consider using the direct technique initially and then switching to the iterative solution technique when the number of iterations is known to be small. Although an iterative technique is generally not used for the solution of a linear heat transfer problem, interchanging algorithms may be an effective way to reduce further the computational speed of heat transfer problem solutions.

EXTENSION TO NONLINEAR PROBLEMS

The extension of the concepts discussed herein to the solution of problems with nonlinear effects, such as surface radiation, creates added complexities. One example of such a nonlinear problem is the one dimensional heat-conduction equation with surface radiation

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2} + \varepsilon \sigma T^4 \tag{9}$$

where ε is the surface emissivity parameter and σ is the universal physical constant.

Application of the Crank-Nicolson method to the nonlinear heat transfer problem and using the average value for the nonlinear term leads to

$$T_{i,j+1} = T_{i,j} + \frac{k\alpha}{2} \{T_{xx_j} + T_{xx_{j+1}}\} + \frac{k\varepsilon\sigma}{16} \{T_{i,j} + T_{i,j+1}\}^4 \tag{10}$$

The iterative solution becomes

$$T^{r+1}_{i,j+1} = T_{i,j} + \frac{k\alpha}{2} \{T_{xx_j} + T^r_{xx_{j+1}}\} + \frac{k\varepsilon\sigma}{16} \{T_{i,j} + T^r_{i,j+1}\}^4 \tag{11}$$

and the concurrent solution method proceeds as discussed for the linear problem.

The nonlinear direct method can be solved using a Newton-Raphson method which leads to tridiagonal equations that can be solved as follows

$$T^{r+1}_{i,j+1} = T^r_{i,j+1} + \delta T_{i,j+1} \tag{12}$$

and the sequence of direct calculations within a time increment take the form of

$$\begin{bmatrix} (1+2\beta)-\gamma & -2\beta & & \\ -\beta & (1+2\beta)-\gamma & -\beta & \\ & \ddots & \ddots & \ddots \\ & & -\beta & (1+2\beta)-\gamma \end{bmatrix}^{r} \begin{Bmatrix} \delta T_{1,j+1} \\ \delta T_{2,j+1} \\ \vdots \\ \delta T_{n,j+1} \end{Bmatrix} = \begin{bmatrix} 1-2\beta & +2\beta & & \\ +\beta & 1-2\beta & +\beta & \\ & \ddots & \ddots & \ddots \\ & & +\beta & 1-2\beta \end{bmatrix}^{r} \begin{Bmatrix} T_{1,j} \\ T_{2,j} \\ \vdots \\ T_{n,j} \end{Bmatrix}$$

$$- \begin{bmatrix} 1+2\beta & -2\beta & & \\ -\beta & 1+2\beta & -\beta & \\ & \ddots & \ddots & \ddots \\ & & -\beta & 1+2\beta \end{bmatrix}^{r} \begin{Bmatrix} T_{1,j+1} \\ T_{2,j+1} \\ \vdots \\ T_{n,j+1} \end{Bmatrix}^{r} + \frac{k\varepsilon\sigma}{16} \begin{Bmatrix} (T_{1,j} + T_{1,j+1})^4 \\ (T_{2,j} + T_{2,j+1})^4 \\ \vdots \\ (T_{n,j} + T_{n,j+1})^4 \end{Bmatrix}^{r} \tag{13}$$

where $\gamma = \frac{k\varepsilon\sigma}{4} (T_{i,j} + T_{i,j+1})^3 \qquad i = 1 \ldots n$

The coefficients of the tridiagonal matrix on the left hand side of equation (13) change with each iteration, therefore the cyclic reduction must be redone with each iteration. Thus, for each increment in time, a sequence of direct calculations must be carried out to obtain a changed result at $j+1$. Therefore, the iterative approach would appear more attractive for the solution of nonlinear heat transfer problems on a concurrent processing computer.

CONCLUDING REMARKS

An iterative technique and a direct solution technique were used to solve a transient heat transfer problem on a varied number of processors of a

MIMD computer and the performance results were compared. A description of both methods, their implementation, and results are discussed. Both methods provided significant speedup in computations as the number of processors increased indicating that parallel computers can be beneficial in solving complex heat transfer problems. Results show that the iterative technique would benefit more than the direct technique if the number of processors were increased. The results also show that although the direct technique is the faster of the two for sequential calculations, the iterative technique is faster on a concurrent processing computer when the number of iterations per time increment is known to be small.

For nonlinear problems, the formulation of the iterative technique remains basically the same as for linear problems. The direct technique, however, is reformulated such that it requires several Newton-Raphson cycles per time increment to obtain a result at a given finite difference station. Therefore, the advantage of the iterative technique increases significantly for concurrent computers.

Many future computers for large-scale engineering analysis will be multiple instruction multiple data systems. While further algorithmic studies are needed in several problem areas to help refine solution strategies, the results of this study suggest that heat transfer analysis problems are well suited for implementation on future parallel computers.

REFERENCES

1. Crank, J. and Nicolson, P.: A Practical Method for Numerical Evaluation of Solutions of Partial Differential Equations of Heat-Conduction Type. Proceedings of the Cambridge Philosophical Society, Vol. 32, 1947, pp. 50-67.

2. Storaasli, O.; Ransom, J. B.; and Fulton, R.: Structural Dynamics Analysis on a Parallel Computer: The Finite Element Machine. AIAA Paper No. 84-0966-CP, May 1984.

3. Ransom, J.; Storaasli, O.; Fulton, R.: Application of Concurrent Processing to Structural Dynamic Response Computations. NASA CP 2335, 1984 pp. 31-44.

4. Storaasli, O.; Peebles, S.; Crockett, T.; Knott, J.; and Adams, L.: The Finite Element Machine: An Experiment in Parallel Processing. Research in Structural and Solid Mechanics, 1982, NASA CP 2245, also, NASA TM 84514, July 1982.

5. Noor, A. K.; Storaasli, O. O.; Fulton, R. E.: Impact of New Computing Systems on Finite Element Computations. Impact of New Computations on Computational Mechanics, A. Noor, Ed., ASME Special Publication H00275, 1983, pp. 1-32.

6. Crockett, T. W. and Knott, J. D.: System Software for the Finite Element Machine. NASA Contractor Report 3870, February 1985.

7. Hockney, R. W. and Jesshope, C. R.: Parallel Computers, Architecture, Programming and Algorithms. Adam Hilger Ltd., Bristol, 1981, pp. 286-294.

TABLE 1 - RELATIVE SPEEDUPS FOR THE ITERATIVE TECHNIQUE

	Speedup				
Processors	1 iteration	2 iterations	3 iterations	4 iterations	5 iterations
1	3.51	1.76	1.17	0.88	0.70
2	5.71	2.90	1.90	1.43	1.14
4	10.14	5.07	3.38	2.54	2.03
6	14.02	7.01	4.67	3.51	2.80
8	17.50	8.75	5.83	4.38	3.50
12	23.26	11.63	7.75	5.82	4.65

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}$$

$$T = T(x, t)$$

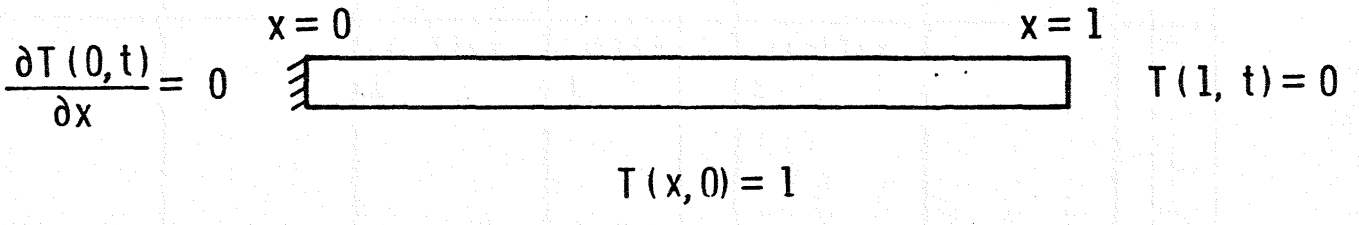


Figure 1.- Transient thermal problem.

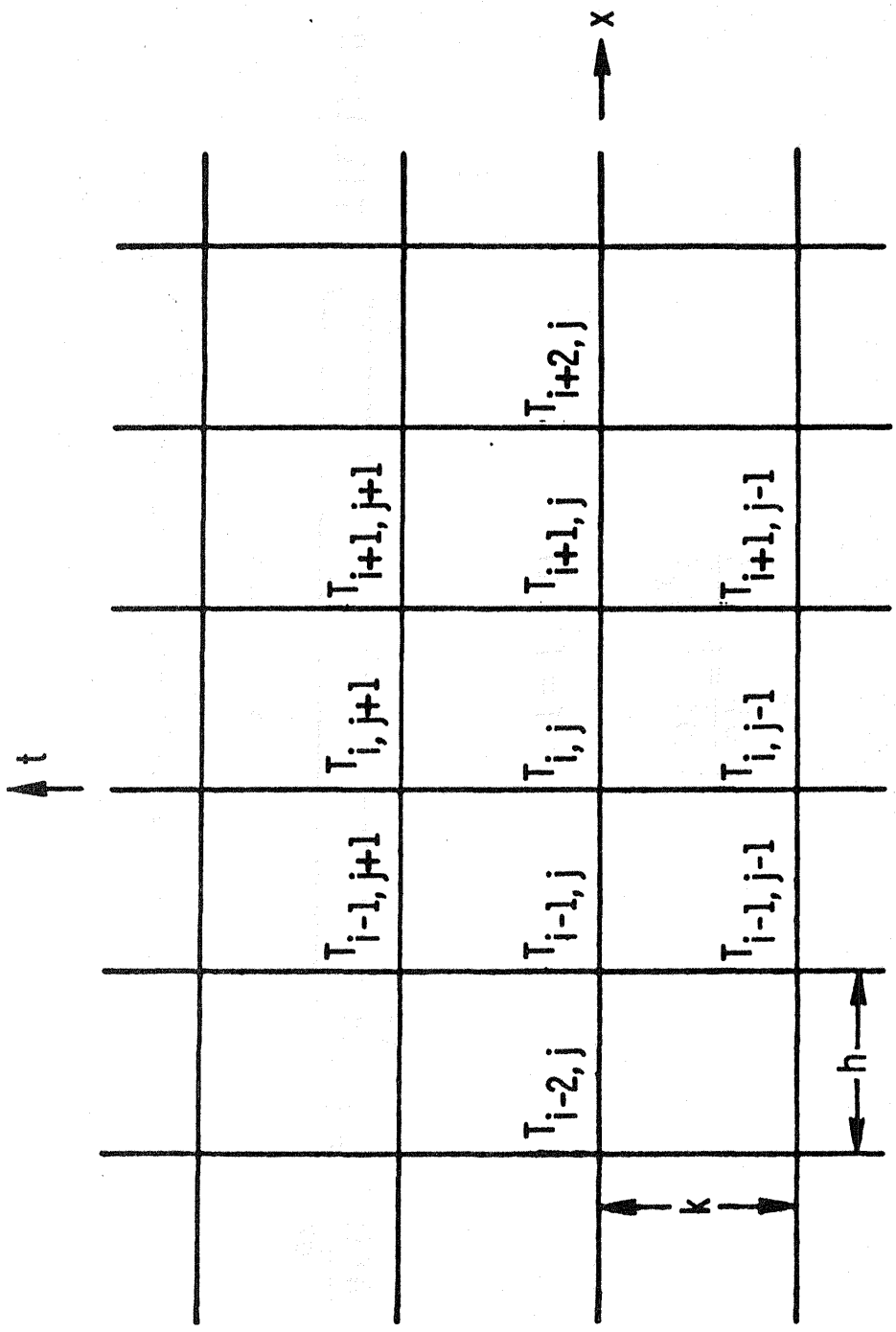


Figure 2.- Finite difference grid.

FEM HARDWARE AND SOFTWARE

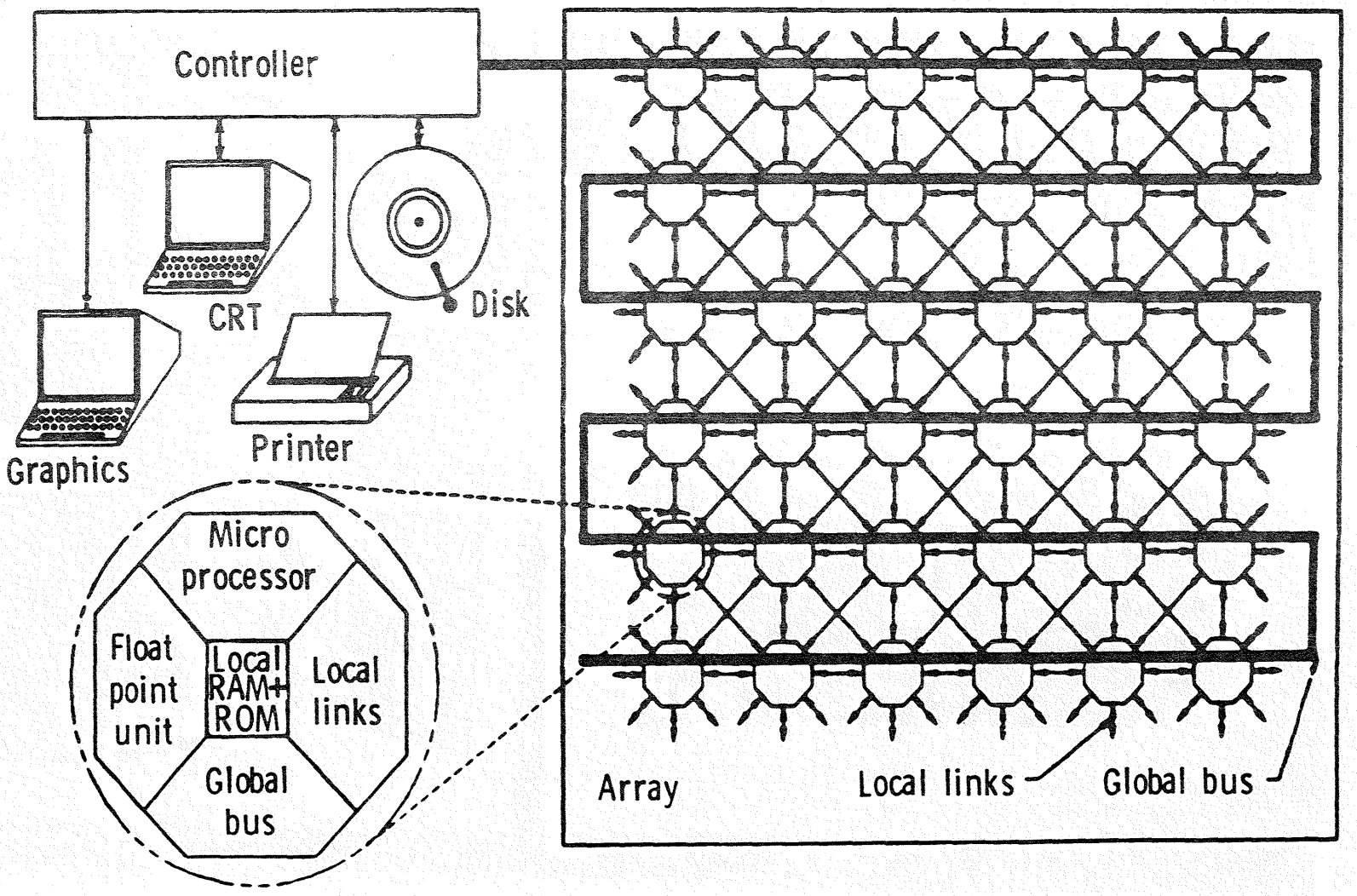


Figure 3.- FEM hardware and software block diagram.

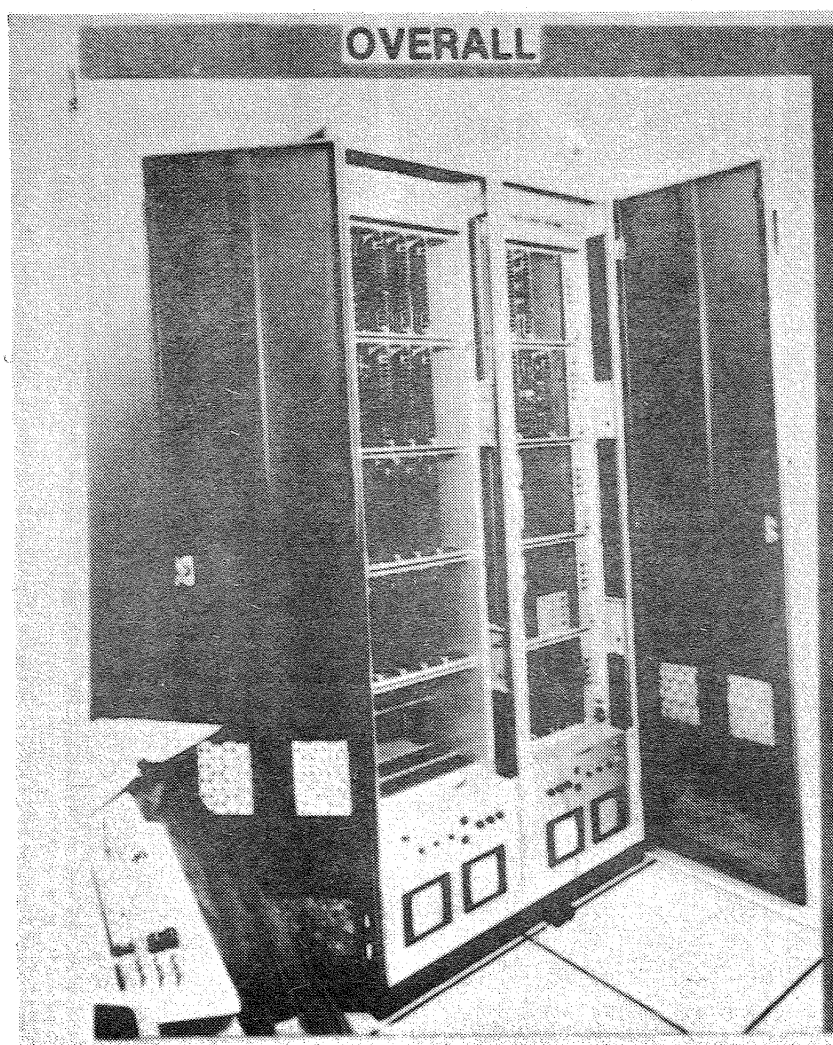


Figure 4a.- FEM system.

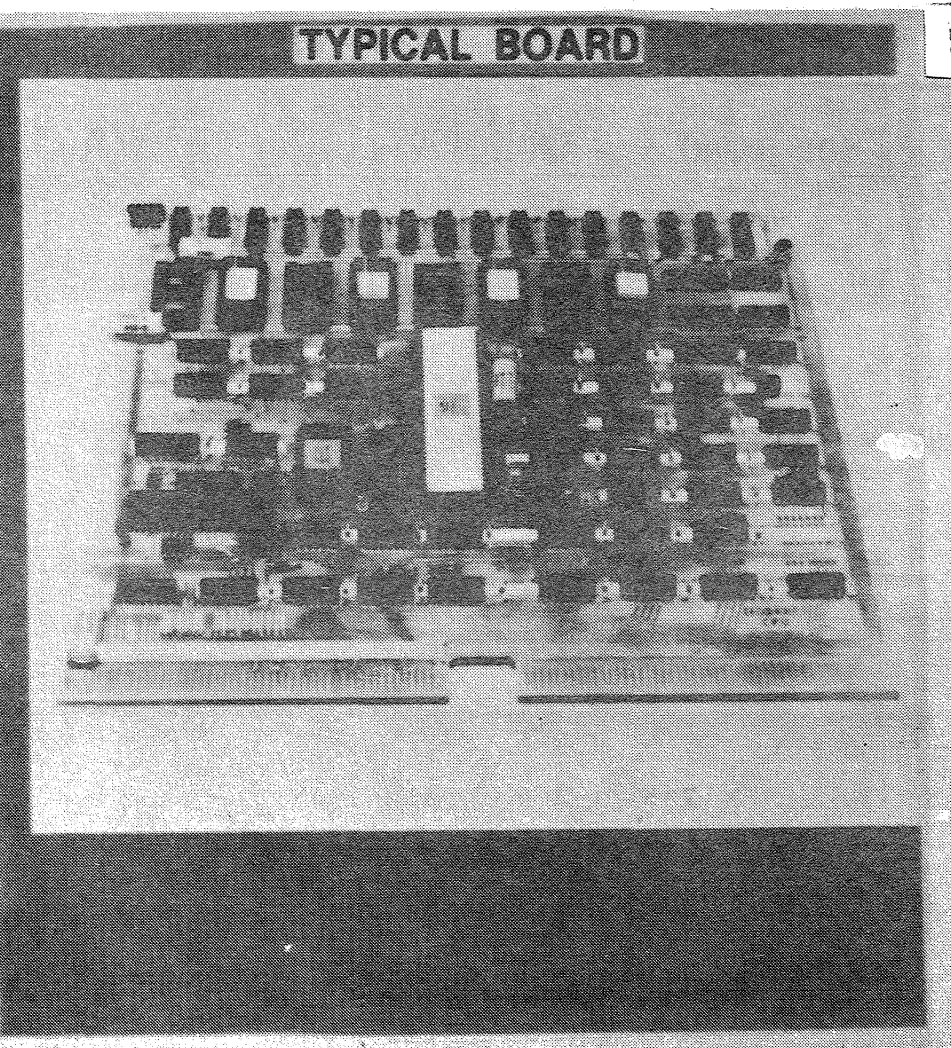


Figure 4b.- Typical board.

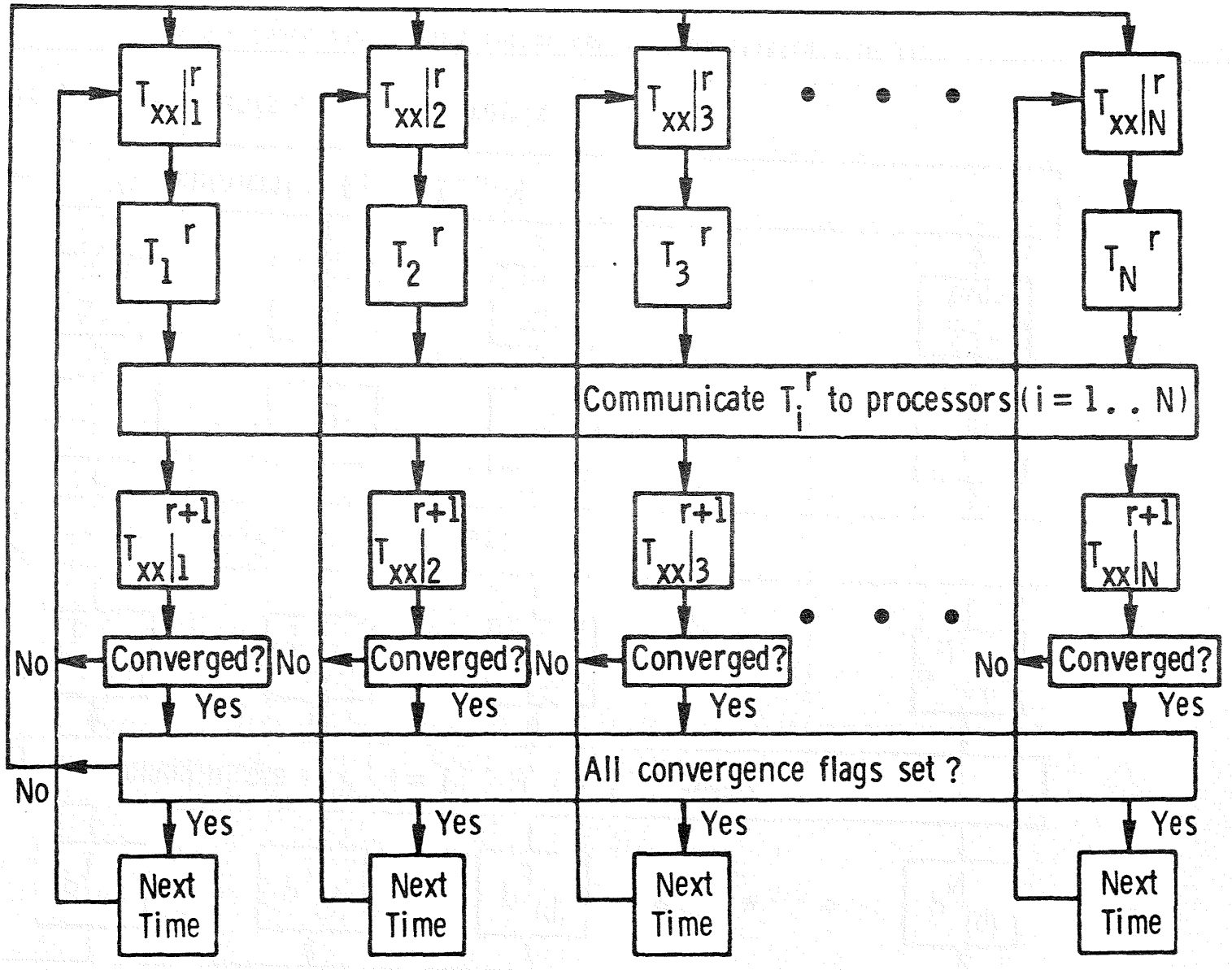
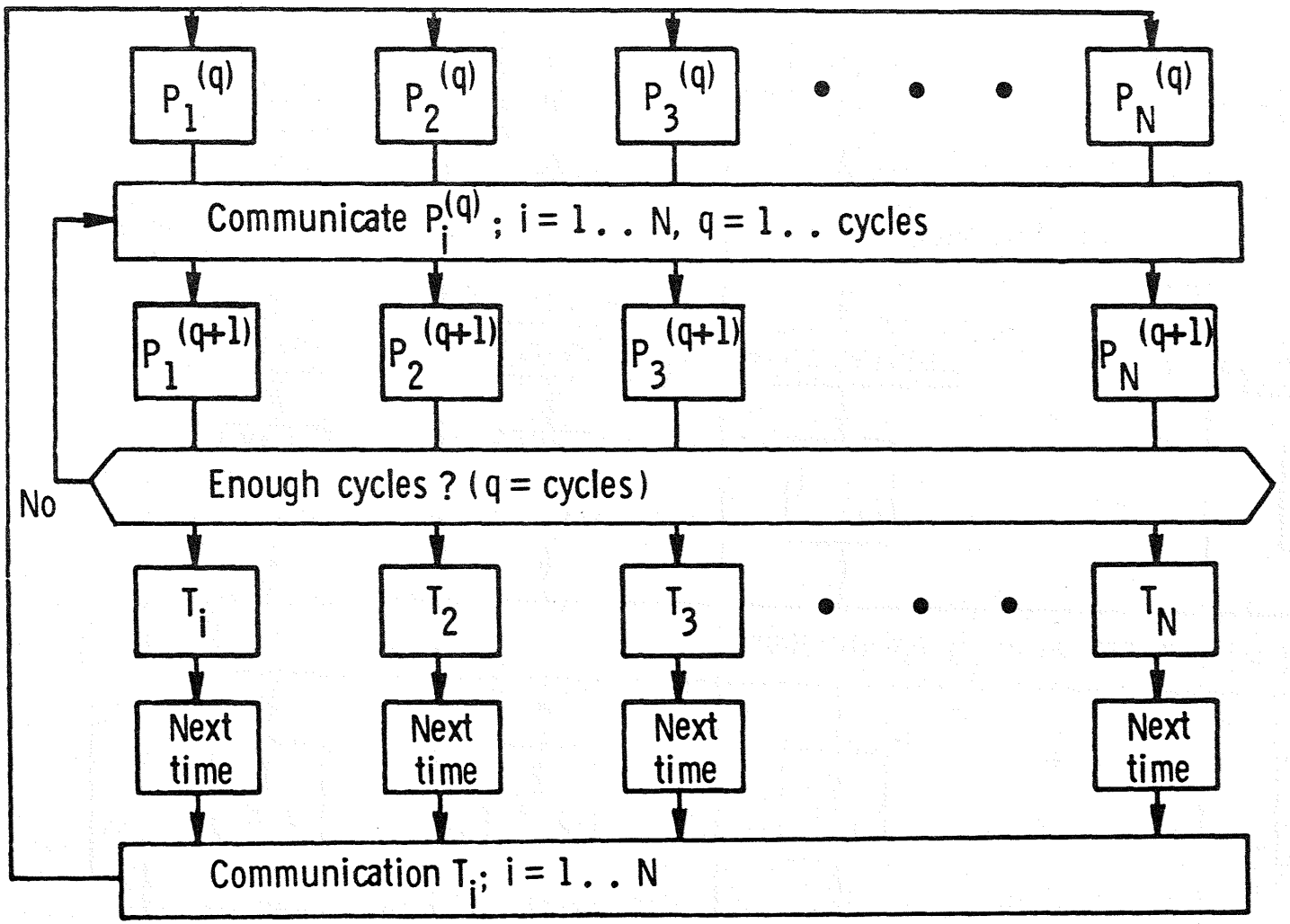


Figure 5.- Concurrent iterative method moving from station j to j+1.



P = array of coefficients of the j + 1 matrix

Figure 6.- Concurrent direct method moving from station j to j+1.

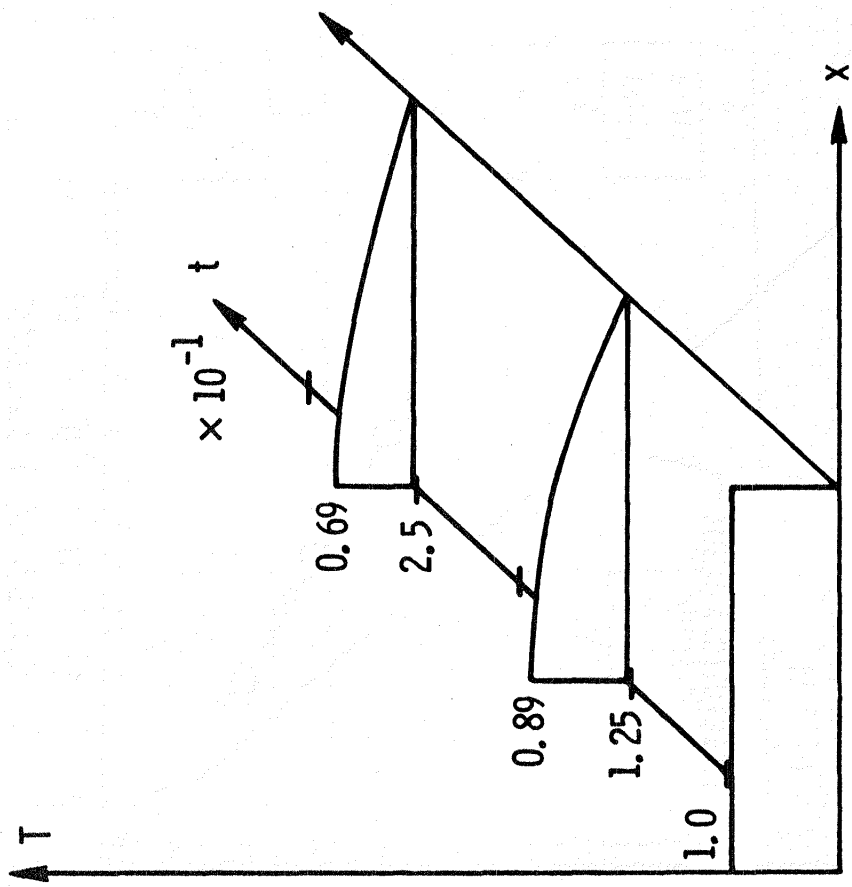


Figure 7.- Temperature distribution results.

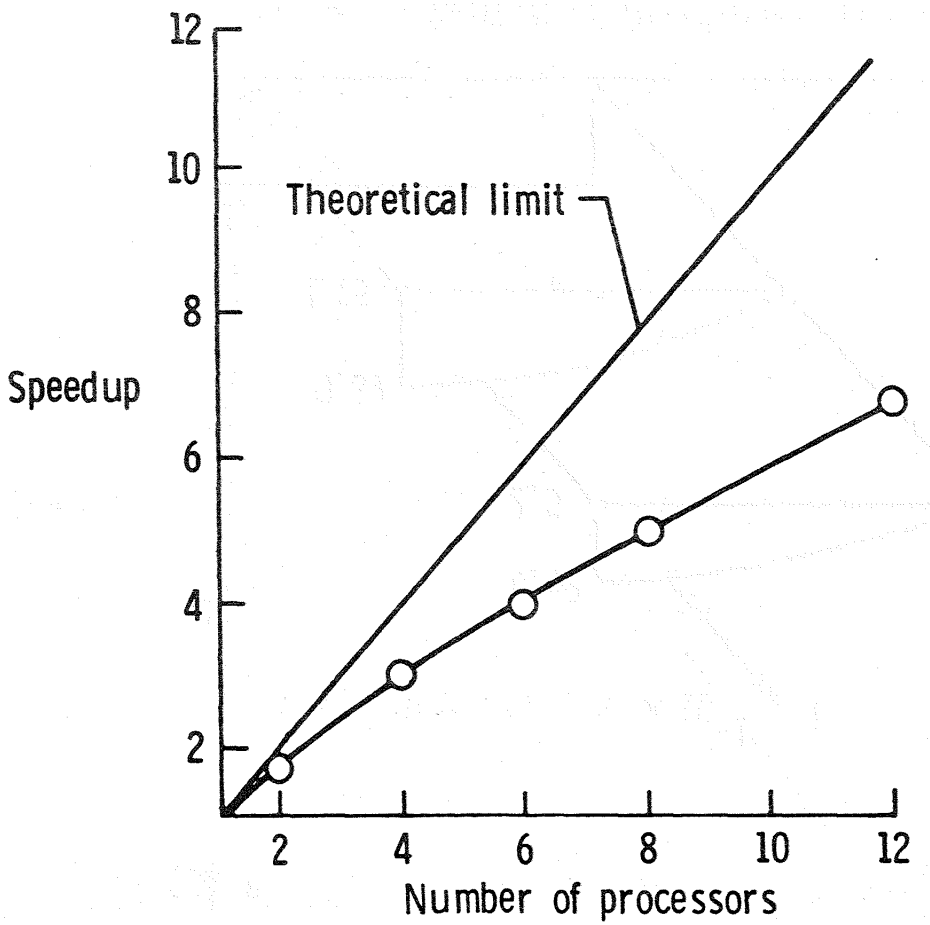


Figure 8.- Computation speedup of iterative method versus number of processors.

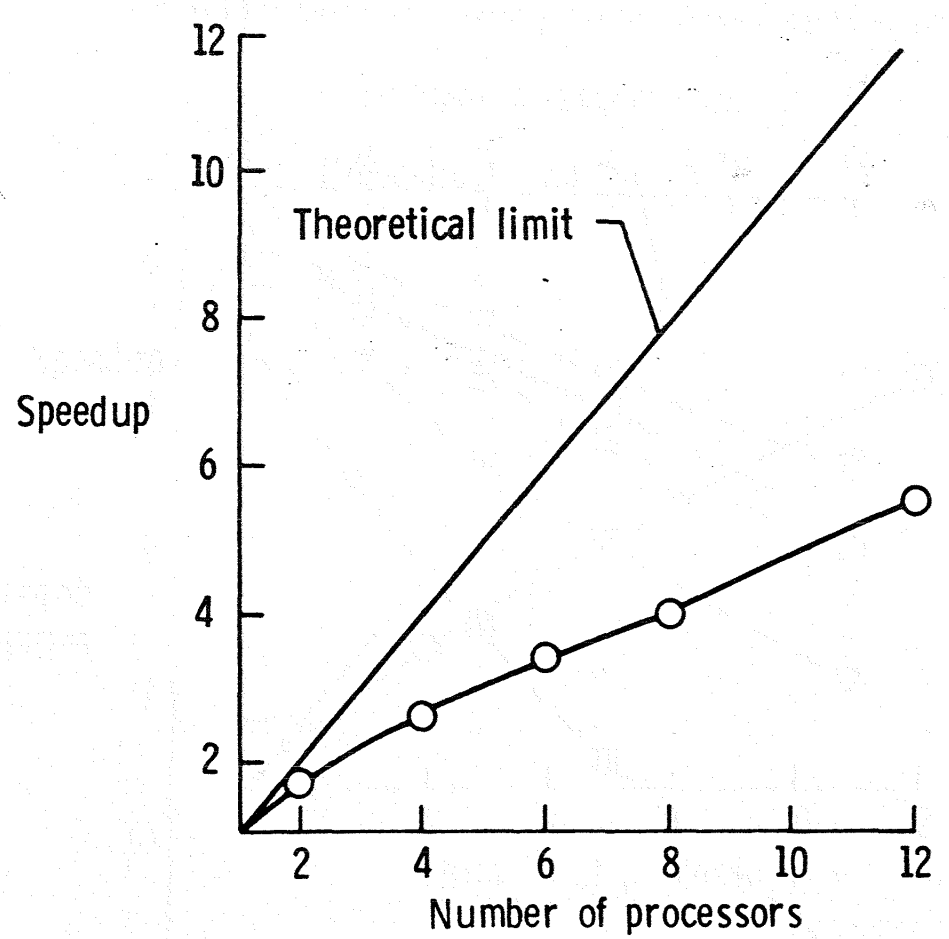


Figure 9.- Computation speedup of direct method versus number of processors.

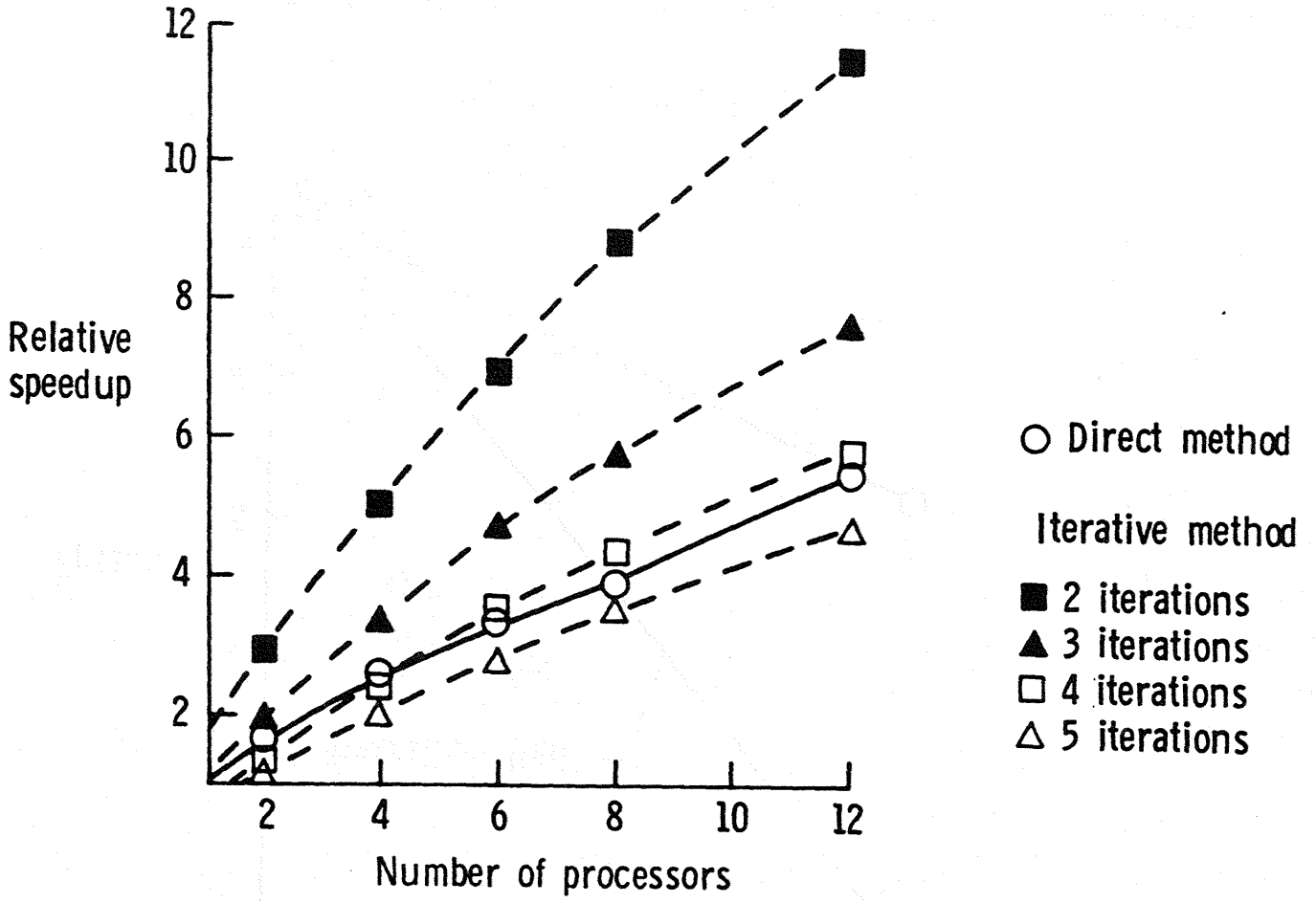


Figure 10.- Relative speedup versus number of processors.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-86448		

4. Title and Subtitle	5. Report Date
Concurrent Implementation of the Crank-Nicolson Method for Heat Transfer Analysis	June 1985
	6. Performing Organization Code: 505-33-53-15

7. Author(s)	8. Performing Organization Report No.
Jonathan B. Ransom and Robert E. Fulton	
	10. Work Unit No.
9. Performing Organization Name and Address: NASA Langley Research Center, Hampton, VA 23665	11. Contract or Grant No.
	13. Type of Report and Period Covered: Technical Memorandum
12. Sponsoring Agency Name and Address: National Aeronautics and Space Administration, Washington, DC 20546	14. Sponsoring Agency Code

15. Supplementary Notes

Jonathan B. Ransom, NASA Langley Research Center, Hampton, Virginia.
Robert E. Fulton, George Washington University, NASA Langley Research Center, Hampton, Virginia.

16. Abstract

To exploit the significant gains in computing speed provided by Multiple Instruction Multiple Data (MIMD) computers, concurrent methods for practical problems need to be investigated and test problems implemented on actual hardware. One such problem class is heat transfer analysis which is important in many aerospace applications. This paper compares the efficiency of two alternate implementations of heat transfer analysis on an experimental MIMD computer called the Finite Element Machine (FEM). The implicit Crank-Nicolson method is used to solve concurrently the heat transfer equations by both iterative and direct methods. Comparison of actual timing results achieved for the two methods and their significance relative to more complex problems are discussed.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Finite Element Machine (FEM) concurrent processing	Unclassified - Unlimited Subject Category 62

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	27	A03

For sale by the National Technical Information Service, Springfield, Virginia 22161

End of Document