111-39
264842
37P

NASA Contractor Report 185199

Transient Finite Element Computations on the Transputer System

Patrick J. Smolinski
University of Pittsburgh
Pittsburgh, Pennsylvania

February 1990

Prepared for
Lewis Research Center
Under Grant NAG3-829

NASA
National Aeronautics and
Space Administration

FOREWORD

The author would like to thank D. Janetzke and J. Kiraly of the Structural Dynamics Branch of the NASA Lewis Research Center for their helpful discussions and assistance in the use of the Transputer system.

TABLE OF CONTENTS

SUMMARY

The aim of this project was to study the solution of transient finite element problems on the Transputer system of parallel processors. The central difference time integration rule was used so that no equation solving was necessary. Also investigated was subcycling time integration which uses different time steps in different subdomains of the finite element mesh.

A one dimensional bar problem was analyzed using the parallel time integration algorithm. This involves subdividing the bar into subproblems which are assigned to different processors. Results show that the significant speed-up can be obtained through parallel processing. Also subcycling can give an additional speed-up in certain classes of problems. A two-dimensional problem was also examined to evaluate the effect of the communication to computation ratio on solution time.

1. INTRODUCTION

Even with the impressive speed of the current generation of computers many problems in computational mechanics still remain untractable. For instance, problems involving nonlinear three-dimensional transient analysis or multidisciplinary optimization problems represent such tremendous computational burdens that they cannot be readily solved with even today's supercomputers. In an effort to create substantially faster computers attention has focused on the development of multiprocessor or parallel computers and to date several models are already being marketed. However, to fully exploit the potential of these new machines special algorithms must be developed which are amenable to this type of computing [1].

In this regard, explicit time integration offers advantages for the solution of transient finite element equations because of the fact that it does not require the solution of a set of equations to perform the update. For this reason it can be easily parallelized, by partitioning the nodes of the mesh into groups which are to be updated concurrently over a time step. However, the major disadvantage of explicit integration is that it is only conditionally stable and the time step must be less than a critical value for a meaningful solution. This restriction can impose an excessive number of time steps in some problems. It should be noted, however, that in certain classes of problems unconditionally stable methods can require as small a time step as explicit methods in order to achieve sufficient accuracy [2]. One way that has been devised to partially overcome this difficulty is through the use of subcycling [3-5].

Subcycling involves the use of different time steps in different subdomains of the mesh. In this way a group of small or stiff elements which normally impose a small time step on the entire mesh can be integrated with a small time step while the remainder of the mesh is updated with a larger time step. Subcycling methods have been used for both heat conduction [5] and structural dynamics problems [4] and also with implicit integration schemes [3].

Currently most of the work done on parallel computers for the analysis of finite element problems has been conducted on shared memory machines which contain a small number of large scale processors [6]. These types of parallel computers are often referred to as "coarse grained computers". With shared memory computers all processors have access to all the data stored in memory which simplifies the programming. However, memory contention problems can develop when several processors try to access the same data simultaneously. The other approach to parallel computations is the so called "fine grained" machines which are composed of many small processors each having some memory. An example of this is the hypercube machine which has been used by several investigators for the solution of structural mechanics problems [7,8]. Presented in this paper are the details and results of implementing a standard explicit time integration program and explicit subcycling program on a system of transputer processors [9]. The transputer is a small chip level processor with local memory that can be linked to other transputers to form a system and interfaced with a personal computer. The modularity of the processors allows the system to be easily expanded and the configuration of the processors may be changed to suit different classes of problems.

To evaluate the efficiency of the program a simple structural dynamics problem has been considered with different combinations of mesh sizes, number of time steps, and number of processors. Also a comparison is made between standard and subcycling time integration methods.

2. GOVERNING EQUATIONS

The finite element equilibrium equations governing structural dynamics problems in which damping is negligable are given by

$$\underset{\sim}{M}\ddot{\underset{\sim}{d}} + \underset{\sim}{f} = \underset{\sim}{F} \tag{1}$$

where $\underset{\sim}{M}$ is the mass matrix, which is assumed to be lumped, $\underset{\sim}{d}$ is the vector of nodal displacements, and $\underset{\sim}{f}$ and $\underset{\sim}{F}$ are the nodal vectors of internal and external forces, respectively. Superposed dots are used to represent time derivatives. For the case of linear systems, the internal forces are directly related to the nodal displacements and Eq. (1) can be written as

$$\underset{\sim}{M}\ddot{\underset{\sim}{d}} + \underset{\sim}{K}\underset{\sim}{d} = \underset{\sim}{F} \tag{2}$$

where $\underset{\sim}{K}$ is the stiffness matrix. The matrices $\underset{\sim}{K}$ and $\underset{\sim}{M}$ are symmetric with $\underset{\sim}{M}$ being positive definite and $\underset{\sim}{K}$ being positive semidefinite. The initial value problem consists of solving either equation (1) or (2) for $\underset{\sim}{d}=\underset{\sim}{d}(t)$ subject to the initial conditions

$$\dot{\underset{\sim}{d}}(t=0) = \dot{\underset{\sim}{d}}^{0} \tag{3a}$$

$$\underset{\sim}{d}(t=0) = \underset{\sim}{d}^{0} \tag{3b}$$

for all time, $t>0$.

3. INTEGRATION ALGORITHM

Perhaps the simplest method of solving the governing equation is the central difference time integration rule which in the asynchronized form is

$$\dot{\underset{\sim}{d}}^{n+1/2} = \dot{\underset{\sim}{d}}^{n-1/2} + \Delta t \ddot{\underset{\sim}{d}}^{n} \tag{4a}$$

$$\underset{\sim}{d}^{n+1} = \underset{\sim}{d}^{n} + \Delta t \dot{\underset{\sim}{d}}^{n+1/2} \tag{4b}$$

where Δt is the time step and superscripts are used to indicate the time, for example $\underset{\sim}{d}^{n} = \underset{\sim}{d}(n\Delta t)$. This rule is referred to as an explicit method since no equations need be solved to update the solution if the mass matrix is diagonal. It is well known that the central difference method is only conditionally stable and that the time step must satisfy

$$\Delta t \leq \frac{2}{\omega_{max}} \tag{5a}$$

where ω_{max} is the maximum frequency given by the eigenvalue problem

$$\underset{\sim}{K}\underset{\sim}{x} = \omega^2 \underset{\sim}{M}\underset{\sim}{x} \tag{5b}$$

A more conservative and easily computed criterion [2] is that

$$\Delta t \leq \frac{2}{\omega^{e}_{max}} \tag{6a}$$

where ω^e_{max} is the maximum frequency of the elemental eigenvalue problem

$$\underset{\sim}{K}^e \underset{\sim}{x}^e = {\omega^e}^2 \underset{\sim}{M}^e \underset{\sim}{x}^e \tag{6b}$$

This time step restriction can preclude the use of the central difference method in certain problems. For instance, in structural dynamics problems where the low frequency modes dominate the response, the critical time step is too restrictive and accurate results can be obtained by using an unconditionally stable integration method with a much larger time step. A discussion of stability and accuracy characteristics of different integration schemes and problem applications can be found in [2].

In an effort to overcome this difficulty, subcycling time integration methods [4] have been proposed. With subcycling different time steps are used in updating different subdomains of the mesh. In this way problems with locally applied loads or problems where a group of stiff elements impose a small time step, can be integrated using a small time step in the critical region while the remainder of the mesh is updated with a much larger time step. While stability proofs exist for various subcycling schemes for the diffusion equation [3,5], to the author's knowledge a rigorous proof for hyperbolic equations has not been shown. However, a nodal interpolation method as proposed in [10] has given good results in many problems and has not displayed any signs of instability when used with the standard stability criterion, equations (6). To illustrate the method with a two subdomain problem consider the vector of nodal displacements partitioned in the form

$$\underset{\sim}{d} = \begin{Bmatrix} \underset{\sim}{d}_A \\ \underset{\sim}{d}_B \end{Bmatrix} \tag{7}$$

where the nodal A and B are integrated with the time steps Δt and $m\Delta t$, respectively and m is the integer time step ratio. For this purpose, equations (4) are rewritten in the form

$$\dot{\underset{\sim}{d}}^{n+1/2} = \dot{\underset{\sim}{d}}^{n-1/2} + \Delta t \underset{\sim}{W} \ddot{\underset{\sim}{d}}^{n} \tag{8a}$$

$$\underset{\sim}{d}^{n+1} = \underset{\sim}{d}^{n} + \Delta t \dot{\underset{\sim}{d}}^{n+1/2} \tag{8b}$$

where

$$\underset{\sim}{W} = \begin{bmatrix} \underset{\sim}{I} & \underset{\sim}{0} \\ \underset{\sim}{0} & m\underset{\sim}{I} \end{bmatrix} \text{ if mod } (n,m) = 0 \tag{9a}$$

$$\underset{\sim}{W} = \begin{bmatrix} \underset{\sim}{I} & \underset{\sim}{0} \\ \underset{\sim}{0} & \underset{\sim}{0} \end{bmatrix} \text{ otherwise} \tag{9b}$$

In the above equations $\underset{\sim}{I}$ is the unit matrix and the $\underset{\sim}{W}$ matrices are partitioned similar to $\underset{\sim}{d}$. In this scheme a total cycle, which advances the solution from t to t + $m\Delta t$, consists of one update using equations (8) and (9a) and m-1 subcycles using equations (8) and (9b). Shown in Fig. 1 is the flow of information in a one dimensional problem with m=4. This figure illustrates that the intermediate values of the displacement for the nodes integrated with the large time step are calculated using linear interpolation. Although analytical stability criterion have not yet been derived for this subcycling method, equation (6a) is used in determining the appropriate time step for a node. The critical time step for a node is defined as the minimum critical time step among all the elements connected to the node. This procedure allows the nodal time steps to be easily chosen for practical problems and has not given rise to any instabilities.

It should be pointed out that subcycling can be easily implemented into a standard explicit program with little change to coding. However, several additional arrays are needed. One array is required to keep track of time step size for each node and an array is necessary for each nodal group. The nodal group arrays contain the numbers of the elements attached to the nodes in the group. These arrays give the elements to use when computing the internal forces for a nodal group. A flowchart for the subcycling algorithm is given in Table 1.

4. PARALLEL IMPLEMENTATION

The basic structure of explicit time integration allows the displacement at a node to be updated independently of other nodes over a time step. Information from other nodes enter into the calculation only through historical terms. This fact allows the nodes of the mesh to be partitioned into subdomains which can be updated by different processors with information being exchanged after every time step. In order to minimize the amount of data that must be exchanged, the nodal groups should be composed of contiguous nodes. This way only the displacements from the boundary of the subdomain must be transferred. This concept is illustrated for a one-dimensional bar in Fig. 2.

The parallel computer used in this study is a Multiple-Instruction Multiple-Data(MIMD) machine composed of a system of transputer processors. Each transputer processor is a VLSI design which contains an INMOS processor, on chip memory, and four bidirectional communication links. Since each processor has only local memory, data to be shared between processors must be explicitly transmitted across the communication channels. The communication links are self synchronizing so that if one processor transmits data to another it must wait until the data is received by the other processor. Also a processor waiting to receive data must wait until it is transmitted before it can continue computing. The four links can simultaneously transmit data at a rate of 10 MBit per second with about a 2μ sec. start up time.

In this study two different types of transputers were used. The INMOS T414 and has 2 KBytes of local memory and a floating point performance of about 0.1 MFLOPS while the INMOS T800 has 4 KBytes of memory and is capble of about 1.2 MFLOPS. Although the processors may be assembled in a number of configuations, for example a torus, a 2-D mesh, or a hypercube, a pipeline configuration, Fig. 3, was chosen to simplify the programming of interprocessor communication. The computer programs were written using the occam language [11] which is specifically designed for the transputer and for the programming of interprocessor communication and parallel algorithms.

The computer system is composed of two types of processors. The transputer development system (TDS) processor is housed inside the personal computer and acts as an interface with the PC and is also used to edit, compile, and distribute the occam programs to the external processors. The external processors are used for computations alone.

In the explicit time integration scheme, the TDS processor serves as a manager of the system of external processors. At the start of the program, the master defines the problem parameters, such as the nodal and element data, assigns the nodes to processors, and transmits this information to the external processors. Once all the external processors have received and stored the appropriate data for its nodal group, the update process is iterated for the number of time steps. After each time step the updated displacement data on the boundary of each nodal subdomain is exchanged with the neighboring processors.

If subcycling is used, the basic structure of the parallel algorithm does not change. However, the standard explicit update is replaced by the subcycling scheme discussed in the previous section. The simplest case of subcycling is when nodes having the same time step are assigned to a processor. Then no additional arrays are needed and the time step counter may be used to determine when it is necessary to compute the acceleration.

A two-dimensional explicit algorithm was also developed in order to examine the efficiency of parallel processing when applied to more complicated problems. As opposed to the one-dimensional program which uses a pipeline processor configuration, the two-dimensional program can use arbitrary processor configurations, however, the nodes of the finite element mesh must communicate. For instance, the three nodes of an element must either be assigned to one processor or neighboring processors. The restriction of nearest neighbor communication is made in an effort to conserve the limited amount of processor memory and also to limit the amount of interprocessor communication which significantly increases the running time.

The structure of the two-dimensional program is similar to the one-dimensional version with the master processor transmitting the data of the problem to the system. However, in this case it also transmits information on the processor connections so that each processor can determine to how many and to which processors it is connected. After all the problem and connection data has been transmitted, each processor communicates with its neighbors to determine the nodal displacements it must send and receive from each neighbor after every time step. Once this information has been found, the time stepping procedures begin with each processor updating its assigned nodes. After each time step the appropriated nodal displacement information is exchanged between processors. The procedure continues until the solution has been computed for the time period of interest. A flowchart for the algorithm is given in Fig. 8.

5. NUMERICAL EXAMPLES

In many cases a paralled processing algorithm is evaluated by comparing the solution times for a problem of a fixed size solved on a single processor and on a multiprocessor system. However, it is arguable [8] that in general problem size expands to fill the available computing resources and that governing factor is the solution time not the problem size. For this reason a more appropriate test is how large a problem can be solved in a given time. This can be examined by choosing a variable size problem in which the amount of processor work remains fixed while the number of processors is increased.

With this in mind, the test problem that has been chosen is a one-dimensional bar composed of linear displacement elements that is fixed at one end and given an initial displacement at the other end. The parameters that were varied were the number of nodes in the mesh, the number of time steps, and the number of processors used to solve the problem. The solution times using explicit integration for three different problem families, 10 elements per processor, 100 elements per processor, and 400 elements per processor, are given in Figs. 4, 5, 6, respectively.

Several conclusions can be drawn from the solution times for these various problems. If only one time step is run the solution time is greater for greater numbers of processors. This is because the problems with more processors have more elements and the data takes longer to set-up and distribute among the processors. This set-up and distribution time is a significant portion of the total solution time if one time step is computed. As the number of time steps increases, the proportion of time used in setting up the problem diminishes and the solution times for different numbers of processors converges. Another point is that if the average times for the 10 element per processor and 100 element per processor problems for 10000 time steps are compared on a per element basis, the larger problem is faster. This is due to the fact that the limiting factor in the smaller problem is the interprocessor communication after each time step and not the element computations.

The second problem that has been investigated is the subcycling time integration of a one-dimensional bar composed of different element sizes, Fig. 7. Since the material properties have been chosen so that the wave speed of the material is $C = 1$, the critical time step for standard explicit integration is $\Delta t = 0.1$. First however, to illustrate the accuracy of the subcycling algorithm a smaller bar problem with 100 elements of length $L = 1.0$, 100 elements of length $L = 0.1$, and 100 elements of length $L = 1.0$ with an applied compressive stress, Fig. 7, has been analyzed. A time step of $10\Delta t$ was used for the nodes connecting two large elements and a time step of Δt for the remaining nodes with $\Delta t = 0.095$. The stress history for subcycling and explicit time integration at various points in the bar is also given in Fig. 7.

The larger problem was run on the transputer system with different numbers of processors and the solution times are given in table 2. Note that these times should not be directly compared with the times from the first example since a T800 TDS processor was used in this example and a T414 TDS processor was used in the first problem. For this subcycling problem all the nodes that are assigned to a processor are updated with the same time step. This simplifies the programming and saves memory, but is not necessary. In all cases, the group of nodes with the small time step were assigned to one processor, while the remaining nodes were divided among the other processors. Even though the subcycling was approximately three to four times as fast as the explicit integration for a given number of processors, this speed-up is not as great as might be expected on a sequential computer. One reason for this is that balancing the work load among the processors is more difficult since two types of updates are involved. For example, if the difference in time steps is large, the majority of updates are the subcycles of the region of small elements. During the subcycle, the displacement of the nodes being integrated with the large time step is incremented with a fixed value. This can be computed very quickly. However, the update of the region of small elements takes longer because the internal forces must be computed, which delays the overall time step. Assigning more processors to update these nodes would not necessarily speed-up the computation, since the increase in communication time would probably offset any gain in computation time because this is such a small group of nodes. Alternatively, using more processors to update the large time step nodes does reduce the solution time up to a point, however, as with any fixed size problem the communication computation ratio increases which leads to diminishing returns.

The simple two-dimensional test problem chosen for study is a rectangular plate which is fixed at one end and has an applied load at the other end as shown in Fig. 9. The plate was partitioned equally by vertical lines and the nodes of the different partitions were assigned to different processors as in Fig. 9. The test problem was chosen so that the effect of varying the computation to communication ratio on the efficiency of the parallel algorithm could be examined. For this reason, the size and geometry of the problem were varied by changing the number of nodes in the x and y directions. For example, the notation 20x10 indicates that the plate is divided so that there are rows of 20 nodes, in the x direction (nsegx=20) and rows of 10 nodes in the y direction (nesgy=10). In this case the total number of nodes in the problem would be 200 (nsegx*nesgy). The number of nodes assigned to each processor is the total number of nodes in the problem divided by the number of processors. The effect of changing the number of nodes in the x and y directions is to alter the ratio of computation to communication for each processor. The amount of processor computation is proportional to the total number of nodes while the amount of communication is proportional to only the number of nodes in the y direction since the problem is partitioned vertically. The three node triangle element was used in this study because of its simplicity.

The results for the first series of test problems for various numbers of time steps are given in table 3. Here two processors were used and the problems have 100, 200, and 400 nodes, respectively. Again, similar to the one-dimensional problem, it can be seen from the data that once a minimum number of time steps are run to overcome the initial parallel overhead, the solution time is proportional to the number of time steps.

In the second problem the problem size was varied along with the number of processors so that the number of nodes per processor was kept fixed at 50. If the parallel algorithm were perfectly efficient the solution times for all three cases would be equal.

From table 4 it is noted that the solution times become closed as the number of time steps is increased and the four and eight processor problems are quite close for 1000 time steps. It should be mentioned, however, that perfect efficiency can never be achieved since the parallel program requires interprocessor communication which entails additional work.

The third problem investigates how the solution time varies as the amount of interprocessor communication is increased. This is done by keeping the number of nodes per processor, and therefore computational load, fixed while varying the numbers of nodes in the x and y directions. By the way the nodes of the problem are partitioned, the amount of interprocessor communication is proportional to length of the problem in the y direction which is 10, 20, 40, and 80 for the four cases considered. As can be seen from the data in table 5 the solution time increases with increasing amounts of interprocessor communication. However, the solution time is not proportional to the amount of interprocessor communication. Comparing the solution times for 1000 time steps, there is only a slight increase as the amont of communication is doubled. One reason for this is that the total solution time is composed of communication time and computation time and in this problem the computation time is kept fixed. Secondly, when communication takes place between processors, a significant amount of time is necessary to set up the exchange regardless of the amount of data that is transferred.

In summary, the explicit structure of the central difference time integration method is well suited to parallel processing because of the ease of load balancing and the minimal communication requirements. However, to use the parallel processor most efficiently the problems to be analyzed should be large enough and structured so that processor communication time to computation time is minimized. Subcycling can be used to minimize the drawback of conditional stability in certain problems with nonuniform meshes. Although load balancing is more difficult with subcycling, substantial speed-ups over single time step explicit integration can still be achieved.

6. <u>CONCLUSIONS</u>

The aim of this project was to provide an introductory study of the use of a transputer processor system in the solution of transient finite element problems. The central difference time integration method was used since it is well suited to parallel processing because of its explicit nature which requires no equation solving. Moreover, the subcycling form of the central difference method can be used to minimize the drawback of conditional stability in problems with nonuniform finite element meshes. These methods have been implemented on a transputer system of processors for both one and two dimensional test problems.

The results of the test problems have shown that to use the parallel processing environment most effectively the problems to be analyzed should be large enough and structured so that the processor communication time to computation time is minimized. For one dimensional problems this is easily accomplished by increasing the size of the mesh. However, large scale one dimensional problems are of limited practical value. For two dimensional problems the amount of interprocessor communication is proportional to the number of nodes on the boundary of the spatial domain assigned to a processor, while the amount of computation is proportional to the total number of nodes. If this spatial domain can be roughly approximated by a square, the amount of communication is proportional to the length of a side while the amount of computation is proportional to the square of the length. For this reason, at least in theory, the size of a problem can be increased so that the amount of computation is much greater than the amount of communication at which point it becomes faster to solve the problem on a multi-processor computer. However, in practice the amount of local memory on the transputer is limited so that this point may not be obtainable.

Some of the key questions to be investigated in the future for multi-dimensional problems are: (1) Given the geometry of the problem, how to partition the problem for processor assignment and what is the best processor configuration to minimize the communication. (2) How many processors should be used to solve a particular problem. With parallel processing, as with all computational methods, the ultimate goal is to minimize the solution time for a problem. For small problems this may mean the use of only one processor of a system leaving the other processors idle. For this reason, any comparisons between methods should be made on the basis of solution time not efficiency.

7. REFERENCES

1. M. Ortiz, B. Nour-Omid, and E. Sotelino, "Accuracy of a Class of Concurrent Algorithms, for Transient Finite Element Analysis," Int. J. Numer. Methods Eng., 26(2), 379-391, (1988).

2. T. Belytschko and T.J.R. Hughes, Eds. Computational Methods for Transient Analysis, North-Holland, Amsterdam, 1983.

3. T. Belytschko, P. Smolinski, and W.K. Liu, "Stability of Multi-Time Step Partitioned Integrators for First Order Finite Element Systems," Comp. Meth. Appl. Eng., 49(3), 281-297, (1985).

4. T. Belytschko, H.J. Yen, and R. Mullen, "Mixed Methods for Time Integration," Comp. Methods Appl. Mech. Eng., 17/18, 259-275, 1979.

5. P. Smolinski, T. Belytschko, and M. Neal, "Multi-Time-Step Integration Using Nodal Partitioning," Int. J. Numer. Methods Eng., 26(2), 349-359, (1988).

6. T. Belytschko and N. Gilbertson, "Concurrent and Vectorized Mixed Time, Explicit Nonlinear Structural Dynamics Algorithms" in: Parallel Computations and Their Impact on Mechanics, ed. A.K. Noor, AMD-Vol. 86, 279-287, (1987).

7. B. Nour-Omid and K.C. Park, "Solving Structural Mechanics Problems on the Caltech HYPERCUBE Machine," Comp. Methods Appl. Mech. Eng., 61, 161-176, (1987).

8. J.L. Gustafson, G.R. Montry, and R.E. Benner, "Development of Parallel Methods for a 1024-Processor HYPERCUBE," SIAM Journal on Scientific and Statistical Computing, 9(4), (1988).

9. Transputer Reference Manual, INMOS Corporation, Colorado Springs, 1986.

10. T. Belytschko, "Partitioned and Adaptive Algorithms for Explicit Time Integration," in: Nonlinear Finite Element Analysis in Structural Mechanics, by W. Wunderlich et.al., eds, (Springer, Berlin, 1981), 572-584.

11. C.A.R. Hoare, Ed. OCCAM 2 Reference Manual, Prentice Hall International Series in Computer Science, Prentice Hall, New York, 1988.

Table 1. Flow chart for nodal subcycling method.

0.) Given $\mathbf{d}^0$ and $\mathbf{v}^0$; Set t=0 and n=0

1.) Compute $\mathbf{M}$

2.) LOOP i = 1 to number nodal groups

2a.) If mod $(n/m_i) = 0$ then

2b.) Compute $\mathbf{f}_i^n$ and $\mathbf{F}_i^n$ for nodal group i

2c.) Compute $\mathbf{a}_i^n = \mathbf{M}_i^{-1}(\mathbf{F}_i^n - \mathbf{f}_i^n)$

2d.) Update $\mathbf{v}^{n+1/2} = \mathbf{v}^{n-1/2} + \Delta t \mathbf{W} \mathbf{a}_i$

3.) Update $\mathbf{d}^{n+1} = \mathbf{d}^n + \Delta t\, \mathbf{v}^{n+1/2}$

4.) Set $n = n + 1$ and $t = t + \Delta t$

5.) if $t < t_{max}$ go to 2; otherwise stop

Table 2. Parallel solution time for subcycling and explicit time integration.

Number of Processors	Solution Time (sec.) Subcycling	Explicit
3	77.83	227.01
9	19.53	76.40
13	12.83	51.24
17	9.91	39.59
19	9.14	35.63
21	8.60	32.73
25	7.93	27.42

Table 3

Parallel Solution Time for Two-dimensional Problems Using Two Processors

number of time steps	time(sec.) 10 x 10 problem	20 x 10 problem	40 x 10 problem
1	0.3	1.0	1.8
10	0.6	1.7	5.0
100	4.0	8.4	18.5
1000	37.4	75.2	153.5

Table 4

Solution Times for Two-dimensional Problems for Various Numbers of Processors Where the Number of Nodes Per Processor is Fixed

	time(sec.)		
number of time steps	2 processors 10 × 10 problem	4 processors 20 × 10 problem	8 processors 40 × 10 problem
1	0.30	0.58	1.02
10	0.64	0.99	1.43
100	3.98	5.06	5.50
1000	37.44	45.78	46.29

Table 5

Solution Times Using Eight Processors for Problems with Different Numbers of Nodes in the x and y Direction

	time(sec.)			
number of time steps	80 x 10 problem	40 x 20 problem	20 x 40 problem	10 x 80 problem
1	3.28	3.83	4.96	7.54
10	4.02	4.68	5.99	8.92
100	11.44	13.23	16.33	22.69
1000	85.64	98.73	119.71	160.37

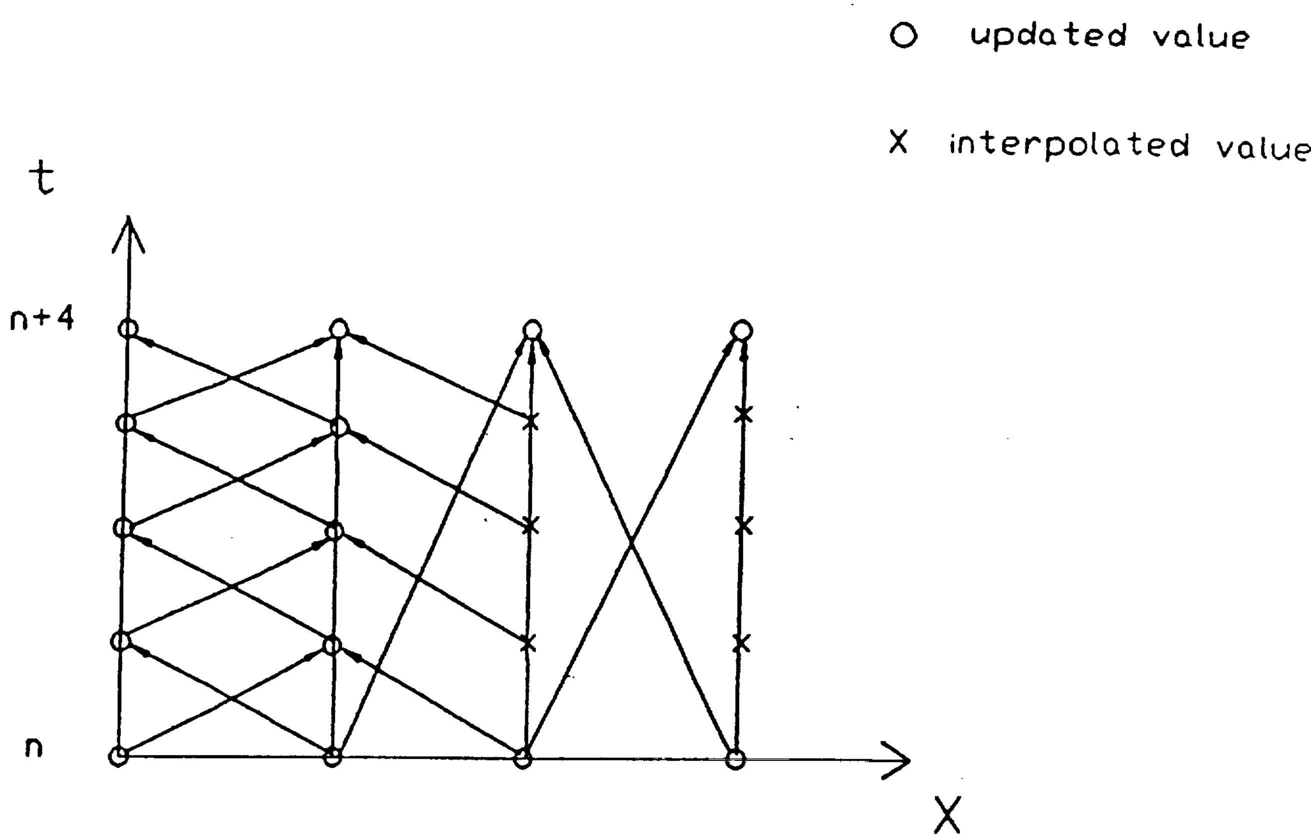

Figure 1. Flow of information in a one-dimensional problem.

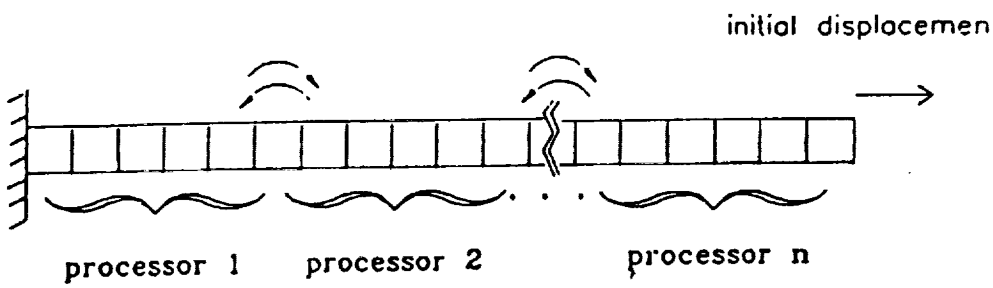

Figure 2. Bar partitioned into nodal groups.

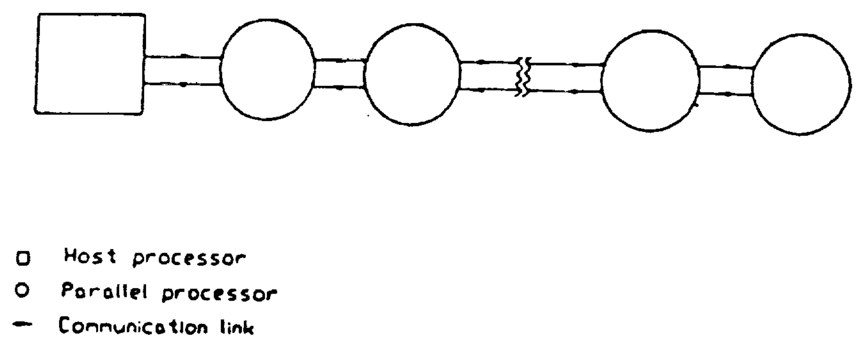

Figure 3. A pipeline processor configuration.

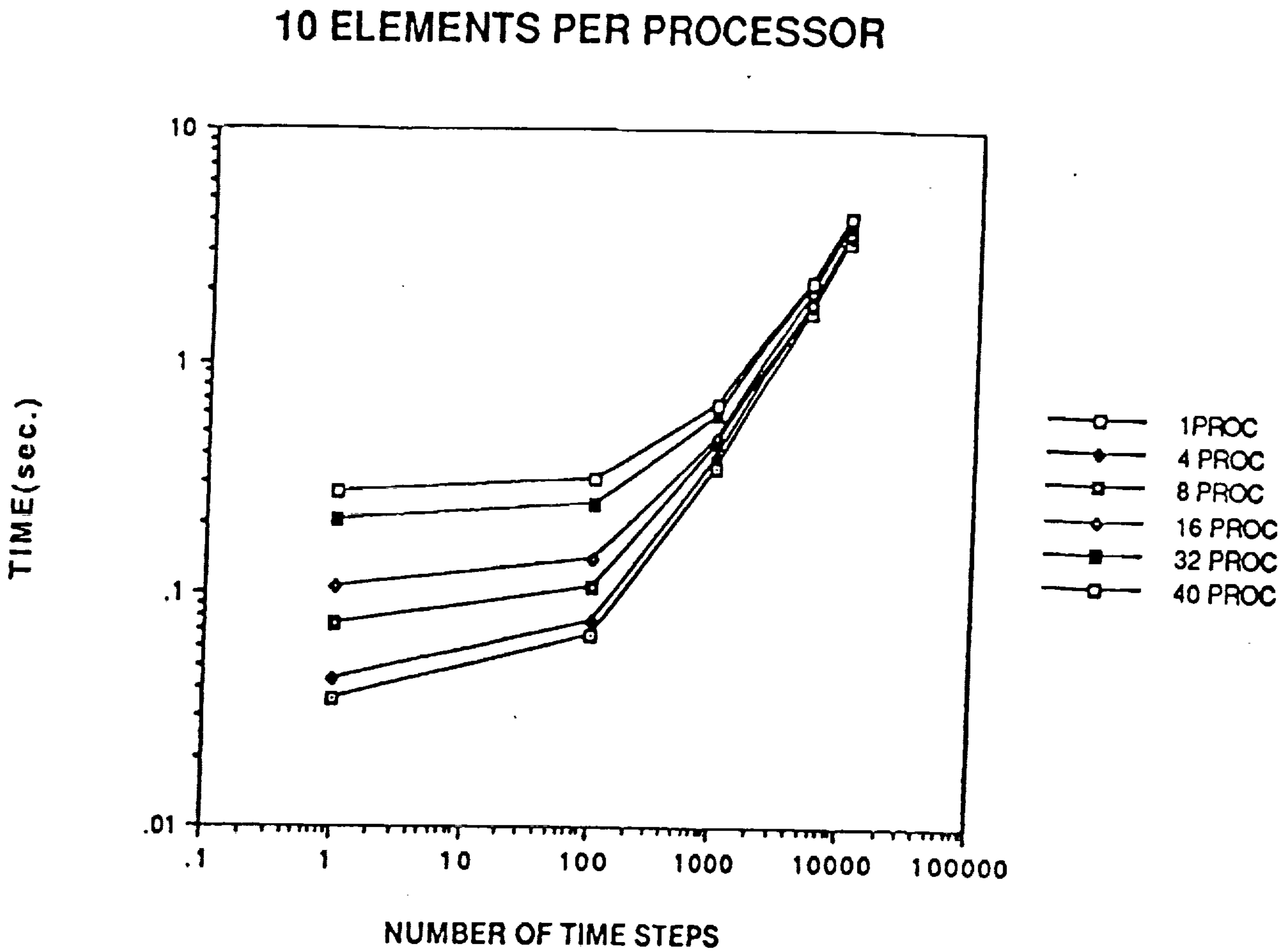

Figure 4. Solution times for 10 elements per processor.

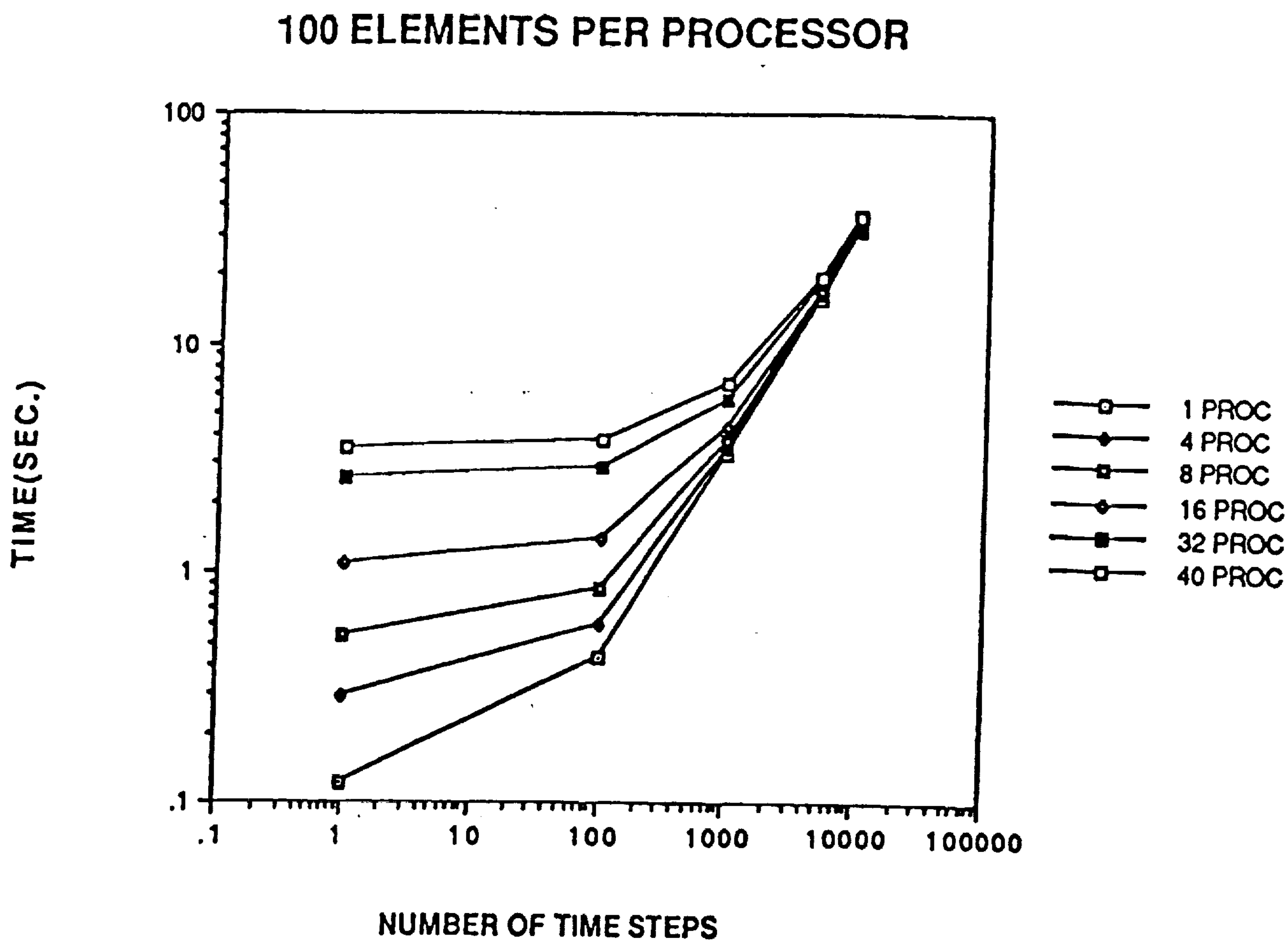

Figure 5. Solution times for 100 elements per processor.

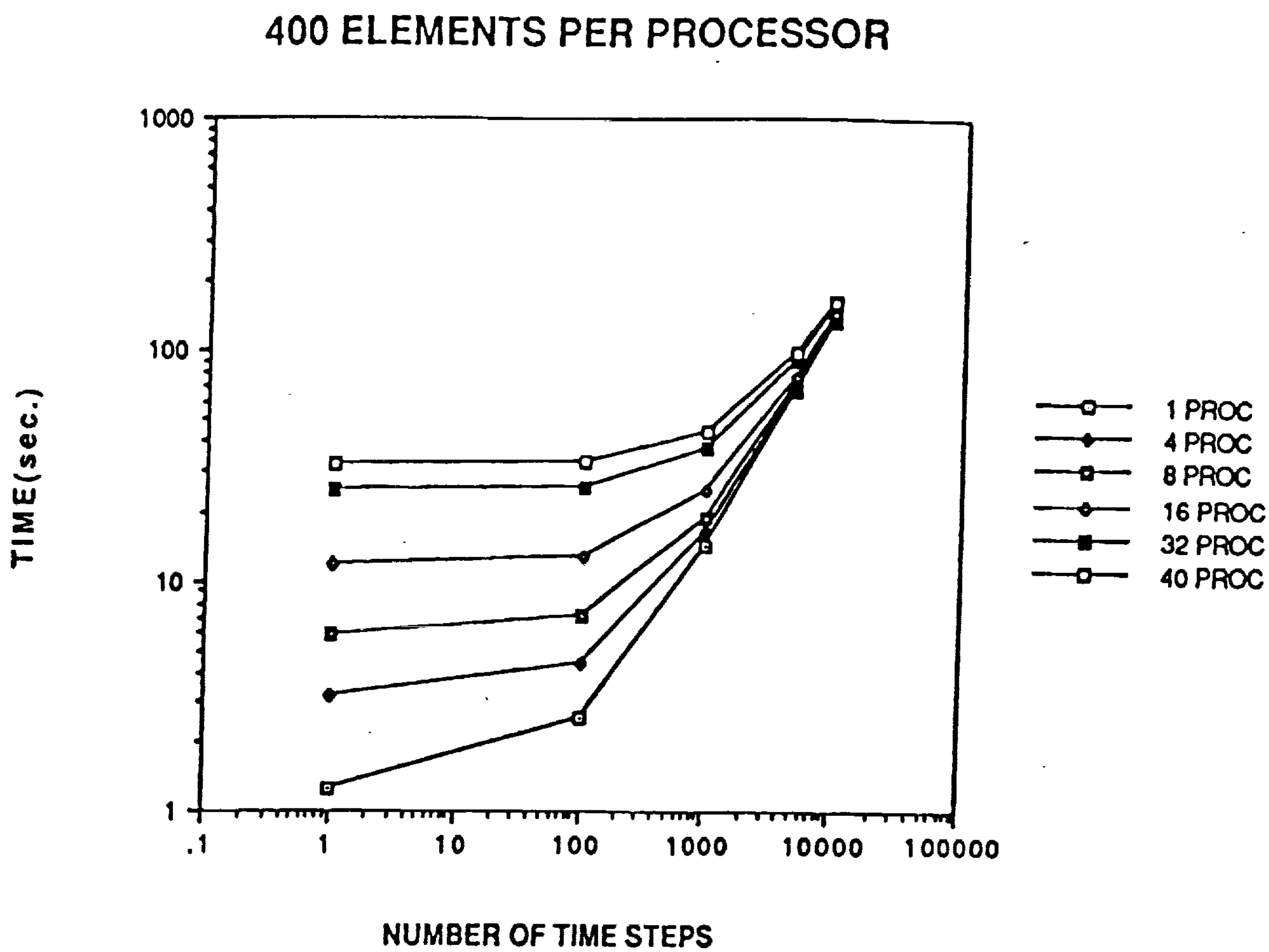

Figure 6. Solution times for 400 elements per processor.

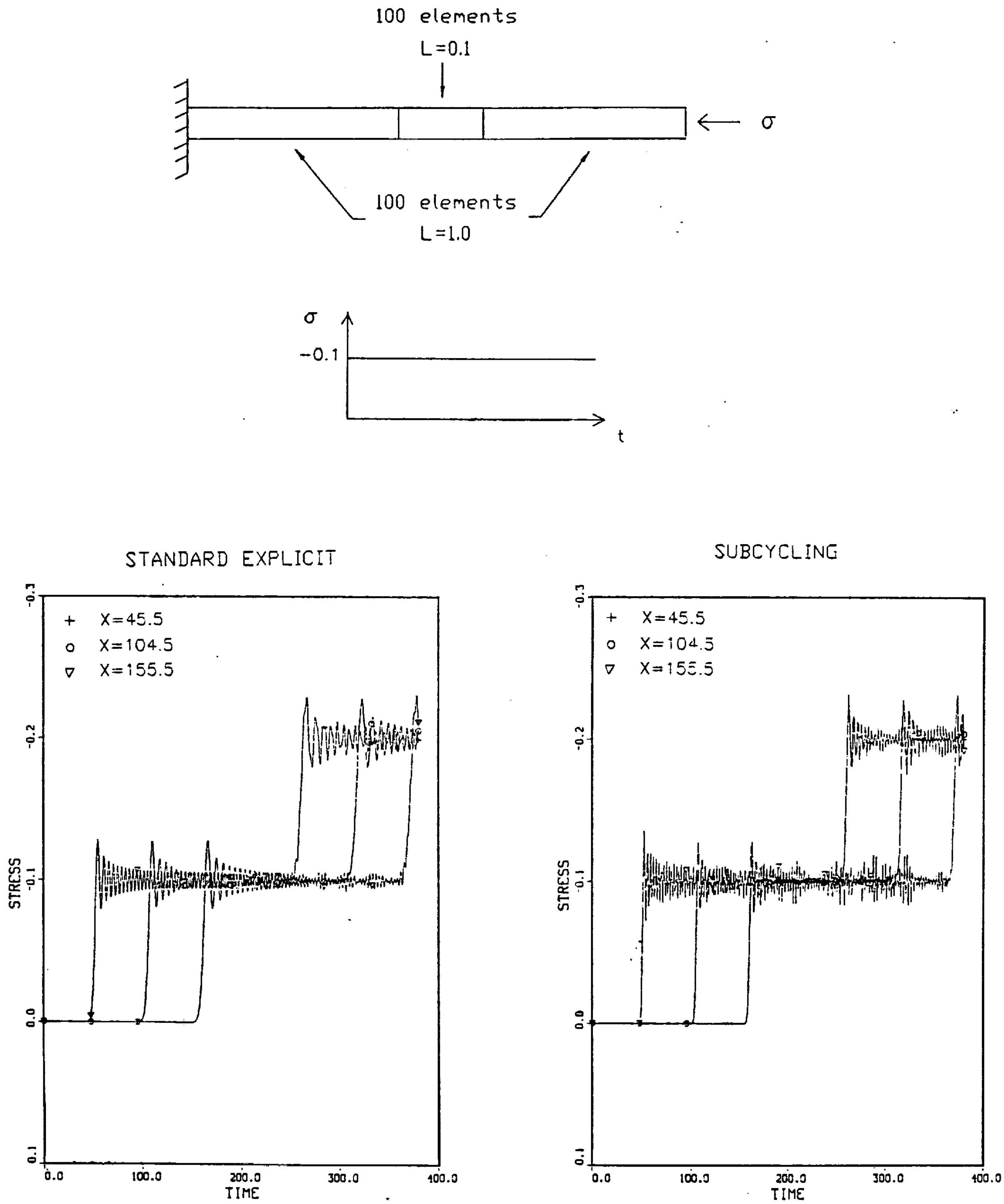

Figure 7. Problem statement and stress history for explicit and subcycling time integration.

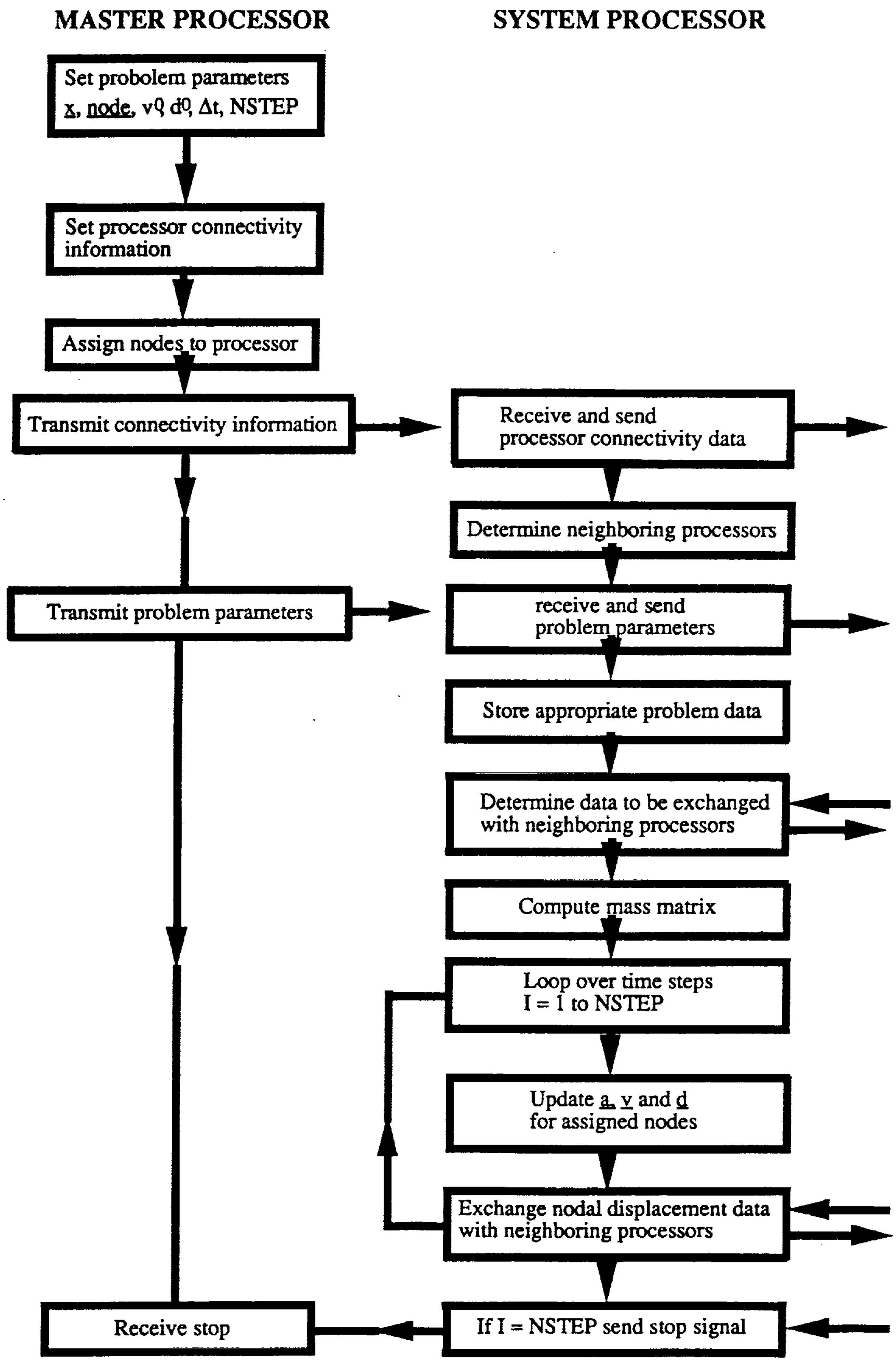

Fig. 8. Flow Chart for two-dimensional finite element example.

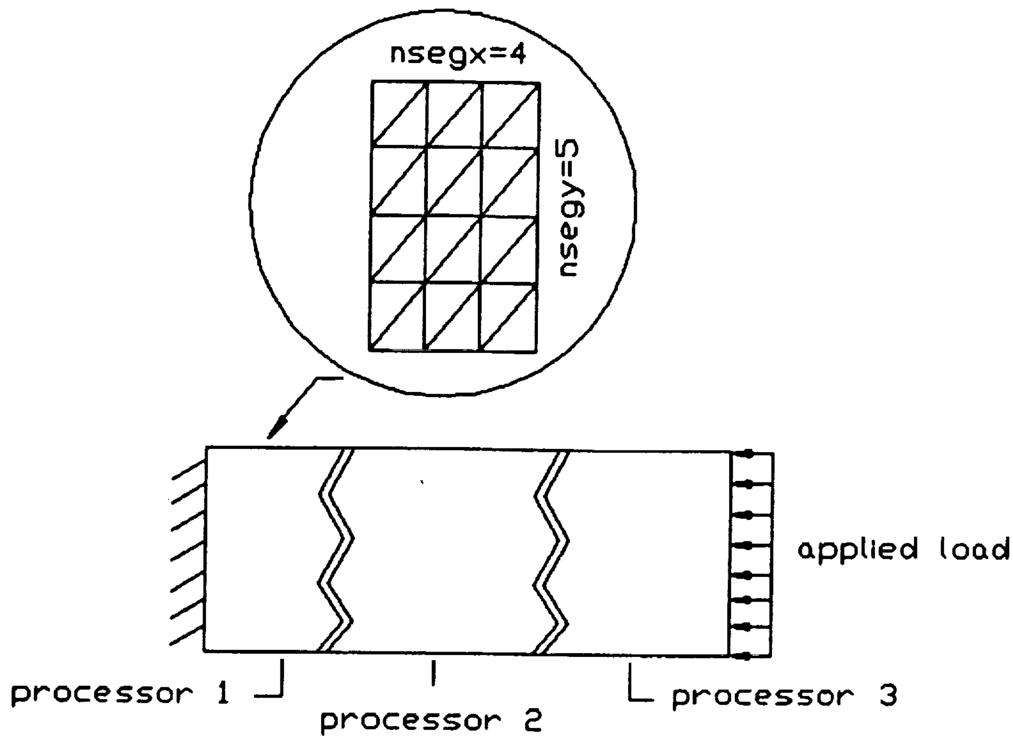

Fig. 9. Problem statement for two-dimensional finite element example.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-185199		

4. Title and Subtitle	5. Report Date
Transient Finite Element Computations on the Transputer System	February 1990
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
Patrick J. Smolinski	None
	10. Work Unit No.
	505-63-1B
9. Performing Organization Name and Address	11. Contract or Grant No.
University of Pittsburgh Department of Mechanical Engineering Pittsburgh, Pennsylvania 15261	NAG3-829
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Contractor Report Final
National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135-3191	14. Sponsoring Agency Code

15. Supplementary Notes

Project Manager, David C. Janetzke, Structures Division, NASA Lewis Research Center.

16. Abstract

The aim of this project was to study the solution of transient finite element problems on the Transputer system of parallel processors. The central difference time integration rule was used so that no equation solving was necessary. Also investigated was subcycling time integration which uses different time steps in different subdomains of the finite element mesh. A one-dimensional bar problem was analyzed using the parallel time integration algorithm. This involves subdividing the bar into subproblems which are assigned to different processors. Results show that the significant speed-up can be obtained through parallel processing. Also subcycling can give an additional speed-up in certain classes of problems. A two-dimensional problem was also examined to evaluate the effect of the communication to computation ratio on solution time.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Finite element; Subcycling; Structural dynamics; Parallel computation; Transputers	Unclassified – Unlimited Subject Category 39

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price*
Unclassified	Unclassified	37	A03

NASA FORM 1626 OCT 86

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