NASA Contractor Report 189060

Transient Finite Element Computations on a Variable Transputer System

Patrick J. Smolinski and Ireneusz Lapczyk
University of Pittsburgh
Pittsburgh, Pennsylvania

January 1993

Prepared for
Lewis Research Center
Under Grant NAG3–1152

FOREWORD

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

SUMMARY

In this study a parallel program to analyze transient finite element problems was written and implemented on a system of transputer processors. The program uses the explicit time integration algorithm which eliminates the need for equation solving making it more suitable for parallel computations. An interprocessor communication scheme was developed for arbitrary two-dimensional grid processor configurations. Several 3-D problems were analyzed on a system with a small number of processors.

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

$\underline{a}$	acceleration vector
$\underline{d}$	displacement vector
$\underline{f}$	internal force vector
$\underline{F}$	external force vector
$\underline{K}$	structure stiffness matrix
$\underline{M}$	mass matrix
next.neigh	a matrix containing numbers of the transputers to which each transputer has to send data
next.neigh.rec	a matrix containing numbers of the transputers from which each transputer has to receive data
N_{el}	num.of.elem, total number of elements in a mesh
N_{ex}	number of nodes which have to be exchanged between processors after each time step
N_{nodes}	num.of.nodes, total number of nodes in a mesh
N_{nd}	num.nodes.needed, the set of all nodes of the elements assigned to a transputer, this number is larger than N_{up}
N_{up}	num.update.nodes, number of nodes assigned to a transputer
$N_{p.el}$	num.proc.elem, number of elements assigned to a transputer
N_{step}	num.time.step, number time steps
num.neigh.rec	number of transputers from which a transputer has to receive data
num.neigh.send	number of transputers to which a transputer has to send data

p	number of processors
Δt	time step
signal.in	the matrix containing information from which direction a transputer has to receive data
signal.out	the matrix containing information in which direction a transputer has to send data
T_{cm}	communication time, the time required to exchange appropriate displacements between transputers
T_{cp}	computation time, the time required to assemble the stiffness matrix and update displacements
T_{prep}	preparation time, the time required to receive and rearrange problem parameters by transputers
T_{tot}	total execution time of a program
$\underline{v}$	nodal velocity vector

TABLE OF CONTENTS

1.0 INTRODUCTION

Today computers are widely used in the engineering field for the analysis of complex problems and to aid in the design of new components. Despite the impressive speed of the current generation of computers, there are many problems such as those involving three-dimensional analysis or multi-disciplinary optimization for which even the speed of today's supercomputers is not sufficient. Also, the solution time for many large scale problems needs to be greatly reduced before they can be effectively incorporated into the engineering design process. In an attempt to achieve a major increase in speed of computers, attention has focused on the development of parallel processing computers.

With parallel computers several processing units are connected together with the idea of subdividing a given problem into separate tasks that can be performed independently on the different processors. Theoretically, this approach gives a decrease in computation time over a traditional sequential computer which performs all the tasks in a sequential fashion.

Here the application of parallel computations to the analysis of transient finite element problems will be investigated. Transient finite element problems are among the most computationally intensive because the time history of interest must be divided into small steps and the solution to

the problem must be computed progressively at each successive step in time. These types of problems arise in the modelling of automobile crashworthiness, nuclear accidents and fluid-structure interaction in liquid storage tanks and generally involve large three-dimensional meshes and nonlinear deformations and material behavior.

The purpose of this study was to implement a three-dimensional transient finite element program on a system of transputer processors. A two-dimensional grid transputer processor configuration was chosen as the most appropriate for the finite element problems of interest in this study. In conjunction with the finite element program an interprocessor communication algorithm was developed that can accommodate any number of processors in an arbitrary grid configuration and can adapt the distance allowable communication to suit different problems. Some simple test problems of various sizes were evaluated on a four processor transputer system to study the effects of communication time on the efficiency of the parallel computation.

2.0 GOVERNING EQUATION

The governing finite element equations for structural dynamics problems can be written in the form

$$\underline{M}\ \underline{a} + \underline{f} = \underline{F} \qquad (2\text{-}1)$$

where

$\underline{F}$ - external force vector

$\underline{f}$ - internal force vector

$\underline{M}$ - mass matrix

$\underline{a}$ - nodal acceleration vector

For linear systems the internal force vector can be expressed by the following formula

$$\underline{f} = \underline{K}\ \underline{d} \qquad (2\text{-}2)$$

here

$\underline{K}$ - structure stiffness matrix

$\underline{d}$ - nodal displacement vector

and consequently equation (2-1) can be rewritten as

$$\underline{M}\ \underline{a} + \underline{K}\ \underline{d} = \underline{F} \qquad (2\text{-}3)$$

The initial conditions for the above equation are given by

$$\underline{d}^0 = \underline{d}(t=0) \qquad (2\text{-}4a)$$

$$\underline{v}^0 = \underline{v}(t=0) \qquad (2\text{-}4b)$$

where $\underline{v}$ is the vector of nodal velocity. Thus, the initial value problem consists of finding $\underline{d}(t)$ satisfying equations (2-3) and (2-4) for $t>0$.

Often it is necessary to analyze free vibrations of a structure in which case the vector of nodal displacement at

$t=0$ is assumed known and $\underline{F}=\underline{0}$ and $\underline{v}^0=\underline{0}$.

The derivation of these governing equations can be found in many books on the finite element method[1,2]*.

*Parenthetical references placed superior to the line of the text refer to the bibliography.

3.0 NUMERICAL INTEGRATION ALGORITHM

In this section the procedures used to solve the initial value problem will be presented. The most general method of solution is referred to as direct integration and involves dividing the time period of interest into steps and progressively computing the solution at each step in time.

Perhaps the most popular direct integration method is the Newmark-Beta method[3,4] and is given by

$$\underline{v}^{n+1} = \underline{v}^{n} + \Delta t\ [(1-\gamma)\ \underline{a}^{n} + \gamma\ \underline{a}^{n+1}] \qquad (3\text{-}1)$$

$$\underline{d}^{n+1} = \underline{d}^{n} + \Delta t\ \underline{v}^{n} + (\Delta t)^{2}\ [(\frac{1}{2} - \beta)\ \underline{a}^{n} + \beta\ \underline{a}^{n+1}] \qquad (3\text{-}2)$$

where Δt is the time step and gamma and beta are parameters that affect the stability and accuracy of the method and have the range $0 < \beta < 1/2$, $0 < \gamma < 1$. The superscript notation is used to indicate the time, for example $\underline{d}^{n}$ stands for $\underline{d}(n\Delta t)$. The most widely used variations of the Newmark-Beta formula corresponding to different combinations of γ and β are given in the Table 1. The various types of Newmark integration can be classified into two general categories: implicit and explicit depending on whether it is necessary to solve a system of linear equations to compute the updated values of the solution.

The first two methods in the table, the trapezoidal

method and the central difference method, are the most frequently used. The trapezoidal rule has been shown to be unconditionally stable in that convergence can be achieved with any time step, however, the accuracy of the method can be poor if too large a time step is used. The disadvantage of this method is that it is implicit and a system of equations needs to be solved in order to compute the next value of displacement, d^{n+1}. Consequently, this method is somewhat more difficult to implement in a parallel computation. A flow chart for this method is given in Table 2.

With $\gamma = 1/2$ and $\beta = 0$ the integration formula is called the central difference method and is an explicit method. This is because there is no need for any equation solving, provided that the mass matrix is lumped (diagonal). This method is particularly well suited for parallel computing because the displacements and velocities can be updated on different processors and only the displacements, used to compute the internal forces, need to be exchange after each time step. The disadvantage of this method is that it is only conditionally stable and the error grows exponentially in time and are meaningless if a certain critical time step is exceeded[5].

The restriction on the time step is given by

$$\Delta t \leq \frac{2}{\omega_{max}} \qquad (3\text{-}3)$$

where ω_{max} is the maximum frequency computed from the

Table 1.

Numerical integration algorithms

METHOD	γ	β
Central difference	1/2	0
Trapezoidal rule	1/2	1/4
Linear acceleration	1/2	1/6
Two step backward difference	1/2	1/2

generalized eigenvalue problem

$$K X = \omega^2 M X \tag{3-4}$$

However, it is more convenient to use more conservative condition

$$\Delta t \leq \frac{2}{\omega^e_{max}} \tag{3-5}$$

where ω^e_{max} is the maximum value of all the element frequencies. In this case the frequencies can be computed from the much smaller element eigenvalue problem

$$K^e X = \omega^2_e M^e X \tag{3-6}$$

and for many elements simple closed form solutions can be found for this problem. The flow chart for the central difference method is given in Table 3.

Table 2.

Flow chart for the trapezoidal method.

$$\left(\beta = \frac{1}{4},\ \gamma = \frac{1}{2} \right)$$

Given initial conditions: $\underline{d}^0$, $\underline{v}^0$ for t=0.

1] Compute $\underline{K}$,$\underline{M}$ and $\underline{M}^{-1}$.

2] Compute acceleration vector $\underline{a}^0$

$$\underline{a}^0 = \underline{M}^{-1} \left(\underline{F}^0 - \underline{K}\ \underline{d}^0\right)$$

3] LOOP n=0 FOR number time steps

a] compute $\hat{\underline{K}}$

$$\hat{\underline{K}} = \underline{M} + \beta\ \Delta t^2\ \underline{K}$$

b] compute $\hat{\underline{f}}^{n+1}$

$$\hat{\underline{f}}^{n+1} = \beta\ \Delta t^2\ \underline{F}^{n+1} + \underline{M}\ [\ \underline{d}^n + \Delta t\ \ \underline{v}^n + \Delta t^2(\tfrac{1}{2}-\beta)\underline{a}^n]$$

c] compute $\hat{\underline{K}}^{-1}$

d] compute $\underline{d}^{n+1}$

$$\underline{d}^{n+1} = \hat{\underline{K}}^{-1}\ \hat{\underline{f}}^{n+1}$$

e] compute $\underline{a}^{n+1}$

$$\underline{a}^{n+1} = \underline{M}^{-1}\ (\underline{F}^{n+1} - \underline{K}\ \underline{d}^{n+1})$$

f] compute $\underline{v}^{n+1}$

$$\underline{v}^{n+1} = \underline{v}^n + \Delta t\ [(1-\gamma\ )\ \underline{a}^n + \gamma\ \ \underline{a}^{n+1}]$$

g] if n = number of time step then terminate,

otherwise n=n+1 and GO TO a].

Table 3.

Flow chart for the central difference method

Given initial conditions: $\underline{d}^0$, $\underline{v}^0$ for t=0.

1] Compute mass matrix $\underline{M}$ and its inverse $\underline{M}^{-1}$, and stiffness matrix $\underline{K}$

2] Compute acceleration

$$\underline{a}^0 = \underline{M}^{-1}(\underline{F}^0 - \underline{K}\,\underline{d}^0)$$

3] LOOP n=0 FOR number of time steps

a] compute external force vector $\underline{F}^n$

b] update displacements

$$\underline{d}^{n+1} = \underline{d}^n + \Delta t \underline{v}^n + (\Delta t^2/2)\,\underline{a}^n$$

c] compute acceleration

$$\underline{a}^{n+1} = \underline{M}^{-1}(\underline{F}^{n+1} - \underline{K}\,\underline{d}^{n+1})$$

d] update velocities

$$\underline{v}^{n+1} = \underline{v}^n + (\Delta t/2)(\underline{a}^n + \underline{a}^{n+1})$$

e] if n = number of time steps then terminate, otherwise n = n+1 and GO TO a].

4.0 PARALLEL COMPUTATIONS

The development of finite element programs for parallel computers depends strongly on the type of machine that is to be used. Parallel processing computers are usually classified according to number of processors in the system or by the type of memory that is accessible to the processors. In the first case parallel computers are usually termed fine grained if there are many processors in the system, which may be as many as 64,000, or coarse grained if the system is composed of relatively few, approximately 5 to 20, large processors.

Also parallel computers can be differentiated by the architecture of the system memory. With shared memory architectures all processors have access to a common global memory while with distributed memory systems each processor has its own local memory and information exchange takes place through interprocessor communication. Because of the difficulties that arise when different processors try to access the same memory at the same time, it is usually the case that only coarse grained systems have shared memory and local or distributed memory is used for fine grained architectures.

On a shared memory system interprocessor communication is not necessary since once a processor writes data into a location in the global memory all other processors have access to this data. The advantage of this type of design is that

computer programming is simplified, however, accessing the common global memory can take longer, because contention problems can occur when several processors try to access the shared memory simultaneously.

On the other hand with distributed memory systems, memory contention problems do not exist, because each processor has its own local memory to store data. However, for problems where data must be shared between processors an interprocessor communication protocol must be developed by the programmer and excessive delays in communication can significantly degrade the performance of the system. It is because of these memory contention and communication problems that parallel processors usually do not approach their theoretical problem solving speed.

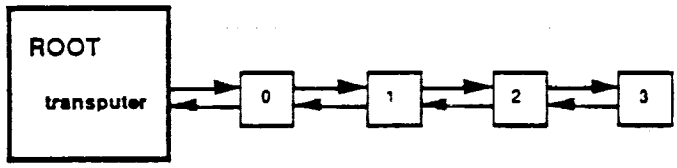
The purpose of this study was to develop a transient finite element program for parallel computation on a transputer system of processors. The transputer is a chip level processor with local memory and four communication links that can be used to connect transputers in a variety of configurations. Two different models of transputers are currently available. The INMOS T414 transputer has 2 MBytes of local memory and floating point performance of 0.1 MFLOPS while the INMOS T800 transputer has a floating point performance of 1.5 MFLOPS. The INMOS T414 transputer was used for the problems in this study. The four transputer links can simultaneously transmit data at a rate of 10 MBits per second.

The transputers are programmed using the OCCAM language[6] which was specifically developed to facilitate parallel programming and inter-processor communication.

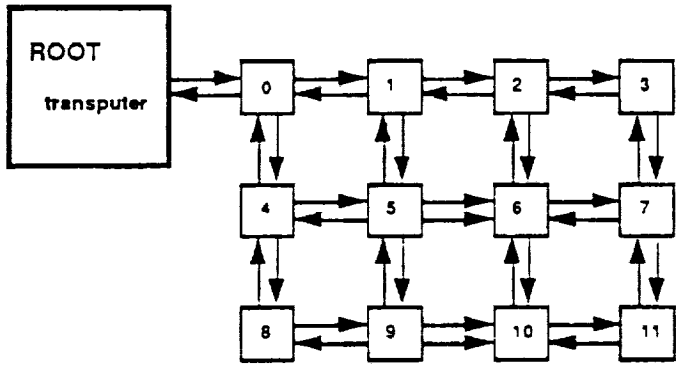
One transputer, known as the root transputer, can communicate with the host computer in this case an IBM PC. It is also used to edit and compile programs and in addition it distributes programs among the other transputers in the system which are referred to as the network.

As stated earlier, each transputer has four communication channels which allows it to be linked with other transputers. This flexibility allows a programmer the choice of different network configurations for instance: a torus, a hypercube, 2-D mesh, a pipeline and a binary tree topology. Some of these are sketched in Figure 1. Some of the configurations may be better than others for certain classes of problems. For the problems of interest here, a two-dimensional grid configuration was thought to be the most appropriate.

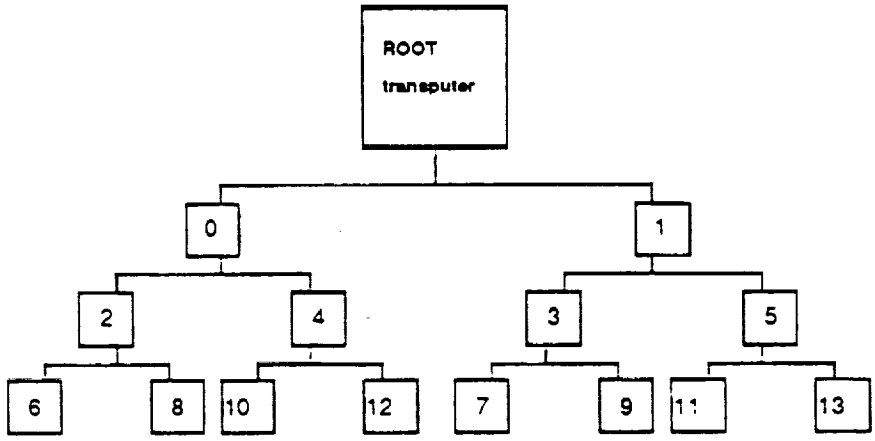
For this study, an explicit, finite-element program was written to analyze two and three dimensional transient problems. An explicit-integration algorithm was chosen since no equation solving is necessary and different nodes or nodal groups can be updated independently. To partition the problem for parallel computation, the nodes of the finite-element mesh are divided into groups and assigned to different processors. This partitioning, however, should be done in a way such that the amount of interprocessor communication is minimized.



a] pipeline



b] 2 - D grid



c] binary tree

Figure 1. Transputer network topologies

The initial task of the root processor is to define the problem data, define the information needed for interprocessor communication, and transmit this data to the network processors. The task of each of the network processors is to update the accelerations, velocities and displacements of the nodes in its group over a time step. It should be noted that although the updates of the nodal groups are uncoupled, the displacements of the nodes in other subdomains at the preceding time must be known in order to compute the internal forces. Therefore, after the new displacements in a subdomain are calculated, they must be communicated to other processors before the next update can proceed. To solve a problem most efficiently and achieve the greatest speedup over a sequential computer, the time used for interprocessor communication should be minimized. A flow chart for this program is given in Figure 2.

The problem parameters which are specified on the root processor are the nodal coordinates, element connectivity, the initial conditions $\underline{v}^0$ and $\underline{d}^0$, the time step, the number of time steps and the data defining the material properties. Because the program was written to be run on a 2-D grid processor configuration, the processor connectivity is defined by giving the number of rows and columns of processors. For example, the grid in Fig. 1b has 3 rows and 4 columns. Data specifying which nodes are assigned to which processors and the limit of interprocessor communication, the maximum

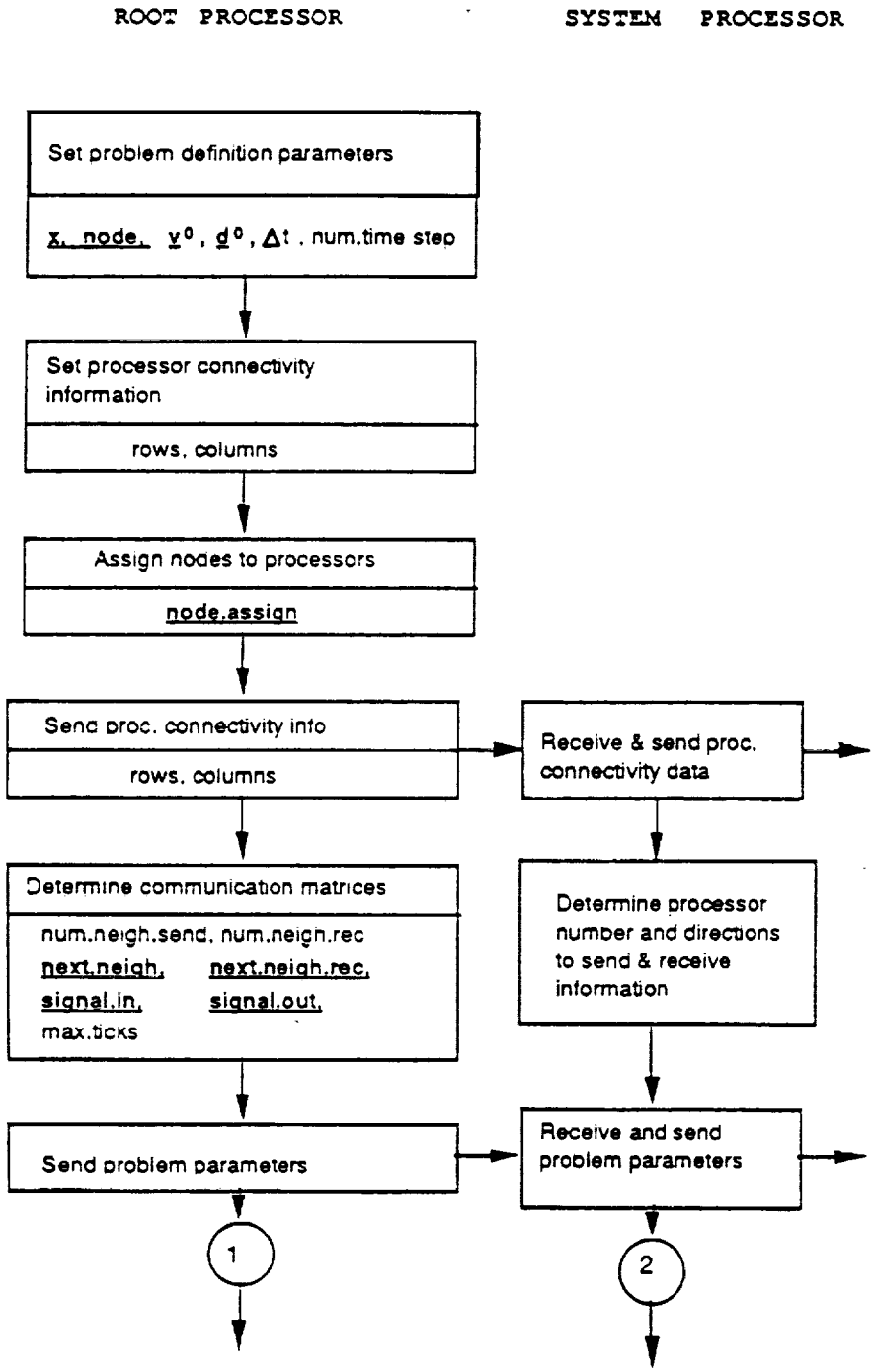


Figure 2. Flow chart of the parallel finite element program

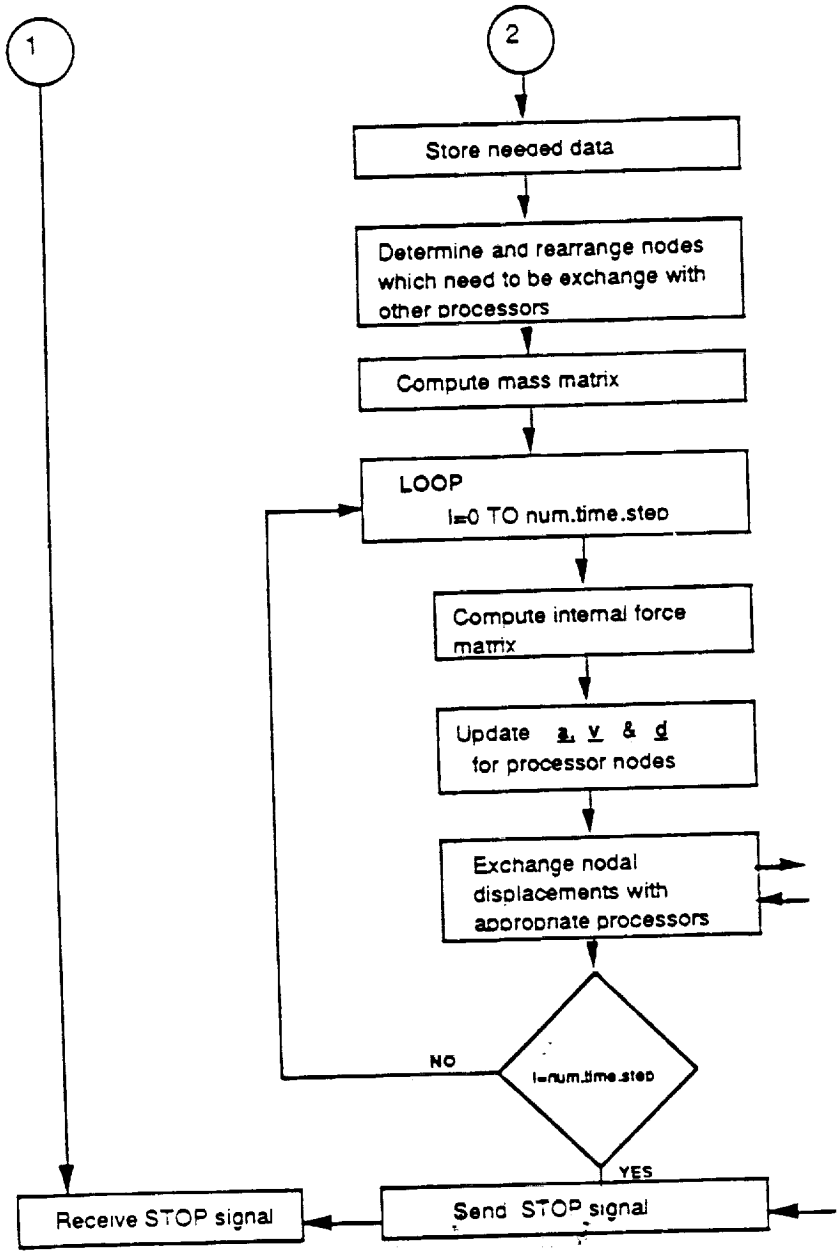


Figure 2. Cont.

distance of neighbors that a processor can communicate with, is also defined. The limit describes the maximum number of steps which is required to perform the interprocessor communication. In each step only the transputers which are connected can exchange data; for example, in Figure 1b, transputers number 5 and 6 can exchange information in one step but transputers number 4 & 6 in two steps.

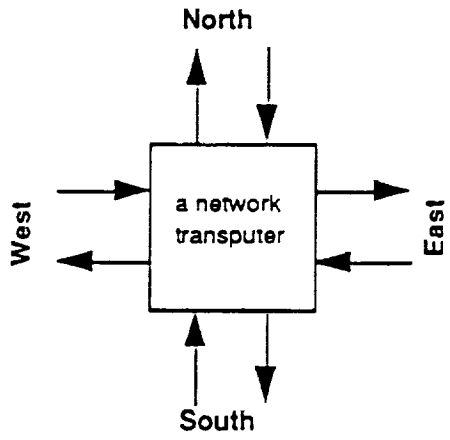
The first task of the root processor is to calculate the various matrices needed to describe the interprocessor communication. These matrices are determined by the structure of the finite element mesh, the grouping of the nodes and the size and shape of the processor grid. The matrix containing information on the number of processors with which each transputer has to exchange information is (num.neigh.send, num.neigh.rec) and the grid locations of these processors are in the matrices (next.neigh, next.neigh.rec). Two other matrices (signal.in and signal.out) contain information on when and from which direction each transputer must send and receive data. The root processor also calculates how many communication steps are needed for each processor to transmit data to its most remote neighbor (Figure 3). This gives an estimate how well the mesh has been partitioned. For instance, if one of the transputers needs many more steps than the others to reach its most remote neighbor, the partitioning of the mesh should be reconsidered.

As the root transputer transmits problem data, network

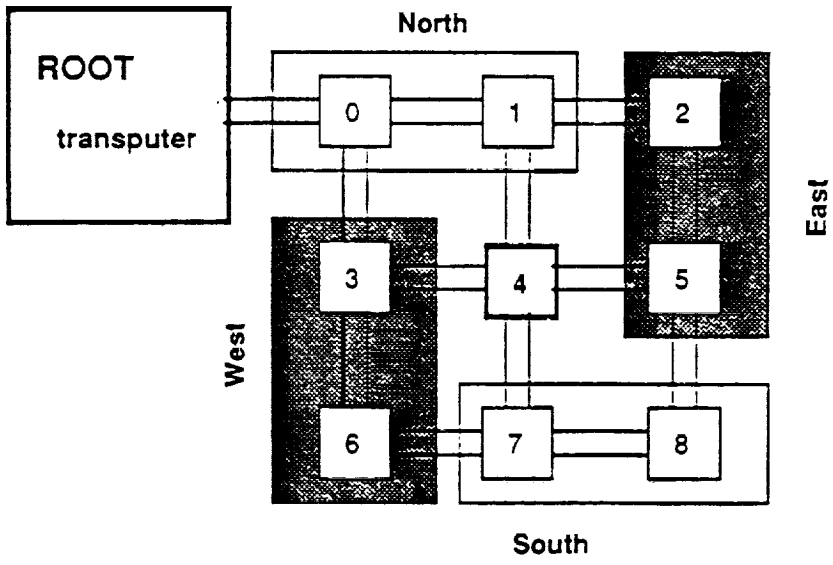
transputers receive this information, keep the needed data and send the information further to other transputers. Depending on the grid position of a transputer, it can receive information from west or north direction, see Fig. 3, and send it to east or south directions. The way in which data is initially distributed among transputers is shown in Figure 4.

Before a network transputer starts the time integration loop, it has to rearrange the problem data. Nodal displacements which must be exchanged are grouped together in increasing order according to the identification number of the transputer that receives data. Grouping displacements in this fashion ensures that nodes in the sending and in the receiving transputers are rearranged in the same order.

After calculating mass matrices for each element the time stepping is performed. First, the internal force matrix is calculated then nodal accelerations, velocities and displacements, respectively are updated. After each time step, updated nodal displacements are exchanged between the network processors.



a] transputer output and input channels



b] dependence of the output direction on the position of the receiving data transputer

Figure 3. Determination of output direction

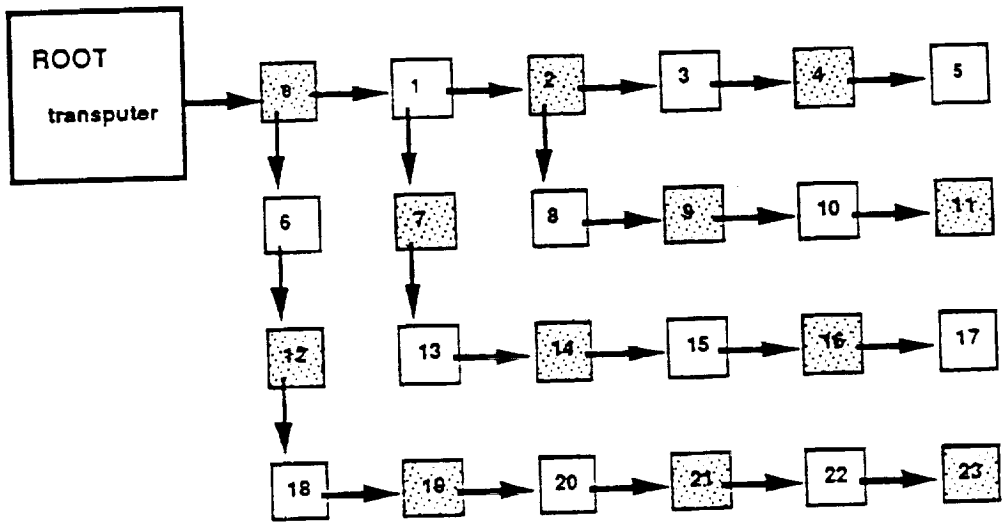


Figure 4. Data distribution among transputers

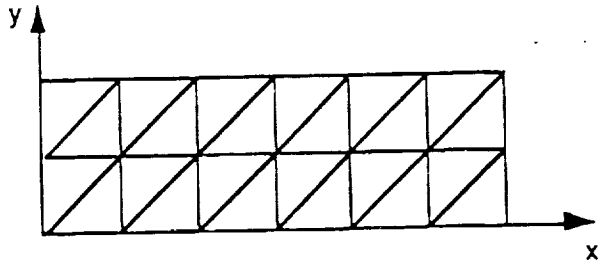
5.0 NUMERICAL EXAMPLES

Several numerical examples have been analyzed to evaluate the efficiency of the parallel algorithm. Unfortunately, since a large system of transputers was unavailable, the previously discussed communication algorithm was implemented on four T414 transputers connected in pipeline (Figure 1a) were used. In these examples, the size and geometry of the finite element mesh was varied to study the effects of different amounts of interprocessor communication. Currently the program uses linear triangular elements (Figure 5a) in two-dimensions and 8 node hexahedrals (Figure 5b) in three-dimensions.

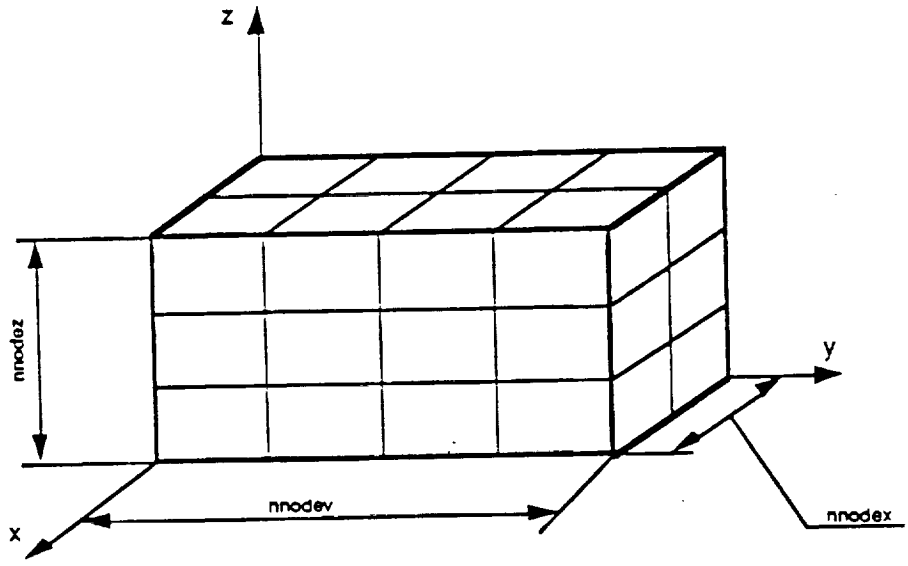
5.1 Three-dimensional Bar Model

The problem statement for this example is shown in Figure 6. One end of the bar was kept fixed while an initial displacement was applied to the opposite end of the bar. Figure 7 gives a plot of the displacement at the end of the bar as a function of time that was computed using the central difference algorithm.

To see how the amount of interprocessor communication affects the solution time, in this example, the number of nodes in the processor groups is varied while the number of nodal displacements that must be exchanged between processors is kept fixed. When the number of network transputers is



a] using linear triangular elements



b] using 8 node hexahedral elements

Figure 5. Finite element model of the bar problem

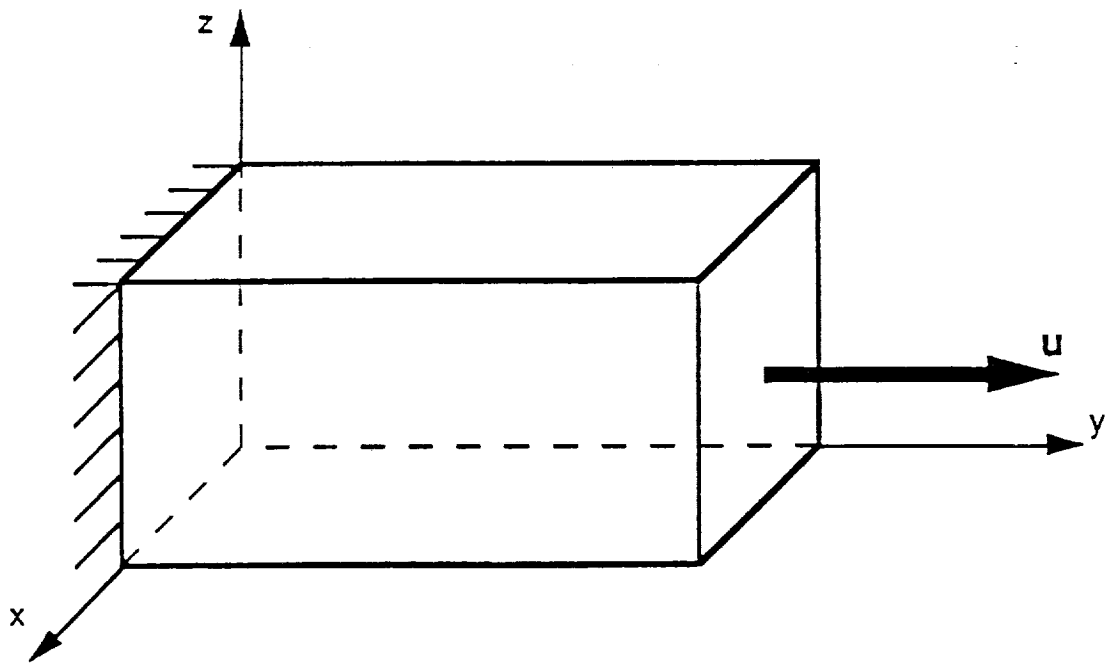


Figure 6. Problem statement for the three-dimensional bar

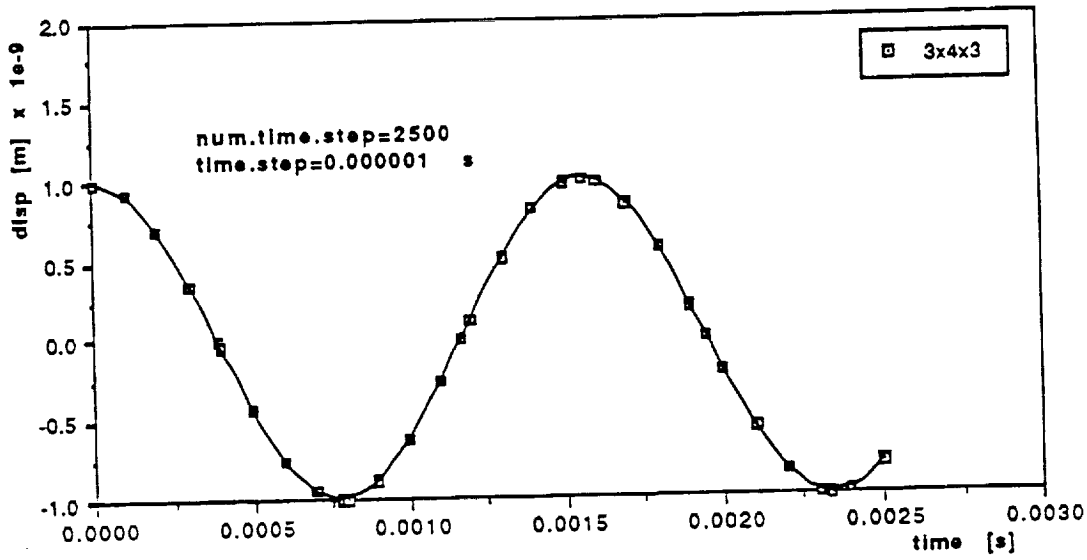


Figure 7. Displacements of the end of the bar as a function of time.

increased, a mesh of appropriate size is chosen to keep the number of processor elements and the number of nodes exchanged the same. The notation is used that nnodex, nnodey, nnodez indicate the number of nodes in x,y and z direction respectively and (nnodex) x (nnodey) x (nnodez) is the total number of nodes in the problem. The different cases that were considered for various number of processors are given in Tables A-1, A-2 and A-3. The solution times for these cases are given in Tables A-4 through A-12.

From an analysis of the measured solution times and of the internal structure of the program it was possible to obtain an approximate formula to calculate the execution time. The total solution time is assumed to be composed of three parts: computation time, communication time and preparation time in the form

$$T_{tot} = T_{cp} + T_{cm} + T_{prep} \qquad (5\text{-}1)$$

T_{tot} - total time

T_{cp} - computation time

T_{cm} - communication time

T_{prep} - preparation time

$$T_{prep} = N_{el}C_0^{*} + N_{el}N_{up}C_1 + kC_2 + kN_{nd}C_3 + N_{p.el}C_4 + N_{p.el}N_{nd}C_5 + N_{p.el}((N_{up}+N_{nd})/2)C_6 +$$

*Values of constants $C_0..C_{11}$ are given in Table 4.

$$k(N_{nd}-N_{up})C_7 \qquad (5\text{-}2)$$

$$T_{cp} = (N_{p.el}C_8 + N_{up}C_9)*N_{step} \qquad (5\text{-}3)$$

$$T_{cm} = (C_{10} + N_{ex}C_{11})*N_{step} \qquad (5\text{-}4)$$

where

k - is the integer of the ratio (N_{nodes}/length); data between transputers is sent in vectors of a fixed size, the time of sending & receiving data is proportional to k

length - the size of the vector sending data among transputers

The constants in these formulas were obtained by using the least square fit of the data and are given in Table 4. In the cases of computation and communication, times the measured values from Tables A-4 through A-12 were used. A different approach was taken for the calculation of the preparation time. To avoid solving large system of equations, the procedure was divided into six parts and approximate formulas were obtained for each part. In this case the computation was greatly simplified since for each part only two constants have to be calculated. The theoretical times obtain from above formulas are compared to actual values in Tables A-4 through A-12.

Several conclusions can be drawn from analysis of the results. First, the communication time is very small compared to the computation time and virtually can be neglected. Because data has to be exchanged after each time step the

Table 4.

Constants for calculation of the execution time

Constant	Value[s]
C_0	1.287×10^{-4}
C_1	1.366×10^{-5}
C_2	9.224×10^{-2}
C_3	1.462×10^{-2}
C_4	2.170×10^{-3}
C_5	1.749×10^{-5}
C_6	2.515×10^{-5}
C_7	6.623×10^{-3}
C_8	9.881×10^{-2}
C_9	8.735×10^{-4}
C_{10}	1.280×10^{-4}
C_{11}	4.640×10^{-5}

computation time in each step is determined by the processor with the largest work load. Consequently, it is very important to distribute the work load uniformly among network transputers.

When assigning the work load to transputers it should be taken into account that number of processor elements rather than number of nodes to update determines the computation time. This conclusion can be drawn from comparison of the constants in equation (5-3). The constant corresponding to $N_{p.el}$ (C_8) is much larger that the one corresponding to N_{up} (C_9).

To minimize the number of elements that must be stored by more than one processor, the assigning of the nodes to the processors should be done along the cross-section plane of the mesh with the smallest number of nodes.

The preparation time can be significant and for a small mesh running only a small number of time steps can even surpass the advantage of shorter computation time. For a given mesh and network of transputers, there exists a minimum number of time steps for which running the program concurrently is more efficient than sequential computation.

The above analysis shows that the total time depends primarily, particularly for larger number of time steps, on the number of elements assigned to each processor. The computation time is determined by the transputer with the largest number of elements, so it is important to assign an equal number of elements to all transputers. A simple

iterative algorithm was developed for nodal assignment that gives a relatively balanced processor work load.

For example, in the simplest case of 2 transputers we have the following relations

$$(N_{p.el})_A = A_{el} + C_{el} \quad (5\text{-}5a)$$

$$(N_{p.el})_B = B_{el} + C_{el} \quad (5\text{-}5b)$$

$$N_{el} = (N_{p.el})_A + (N_{p.el})_B - C_{el} \quad (5\text{-}5c)$$

$(N_{p.el})_A$ - total number of elements assigned to transputer A

$(N_{p.el})_B$ - total number of elements assigned to transputer B

A_{el} - elements assigned only to transputer A

B_{el} - elements assigned only to transputer B

C_{el} - common elements assigned to both transputers

The goal is to achieve $(N_{p.el})_A = (N_{p.el})_B$ or as close as possible, what occurs when $A_{el} = B_{el}$.

The algorithm assigning nodes in this way is presented in Figure 8. However, it should be noted that the results are influenced by the manner in which the numbering of elements was done. Figure 9 illustrates this for the case of 2-D grid.

5.1.1 Analysis of the Results

Figure 10 shows the plot of the total solution time as a function of the number of processors used in solving the cube problem; the points were obtained based on the formula (5-1). There are two reasons why the solution time does not decrease linearly as the number of processors increases. First, the data preparation time on the host computer depends on the size of the local processor matrices, which do not decrease proportianally to the number of processors, and also on the global problem parameters which are constant. So that the actual preparation time can increase with increasing numbers of processors. Second, the number of elements assigned to each transputer is not equal to the total number of elements divided by number of processors. It is increased by the number of elements which are shared by neighboring processors; this number has a constant value. This value depends on the shape of the structure being modelled. This dependence is illustrated by Figure 11; r represents the ratio A/A_{cube} where

A_{cube} - number of elements in the cross sectional area of the cube which has the given number of elements.

A - number of elements in the cross sectional area of a parallelepiped which has the same number elements.

The three possible shapes corresponding to different values of

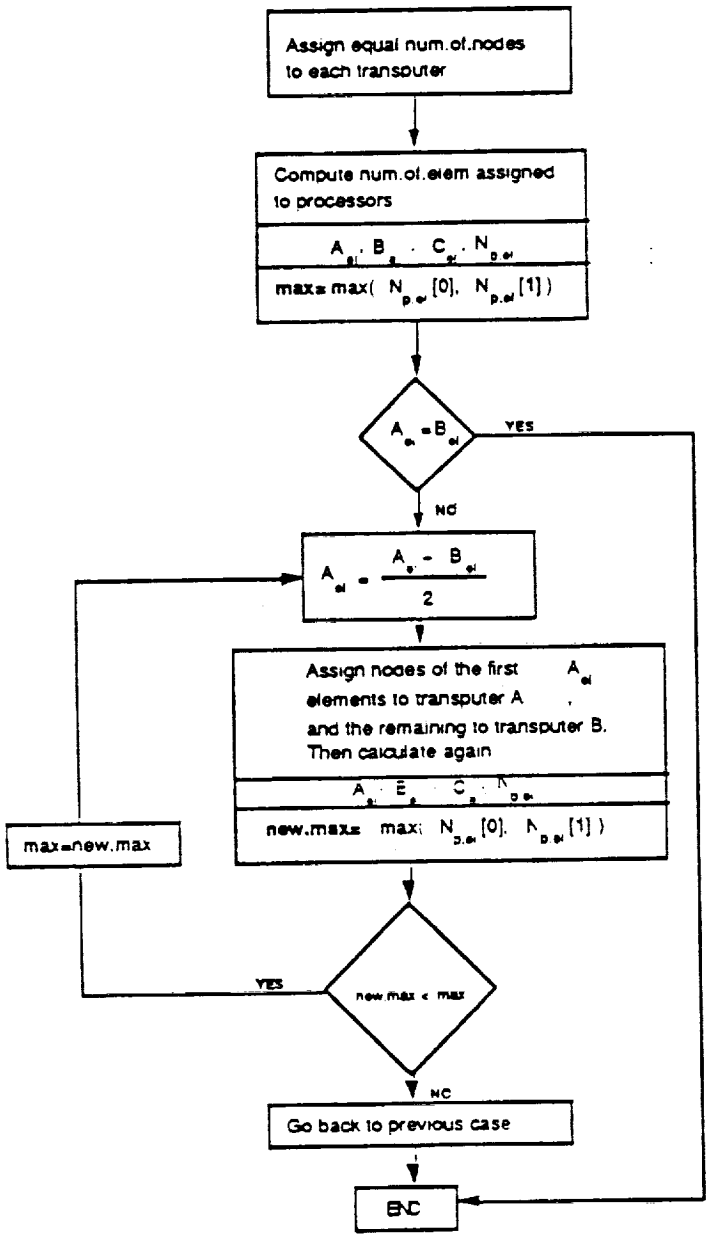
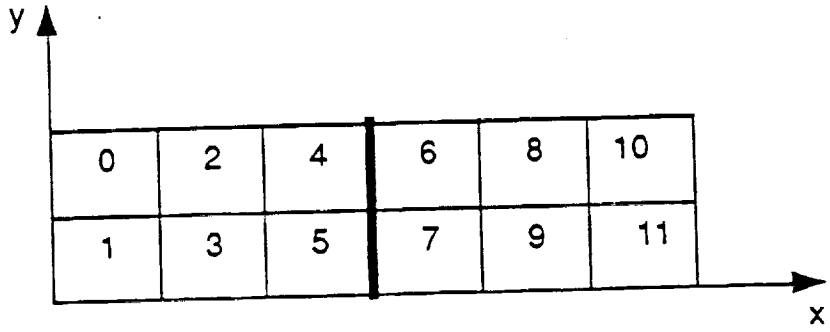
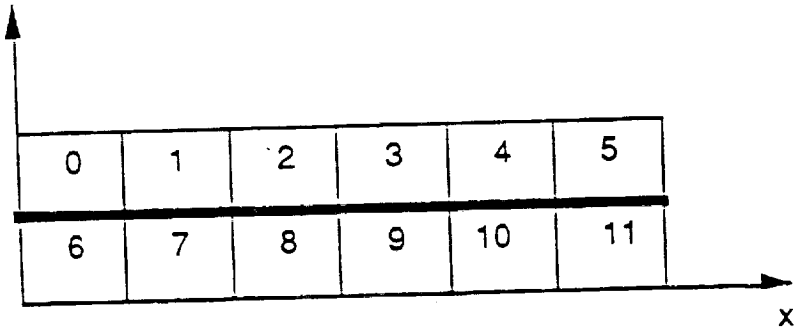


Figure 8. Flow chart of the procedure assigning nodes to transputers



a] numbering along the shorter side, $N_{p.el}=8$



b] numbering along the longer side, $N_{p.el}=12$

Figure 9. Dependence of the maximum num.proc.elem on the order of element numbering

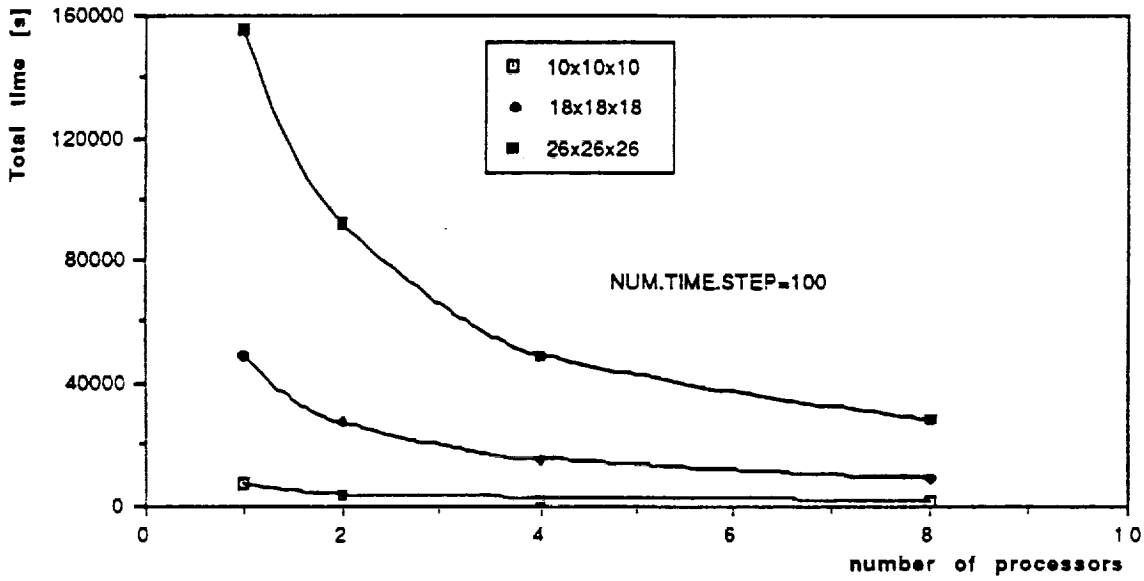
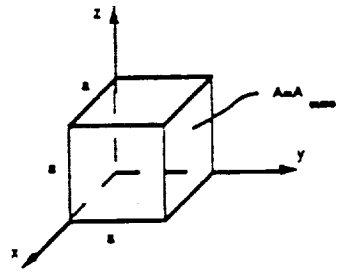
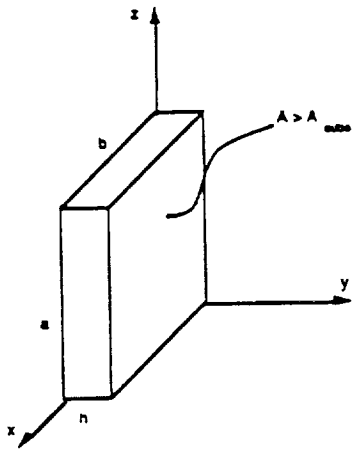


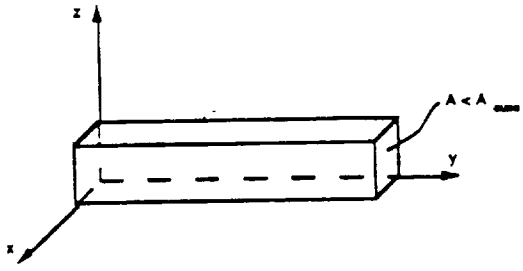
Figure 10. Total time as a function of number of processors



a] $r = \frac{A}{A_{cubo}} = 1$



b] $r = \frac{A}{A_{cubo}} > 1$



c] $r = \frac{A}{A_{cubo}} < 1$

Figure 11. Dependence of the shape of a parallelepiped on the value of the coefficient r.

the coefficient r are sketched in Figure 11 and Figure 12 represents the dependence of the total time on the shape of a parallelepiped. Theoretically, the larger the ratio the longer the total solution time. However, if r > 1, then the nodes have not been assigned to transputers along the shortest side. Consequently, if decomposition and numbering of the mesh has been done correctly the cube is the worst case. In Figure 12 the total time is increasing almost linearly. This results from the fact that, in this example, the computation time plays the dominating role (relatively large N_{step}=100). The computation time grows approximately linearly with $N_{p.el}$ because the influence of N_{up} is very small (see equation 5-3). Since number of elements assigned to each transputer grows linearly with r, the total time grows approximately linearly.

The results in Tables A-4 through A-12 also indicate that the communication time (T_{cm}) is very small and can be neglected. Then, the solution time can be divided into the preparation time (T_{prep}) and the computation time (T_{cp}). Preparation time is used to set up the problem data for parallel computations and would be absent in a sequential computation. Here, the transputer efficiency will be defined as the ratio of $([T_{cp}]_{seq}/p)/T_{tot}$. Since the transputer time is composed of computation time and data manipulation time, this value indicates what parts of the overall time is devoted to calculations & data manipulation. The results obtained from formula (5-1) are presented in Figure 13. In order to obtain

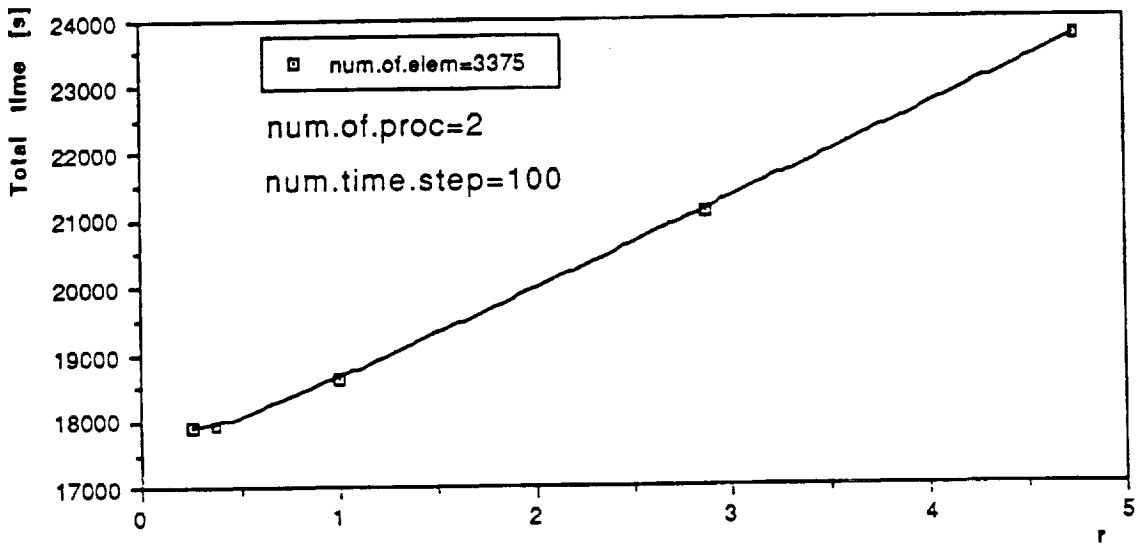


Figure 12. Total time as a function of the shape of a parallelepiped*, num.of.elem=const

* $r=A/A_{cube}$ for further explanation see page 31

equal values for local parameters; $N_{p.el}$, N_{up}, N_{nd}, the shape of the cross sectional area was kept unchanged, (nnodex) x (nnodez) = 5 x 6, while the value of nnodey was adjusted to give the required number of elements per processor. If T_{cm} is neglected, there are two additional times which occur in parallel computations and are absent in sequential. First is T_{prep} which does not depend on N_{step}. Second is the part of T_{cp} which requires duplicate computations over the elements belonging to the division border (these elements are assigned to two different transputers); this time is proportional to N_{step}. Three possible cases are illustrated by Figure 13. When T_{prep} is much larger than the additional computation time, that occurs for small N_{step} and large meshes (Figure 13a), processor efficiency decreases with num.of.elem per processor since T_{prep} grows faster than T_{cp}. The second extreme takes place when T_{prep} is small compared to additional computation time; this is true for large N_{step} (Figure 13c). Because the num.of.elem on the division border is kept constant, the processor efficiency increases with num.of.elem per processor. The third case occurs when both additional times have comparable influence (Figure 13b).

5.1.2 Estimation of Optimal Number of System Processors

Another problem of interest is to determine the number of transputers in the network that should be used to execute a given problem in the shortest time. It should be noted that

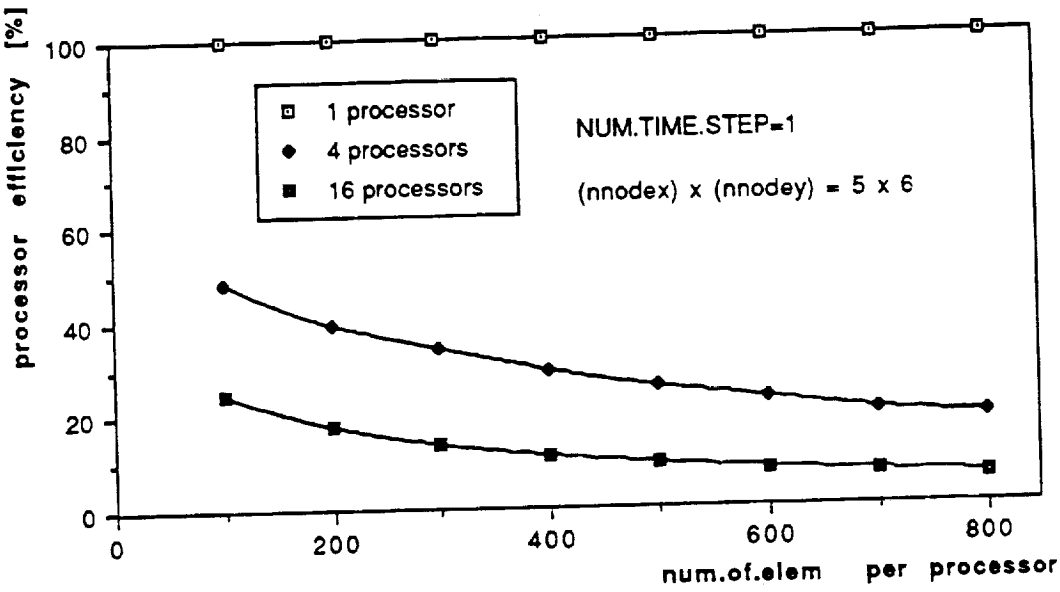


Figure 13a. Processor efficiency as a function of num.of.elem per processor*, num.time.step=1

* Here num.of.elem per processor is defined to be (N_{el}/p), which means that elements assigned to more than one transputer are counted only once. This insures that the efficiency is measured w.r.t. the sequential computation.

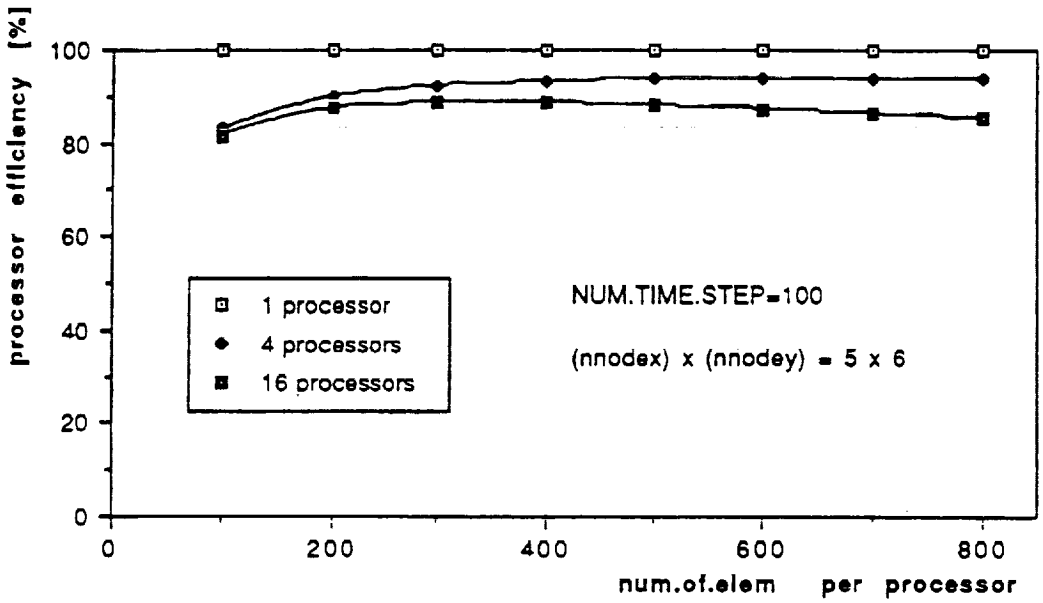


Figure 13b. Processor efficiency as a function of num.of.elem per processor*, num.time.step=100

* In this case num.of.elem per processor is equal to (N_{el}/p), what means that elements assigned to more than one transputer are counted only once. This insures that the efficiency is measured w.r.t. the sequential computation.

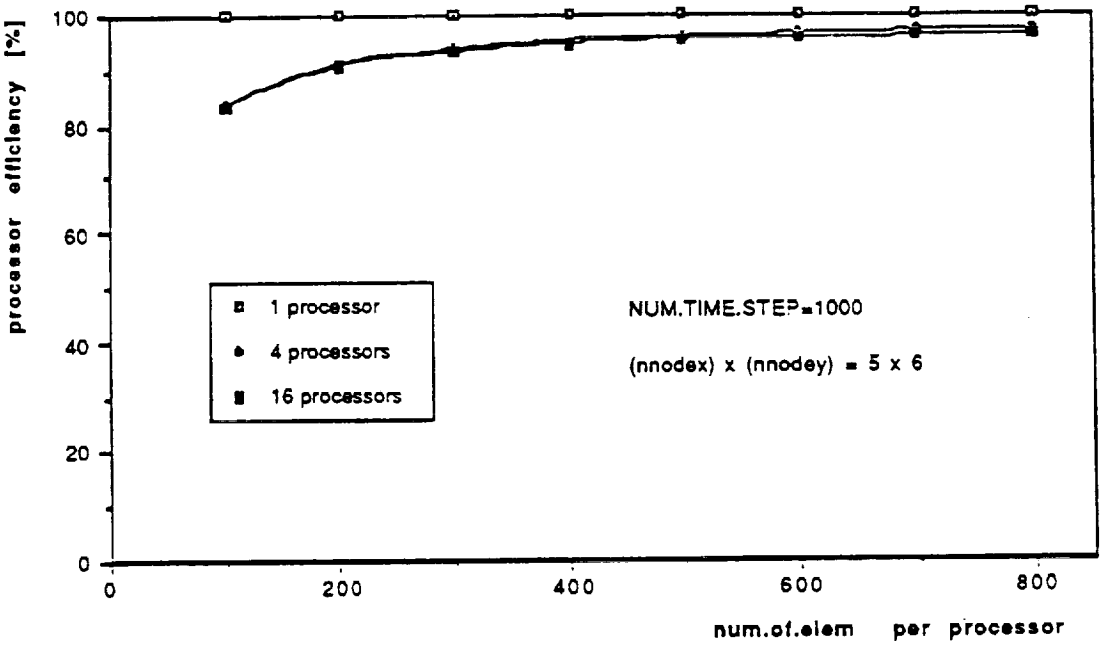


Figure 13c. Processor efficiency as a function of num.of.elem per processor*, num.time.step=1000

* In this case num.of.elem per processor is equal to (N_{el}/p), what means that elements assigned to more than one transputer are counted only once. This insures that the efficiency is measured w.r.t. the sequential computation.

depending on the problem size and the number of time steps, using the maximum number of processors may not yield the shortest time. To determine the shortest solution time, the following inequality needs to be analyzed

$$(T_{tot})_{1\ proc} > T_{prep} + T_{cp} \qquad (5\text{-}6)$$

$T_{tot})_{1\ proc}$ - total time for sequential computation

T_{prep} - preparation time for parallel computation

T_{cp} - computation time for parallel computation

For the bar 3-dimensional problem, the following equations can be used to calculate num.proc.elem, num.nodes.needed and num.update.nodes

$$h = N_{nodes}/(ab) \qquad (5\text{-}7a)$$

$$N_{up} = (sab) \qquad (5\text{-}7b)$$

$$N_{nd} = (s+1)(ab) \qquad (5\text{-}7c)$$

$$N_{p.el} = s[(a-1)(b-1)] \qquad (5\text{-}7d)$$

where

$$s = [((h-2)/p) + 1] \qquad (5\text{-}7e)$$

p - number of processors

a, b, h - number of nodes along the sides of the parallelepiped

After substitution into equation (5-6), we can obtain the minimum number of processors which must be used to reduce the total time below the total time for one processor. Data for

few chosen cases are presented in Figure 14. There are two factors which influence the value of the minimum number of processors. First, the preparation time increases for smaller numbers of processors. Second, the computation time decreases inversely with the number of processors. The minimum number of processors is the smallest value for which the decrease of T_{cp} compared to sequential computation is larger than T_{prep}. Of course, this value is smaller for problems where T_{cp} plays dominating role. This occurs in the case of problems with large number of time steps because T_{cp} increases linearly with N_{step} while T_{prep} remains unchanged, or when the assemble of the stiffness matrix requires a relatively long time. Figure 14 shows that even for a very small value of N_{step}, the minimum number of processors is the smallest possible, two, which means that two or more processors will give a faster solution than a sequential computation. The reason for this is because for 8 node hexahedrals time for the computation of the stiffness matrix is relatively large, so that even if only a few elements are involved this time is greater than the communication time. As was stated above, the minimum num.of.proc is the smallest value for which decrease of T_{cp} is larger than T_{prep}. Based on this, it can be explained why, for small N_{step}, its value is very large. If num.of.proc=2; T_{prep}, which does not depend on N_{step} and decreases with num.of.proc, can greatly exceed T_{cp}. When num.of.proc is increased, the gain in T_{cp} in absolute value is relatively small, so large

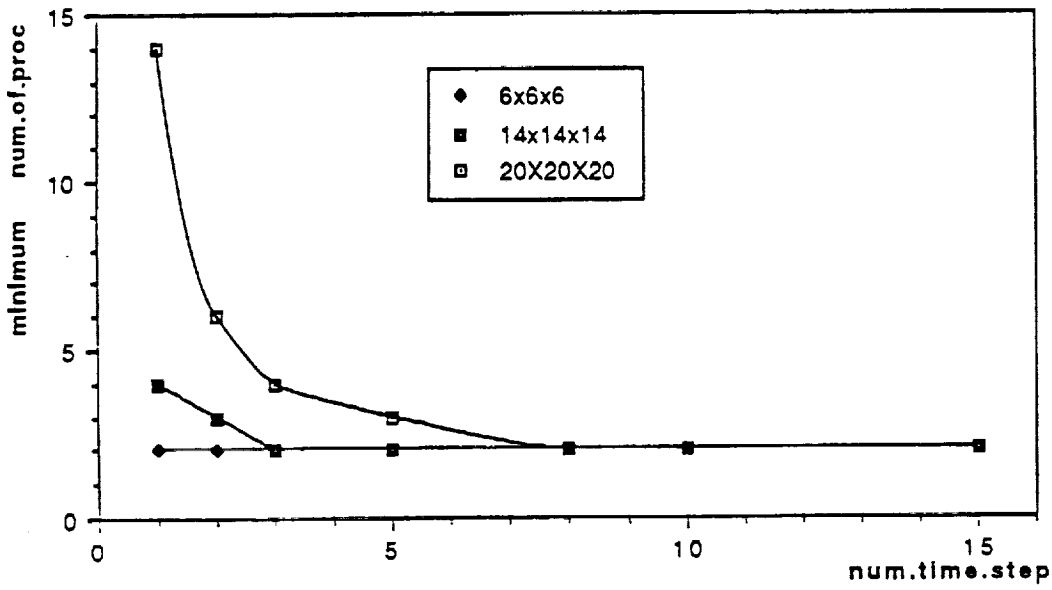


Figure 14. Minimum number of processors as a function of num.time.step

num.of.proc is required to reduce T_{prep} under this small value.

Equation (5-1) can be used to calculate the approximate execution time. However, the local data such as the number of update nodes, number of nodes needed, number of processor elements are not known before the program is executed since they are determined by the procedure performing the partition. Good accuracy can be achieved when the procedure calculating time is used along with the part of the program which performs the decomposition of the mesh and supplies the required data. In practice, however, the approximate execution time may be needed before input data is prepared. If we suppose that the structure has approximately parallelepiped shape, the local parameters mentioned above can be estimated.

For a given problem with N_{el}, number of elements, and N_{nodes}, number of nodes, T_{tot} can be estimated by one of the two methods presented below. First, as was shown previously, Figure 11, the longest total time for a constant N_{el} occurs in the case of a cube. We can assume that the structure is a cube composed of number of elements equal to N_{el} and then calculate components needed to compute T_{tot} from equations (5-7) and (5-1). In the second approach we can anticipate the shape of the parallelepiped using both N_{el} and N_{nodes}. The sides of the parallelepiped can be obtained from the following formulas

$$N_{nodes} = abh \qquad (5\text{-}8a)$$

$$N_{el} = (a-1)(b-1)(h-1) \qquad (5\text{-}8b)$$

or after eliminating h

$$(b^2-b)a^2 + (N_{nodes} + b[1-(N_{nodes}-N_{el})]-b^2)a + N_{nodes}(b-1)=0 \quad (5\text{-}9)$$

Since we have one equation and two unknowns, we have to assume the ratio a/b. For (a/b)=1 the above equation reduces to

$$a^4 - 2a^3 + [1-(N_{nodes}-N_{el})]a^2 + 2N_{nodes}a - N_{nodes} = 0 \quad (5\text{-}10)$$

Solving the equation and taking the root $\sqrt[3]{N_{nodes}} \geq a > 0$, the equations (5-7) and (5-1) can be used to estimate the total time.

5.2 Turbine Blade

A problem involving the analysis of a turbine blade was executed sequentially and on a two transputer network, with a mesh of 1575 nodes and 1025 elements (Figure 15). It was assumed that the bottom nodes of the blade are fixed while initial displacements are applied to the top nodes. The results together with the estimated times are grouped in the Tables 5 and 6.

In Table 5 the sequential and parallel solutions are compared. For num.time.step = 1, the sequential solution is faster than the concurrent one because the data manipulation time exceeds the gain in the computation time resulting from using two processors. It also should be noticed that with growing num.time.step the ratio $(T_{tot})_{2\ proc}/(T_{tot})_{1\ proc}$ becomes smaller but never reaches 50%. This is so because more than half of the elements are assigned to each processor and thus

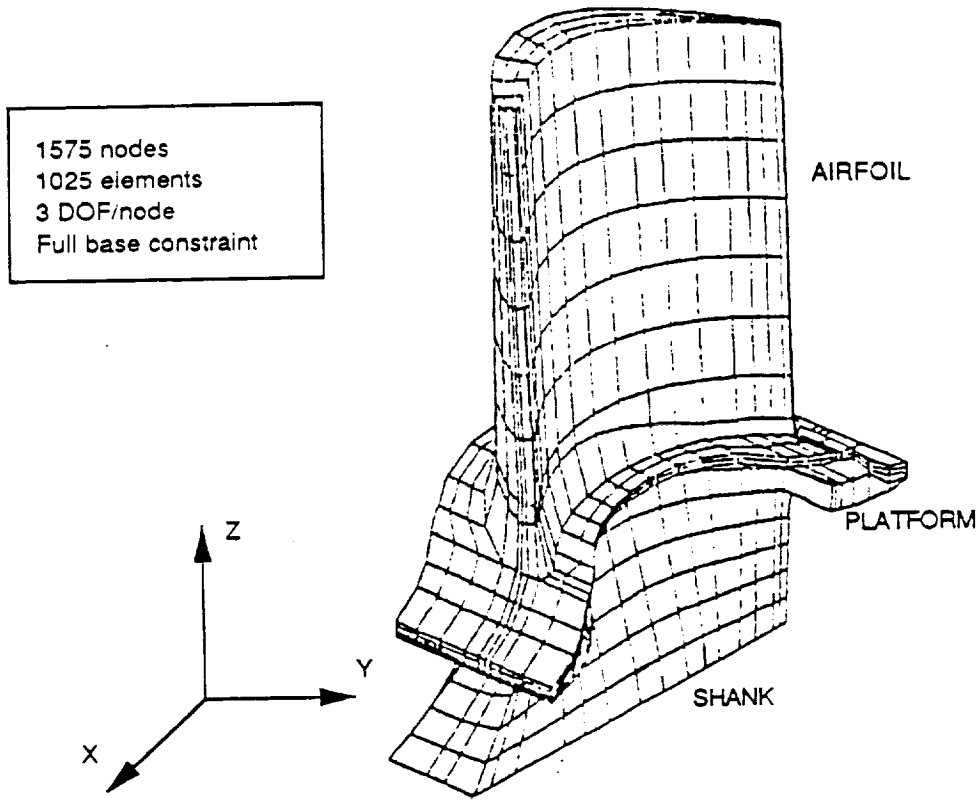


Figure 15. The turbine blade finite element model[7]

Table 5. Comparison of the sequential and parallel solutions for the turbine blade

number time steps	1 processor	2 processors	$(T_{tot})_{2\ proc}/(T_{tot})_{1\ proc}$ [%]
	T_{tot} [s]	T_{tot} [s]	
1	105.11	160.75	152.94
10	1054.20	737.70	69.98
20	2213.72	1433.20	64.74
50	5755.42	3519.79	61.16
100	11689.62	6996.75	59.85

Table 6. Solution times for the turbine problem.

num. time. step	MEASURED TIME*							COMPUTED TIME**			
	Total time [s]	Comp. time		Prep. time		Comm. time		Total time [s]	Comp. time [s]	Prep. time [s]	Comm. time [s]
		[s]	[%]	[s]	[%]	[s]	[%]				
1	170.85	66.82	39.11	104.02	60.88	0.0110	0.0064	160.74	65.60	95.13	0.0064
	160.75	59.65	37.11	101.09	62.89	0.0110	0.0068	138.16	53.8	84.32	0.0036
10	812.30	708.17	87.18	104.02	12.81	0.1101	0.0136	751.16	656.02	95.13	0.0640
	737.70	636.50	86.28	101.09	13.70	0.1101	0.0149	622.68	538.35	84.32	0.0360
20	1591.38	1,487.14	93.45	104.02	6.54	0.2202	0.0138	1407.1	1312.0	95.13	0.1281
	1433.20	1,331.89	92.93	101.09	7.05	0.2202	0.0154	1161.03	1076.7	84.32	0.0718
50	3929.37	3,824.80	97.34	104.02	2.65	0.5504	0.0140	3375.2	3280.1	95.13	0.3202
	3519.79	3,418.15	97.11	101.09	2.87	0.5504	0.0156	2776.09	2691.8	84.32	0.1795
100	7825.94	7,720.82	98.66	104.02	1.33	1.1008	0.0141	6655.3	6560.2	95.13	0.6404
	6996.75	6,894.56	98.54	101.09	1.44	1.1008	0.0157	5467.83	5383.5	84.32	0.3590

*The sdaded row - equal num.of.nodes assigned to each transputer, the unshaded row - equal num.of.elem assigned to transputers (see algorithm, Figure 8).

**The unshaded row - times estimated using the cube method, the shaded row - times estimated by assuming the shape of a parallelepiped (see section 5.1.2).

the total work load is larger than in the sequential computation. Also, the time is increased by the preparation time which is absent in sequential computation.

The measured times were compared with the calculated times in Table 6; the calculation was carried out using the two partitioning methods presented in the previous section. First, an equal number of nodes were sequentially assigned to each transputer, then the nodes were assigned in the manner such that each transputer handles an equal number of elements. As expected, in the second case the execution time is shorter; a reduction of about 10% was achieved.

In both cases, the calculated times are smaller than the measured times. This is because the shape of the turbine blade does not exactly correspond to the assumed parallelepiped model, and more nodes have to be exchanged between processors than it is predicted by the cube model. The element partition was made along the blade platform (see Figure 15).

5.3 Two-dimensional Example

The program was modified slightly in order to analyze two-dimensional problems more efficiently. The eight node solid element was replaced by a linear displacement triangular element which decreased the time of computation and assembly of the stiffness matrix. Also, where appropriate, three

dimensional vectors and matrices were replaced by two dimensional versions. The program was used to analyze several two-dimensional bar test problems (Table A-13) and the results were presented in Tables A-14 through A-19.

In comparison with the three-dimensional case, the ratio of communication time to computation time (T_{cm}/T_{cp}) for the two-dimensional problems is much larger, Table 7. Both times, T_{cm} and T_{cp}, are smaller for 2-D problems than for 3-D ones. However, the decrease of the computation time is much larger primarily because the calculation of the stiffness matrix is significantly faster than for the 3-dimensional element.

The 2-D problem was also examined by Patrick Smolinski[8] on T800 transputer (see page 12). Some results of his study are presented in Tables A-20 and A-21.

Comparing these results with those for T414 transputer (Table A-13 through A-19), we see that the total computation time is smaller for the T800 transputer. This is due to the higher performances of the T800 transputer (see page 12). However, we cannot calculate the ratio of the communication to the calculation time, since the communication time has not been measured for these problems.

Table 7.

Comparison of the ratio of the communication time per node to the computation time per element

CASE	$\frac{(T_{cm})_{per.node}}{(T_{cp})_{per.elem}}$
3-D	1.2 %
2-D	4.13 %

6.0 CONCLUSIONS

The results of this study indicate that the execution time of a finite element program can be considerably reduced by parallel computation using a relatively inexpensive transputer system. However, a price is paid since a larger and more complex computer program is required. In addition to the part of the computer program performing the actual calculations, routines performing communication and decomposition of the mesh have to be written.

For cases examined here, two and three-dimensional problems, the communication time, the time of exchanging displacements after each time step, was very small. However, if communication between more remote transputers is required this time will increase. In addition, when the time for assembling the stiffness matrix is relatively small and the number of exchanged nodes is large, the communication time has to be taken into account. For the problems solved in this study, the preparation time, the time of receiving and rearranging data, is more important. This time depends on the size of the structure and number of transputers used but not on the number of time steps. Consequently, for large number of time steps it becomes small in comparison to the computation time.

The total execution time depends on the size of the mesh and local parameters such as number of nodes and elements

assigned to each transputer. The computation time depends mostly on number of elements and other local parameters that are related to this value. For problems that require a large number of time steps for the solution, the computation time is usually much larger than other components of the total time. Because this is usually the case when a transputer systems would be used, it means that partitioning of the mesh can be done considering only the number of elements. For this reason perhaps the most important factor in minimizing the solution time is to assign an equal work load to each transputer since the total time is primarily governed by the transputer with the largest work load.

In most cases the parallel computation proves to be faster than the sequential one. The only cases when the sequential computation is faster occur for small number of time steps and are of no practical importance since most engineering problems require a significant number of time steps.

In the future it would be desirable to write a program which checks the order of element numbering in the connectivity matrix and if necessary rearrange it. This would optimize the results of the program performing the decomposition of a mesh, see Figure 7.

In this study, the program was executed on the only available transputer configuration, the pipeline (Figure 1a). The correctness of the program was checked by using a special

feature of the OCCAM language which allows to simulate an arbitrary transputer network while a program is executed sequentially. It would be interesting to run this program on a large grid of T800 processors to study the efficiency of the algorithm and to determine the speed-up that could be obtained with these much faster processors.

APPENDIX

APPENDIX

Table A-1.

Problem parameters for the three dimensional case, number of processors = 2

CASE	(nnodex)x (nnodey)x (nnodez)	num.of.proc = 2				
		GLOBAL		LOCAL		
		N_{nodes}	N_{el}	N_{nd}	N_{up}	$N_{p.el}$
0	4x10x5	200	108	120	100	60
1	4x20x5	400	228	220	200	120
2	4x30x5	600	348	320	300	180
3	8x10x5	400	252	240	200	140
4	8x20x5	800	532	440	400	280
5	8x30x5	1200	812	640	600	420
6	12x10x5	600	396	360	300	220
7	12x20x5	1200	836	660	600	440
8	12x30x5	1800	127	960	900	660
9	8x10x10	800	567	480	400	315
10	8x20x10	1600	1197	880	800	630
11	8x30x10	2400	1827	1280	1200	945

Table A-2.

Problem parameters for the three-dimensional case, number of processors = 3

CASE	(nnodex)x (nnodey)x (nnodez)	num.of.proc = 3				
		GLOBAL		LOCAL		
		N_{nodes}	N_{el}	N_{nd}	N_{up}	$N_{p.el}$
0	4X14X5	280	156	120	100	60
1	4x29x5	580	336	220	200	120
2	4x44x5	880	516	320	300	180
3	8x14x5	560	364	240	200	140
4	8x29x5	1160	784	440	400	280
5	8x44x5	1760	1204	640	600	420
6	12x14x5	840	572	360	300	220
7	12x29x5	1740	1232	660	600	440
8	12x44x5	2640	1892	960	900	660
9	8x14x10	1120	819	480	400	315
10	8x29x10	2320	1764	880	800	630
11	8x44x10	3520	2709	1280	1200	945

Table A-3.

Problem parameters for the three-dimensional case, number of processors = 4

CASE	(nnodex)x (nnodey)x (nnodez)	num.of.proc = 4				
		GLOBAL		LOCAL		
		N_{nodes}	N_{el}	N_{nd}	N_{up}	$N_{p.el}$
0	4X18X5	360	204	120	100	60
1	4x38x5	760	444	220	200	120
2	4x58x5	1160	684	320	300	180
3	8x18x5	720	476	240	200	140
4	8x38x5	1520	1036	440	400	280
5	8x58x5	2320	1596	640	600	420
6	12x18x5	1080	748	360	300	220
7	12x38x5	2280	1628	660	600	440
8	12x58x5	3480	2508	960	900	660
9	8x18x10	1440	1071	480	400	315
10	8x38x10	3040	2331	880	800	630
11	8x58x10	4640	3591	1280	1200	945

Table A-4.

Solution times for the three-dimensional bar,
number of processors=2,
num.time.step=10

CASE	GLOBAL*		MEASURED TIME							COMPUTED TIME			
	num of nodes	num of elem	T_{tot}	T_{cp}		T_{prep}		T_{cm}		T_{tot}	T_{cp}	T_{prep}	T_{cm}
				[s]	[%]	[s]	[%]	[s]	[%]	[s]	[s]	[s]	[s]
0	200	108	66.78	64.28	96.26	2.49	3.73	0.0102	0.0153	62.72	60.16	2.55	0.0102
1	400	228	129.55	124.16	95.84	5.38	4.15	0.0102	0.0079	125.75	120.31	5.43	0.0102
2	600	348	193.89	179.97	92.82	13.91	7.17	0.0102	0.0053	194.51	180.47	14.03	0.0102
3	400	252	153.59	147.43	95.99	6.14	4.00	0.0195	0.0127	146.32	140.07	6.23	0.0195
4	800	532	309.47	287.43	92.88	22.02	7.12	0.0195	0.0063	302.38	280.15	22.21	0.0195
5	1200	812	465.59	417.97	89.77	47.60	10.22	0.0195	0.0042	468.16	420.22	47.92	0.0195
6	600	396	244.02	227.50	93.23	16.49	6.76	0.0288	0.0118	236.83	219.99	16.81	0.0288
7	1200	836	493.59	443.76	89.90	49.80	10.09	0.0288	0.0058	490.25	439.98	50.24	0.0288
8	1800	1276	765.37	650.10	84.94	115.24	15.06	0.0288	0.0038	776.08	659.97	116.08	0.0288
9	800	567	350.21	325.32	92.89	24.85	7.10	0.0381	0.0109	339.96	314.73	25.19	0.0381
10	1600	1197	729.48	638.91	87.58	90.53	12.41	0.0381	0.0052	720.81	629.46	91.31	0.0381
11	2400	1827	1130.76	933.73	82.58	196.99	17.42	0.0381	0.0034	1,142.5	944.19	198.33	0.0381

*Further discription of the cases can be found in Table A-1.

Table A-5.

Solution times for the three-dimensional bar,
number of processors=2,
num.time.step=50

CASE	GLOBAL* num of nodes	GLOBAL* num of elem	MEASURED TIME T_{tot}	T_{cp} [s]	T_{cp} [%]	T_{prep} [s]	T_{prep} [%]	T_{cm} [s]	T_{cm} [%]	COMPUTED TIME T_{tot} [s]	T_{cp} [s]	T_{prep} [s]	T_{cm} [s]
0	200	108	306.62	304.09	99.17	2.48	0.81	0.0512	0.0167	303.38	300.78	2.55	0.0512
1	400	228	610.37	604.94	99.11	5.38	0.88	0.0512	0.0084	607.05	601.57	5.43	0.0512
2	600	348	913.36	899.40	98.47	13.91	1.52	0.0512	0.0056	916.43	902.35	14.03	0.0512
3	400	252	713.55	707.31	99.13	6.14	0.86	0.0976	0.0137	706.70	700.37	6.23	0.0976
4	800	532	1432.01	1,409.90	98.46	22.01	1.54	0.0976	0.0068	1,423.05	1400.74	22.21	0.0976
5	1200	812	2145.17	2,097.47	97.78	47.60	2.22	0.0976	0.0045	2,149.13	2101.11	47.92	0.0976
6	600	396	1124.58	1,107.95	98.52	16.49	1.47	0.1440	0.0128	1,116.91	1099.96	16.81	0.1440
7	1200	836	2259.50	2,209.57	97.79	49.79	2.20	0.1440	0.0064	2,250.30	2199.92	50.24	0.1440
8	1800	1276	3406.67	3,291.30	96.61	115.23	3.38	0.1440	0.0042	3,416.09	3299.87	116.08	0.1440
9	800	567	1609.96	1,584.93	98.45	24.84	1.54	0.1904	0.0118	1,599.03	1573.65	25.19	0.1904
10	1600	1197	3254.71	3,164.00	97.21	90.52	2.78	0.1904	0.0058	3,238.80	3147.30	91.31	0.1904
11	2400	1827	4907.59	4,710.41	95.98	196.99	4.01	0.1904	0.0039	4,919.47	4720.95	198.33	0.1904

*further discription of the case can be found in Table A-1.

Table A-6.

Solution times for the three-dimensional bar,
number of processors=2,
num.time.step=100

CASE	GLOBAL*		MEASURED TIME							COMPUTED TIME			
	num of nodes	num of elem	T_{tot}	T_{cp}		T_{prep}		T_{cm}		T_{tot}	T_{cp}	T_{prep}	T_{cm}
				[s]	[%]	[s]	[%]	[s]	[%]	[s]	[s]	[s]	[s]
0	200	108	605.11	602.53	99.57	2.48	0.41	0.1024	0.0169	604.22	601.57	2.55	0.1024
1	400	228	1211.33	1,205.85	99.55	5.38	0.44	0.1024	0.0085	1,208.66	1203.13	5.43	0.1024
2	600	348	1813.48	1,799.47	99.23	13.91	0.77	0.1024	0.0056	1,818.83	1804.70	14.03	0.1024
3	400	252	1410.22	1,403.88	99.55	6.14	0.44	0.1952	0.0138	1,407.17	1400.74	6.23	0.1952
4	800	532	2834.81	2,812.60	99.22	22.01	0.78	0.1952	0.0069	2,823.89	2801.48	22.21	0.1952
5	1200	812	4246.47	4,198.67	98.87	47.60	1.12	0.1952	0.0046	4,250.34	4202.22	47.92	0.1952
6	600	396	2219.92	2,203.14	99.24	16.49	0.74	0.2880	0.0130	2,217.02	2199.92	16.81	0.2880
7	1200	836	4465.07	4,414.99	98.88	49.79	1.12	0.2880	0.0065	4,450.36	4399.83	50.24	0.2880
8	1800	1276	6710.76	6,595.24	98.28	115.23	1.72	0.2880	0.0043	6,716.12	6599.75	116.08	0.2880
9	800	567	3176.70	3,151.48	99.21	24.84	0.78	0.3808	0.0120	3,172.87	3147.30	25.19	0.3808
10	1600	1197	6408.81	6,317.91	98.58	90.52	1.41	0.3808	0.0059	6,386.28	6294.59	91.31	0.3808
11	2400	1827	9632.30	9,434.93	97.95	196.99	2.05	0.3808	0.0040	9,640.60	9441.89	198.33	0.3808

*Further discription of the cases can be found in Table A-1.

Table A-7.

Solution times for the three-dimensional bar,
number of processors=3,
num.time.step=10

CASE	GLOBAL*		MEASURED TIME							COMPUTED TIME			
	num of nodes	num of elem	T_{tot}	T_{cp}		T_{prep}		T_{cm}		T_{tot}	T_{cp}	T_{prep}	T_{cm}
				[s]	[%]	[s]	[%]	[s]	[%]	[s]	[s]	[s]	[s]
0	280	156	67.53	64.88	96.08	2.63	3.89	0.0205	0.0304	62.80	60.16	2.62	0.0205
1	580	336	132.25	123.07	93.06	9.16	6.93	0.0205	0.0155	129.49	120.31	9.16	0.0205
2	880	516	198.33	178.69	90.10	19.62	9.89	0.0205	0.0103	200.11	180.47	19.62	0.0205
3	560	364	159.24	148.80	93.44	10.40	6.53	0.0390	0.0245	150.50	140.07	10.40	0.0390
4	1160	784	315.61	285.15	90.35	30.42	9.64	0.0390	0.0124	310.56	280.15	30.38	0.0390
5	1760	1204	485.76	415.18	85.47	70.54	14.52	0.0390	0.0080	490.78	420.22	70.53	0.0390
6	840	572	253.13	229.80	90.78	23.27	9.19	0.0576	0.0228	243.33	219.99	23.29	0.0576
7	1740	1232	514.96	441.15	85.67	73.75	14.32	0.0576	0.0112	513.75	439.98	73.72	0.0576
8	2640	1892	798.86	646.11	80.88	152.69	19.11	0.0576	0.0072	812.68	659.97	152.66	0.0576
9	1120	819	362.78	328.41	90.53	34.29	9.45	0.0762	0.0210	349.00	314.73	34.21	0.0762
10	2320	1764	759.69	635.07	83.60	124.54	16.39	0.0762	0.0100	753.95	629.46	124.43	0.0762
11	3520	2709	1198.7	927.92	77.41	270.70	22.58	0.0762	0.0064	1,214.89	944.19	270.64	0.0762

*Further discription of the cases can be found in Table A-2.

Table A-8.

Solution times for the three-dimensional bar,
number of processors=3,
num.time.step=50

CASE	GLOBAL*		MEASURED TIME							COMPUTED TIME			
	num of nodes	num of elem	T_{tot}	T_{cp}		T_{prep}		T_{cm}		T_{tot}	T_{cp}	T_{prep}	T_{cm}
				[s]	[%]	[s]	[%]	[s]	[%]	[s]	[s]	[s]	[s]
0	280	156	310.04	307.31	99.12	2.63	0.85	0.1024	0.0330	303.49	300.78	2.62	0.1024
1	580	336	613.05	603.88	98.50	9.16	1.49	0.1024	0.0167	610.82	601.57	9.16	0.1024
2	880	516	917.40	897.77	97.86	19.62	2.14	0.1024	0.0112	922.06	902.35	19.62	0.1024
3	560	364	725.28	714.68	98.54	10.40	1.43	0.1952	0.0269	710.93	700.37	10.40	0.1952
4	1160	784	1437.86	1,407.24	97.87	30.42	2.12	0.1952	0.0136	1,431.28	1400.74	30.38	0.1952
5	1760	1204	2164.39	2,093.65	96.73	70.54	3.26	0.1952	0.0090	2,171.80	2101.11	70.53	0.1952
6	840	572	1143.38	1,119.82	97.94	23.27	2.04	0.2880	0.0252	1,123.49	1099.96	23.29	0.2880
7	1740	1232	2279.65	2,205.61	96.75	73.75	3.24	0.2880	0.0126	2,273.88	2199.92	73.72	0.2880
8	2640	1892	3438.17	3,285.19	95.55	152.69	4.44	0.2880	0.0084	3,452.77	3299.87	152.66	0.2880
9	1120	819	1636.37	1,601.70	97.88	34.29	2.10	0.3808	0.0233	1,608.18	1573.65	34.21	0.3808
10	2320	1764	3283.37	3,158.45	96.20	124.54	3.79	0.3808	0.0116	3,272.05	3147.30	124.43	0.3808
11	3520	2709	4973.70	4,702.62	94.55	270.70	5.44	0.3808	0.0077	4,991.91	4720.95	270.64	0.3808

*Further discription of the cases can be found in Table A-2.

Table A-9.

Solution times for the three-dimensional bar,
number of processors=3,
num.time.step=100

CASE	GLOBAL*		MEASURED TIME							COMPUTED TIME			
	num of nodes	num of elem	T_{tot}	T_{cp}		T_{prep}		T_{cm}		T_{tot}	T_{cp}	T_{prep}	T_{cm}
				[s]	[%]	[s]	[%]	[s]	[%]	[s]	[s]	[s]	[s]
0	280	156	612.63	609.80	99.54	2.63	0.43	0.2048	0.0334	604.36	601.57	2.62	0.2048
1	580	336	1214.06	1,204.70	99.23	9.16	0.75	0.2048	0.0169	1,212.46	1203.13	9.16	0.2048
2	880	516	1817.99	1,798.17	98.91	19.62	1.08	0.2048	0.0113	1,824.49	1804.70	19.62	0.2048
3	560	364	1431.38	1,420.59	99.25	10.40	0.73	0.3904	0.0273	1,411.47	1400.74	10.40	0.3904
4	1160	784	2840.69	2,809.88	98.92	30.42	1.07	0.3904	0.0137	2,832.19	2801.48	30.38	0.3904
5	1760	1204	4266.82	4,195.89	98.34	70.54	1.65	0.3904	0.0091	4,273.08	4202.22	70.53	0.3904
6	840	572	2253.58	2,229.73	98.94	23.27	1.03	0.5760	0.0256	2,223.69	2199.92	23.29	0.5760
7	1740	1232	4485.30	4,410.97	98.34	73.75	1.64	0.5760	0.0128	4,474.03	4399.83	73.72	0.5760
8	2640	1892	6743.84	6,590.57	97.73	152.69	2.26	0.5760	0.0085	6,752.89	6599.75	152.66	0.5760
9	1120	819	3224.17	3,189.12	98.91	34.29	1.06	0.7616	0.0236	3,182.14	3147.30	34.21	0.7616
10	2320	1764	6437.72	6,312.42	98.05	124.54	1.93	0.7616	0.0118	6,419.65	6294.59	124.43	0.7616
11	3520	2709	9701.57	9,430.11	97.20	270.70	2.79	0.7616	0.0079	9,713.16	9441.89	270.64	0.7616

*Further discription of the cases can be found in Table A-2.

Table A-10.

Solution times for the three-dimensional bar,
number of processors=4,
num.time.step=10

CASE	GLOBAL*		MEASURED TIME							COMPUTED TIME			
	num of nodes	num of elem	T_{tot}	T_{cp}		T_{prep}		T_{cm}		T_{tot}	T_{cp}	T_{prep}	T_{cm}
				[s]	[%]	[s]	[%]	[s]	[%]	[s]	[s]	[s]	[s]
0	360	204	67.13	64.41	95.95	2.70	4.02	0.0205	0.0305	62.88	60.16	2.70	0.0205
1	760	444	131.93	122.41	92.78	9.50	7.20	0.0205	0.0155	129.81	120.31	9.48	0.0205
2	1160	684	198.20	177.82	89.72	20.36	10.27	0.0205	0.0103	200.82	180.47	20.33	0.0205
3	720	476	158.60	147.82	93.20	10.74	6.77	0.0390	0.0246	150.84	140.07	10.73	0.0390
4	1520	1036	322.28	283.64	88.01	38.60	11.98	0.0390	0.0121	318.74	280.15	38.55	0.0390
5	2320	1596	496.67	413.10	83.17	83.53	16.82	0.0390	0.0079	503.72	420.22	83.46	0.0390
6	1080	748	252.36	228.23	90.44	24.07	9.54	0.0576	0.0228	244.08	219.99	24.03	0.0576
7	2280	1628	526.36	439.10	83.42	87.20	16.57	0.0576	0.0109	527.15	439.98	87.11	0.0576
8	3480	2508	832.59	643.23	77.26	189.30	22.74	0.0576	0.0069	849.25	659.97	189.22	0.0576
9	1440	1071	369.91	326.53	88.27	43.30	11.71	0.0762	0.0206	358.03	314.73	43.22	0.0762
10	3040	2331	789.44	631.65	80.01	157.71	19.98	0.0762	0.0097	787.09	629.46	157.55	0.0762
11	4640	3591	1266.49	923.24	72.90	343.17	27.10	0.0762	0.0060	1,287.22	944.19	342.95	0.0762

*Further discription of the cases can be found in Table A-3.

Table A-11.

Solution times for the three-dimensional bar,
number of processors=4,
num.time.step=50

CASE	GLOBAL*		MEASURED TIME							COMPUTED TIME			
	num of nodes	num of elem	T_{tot}	T_{cp}		T_{prep}		T_{cm}		T_{tot}	T_{cp}	T_{prep}	T_{cm}
				[s]	[%]	[s]	[%]	[s]	[%]	[s]	[s]	[s]	[s]
0	360	204	308.36	305.56	99.09	2.70	0.88	0.1024	0.0332	303.57	300.78	2.70	0.1024
1	760	444	612.56	602.96	98.43	9.50	1.55	0.1024	0.0167	611.14	601.57	9.48	0.1024
2	1160	684	917.10	896.64	97.77	20.36	2.22	0.1024	0.0112	922.77	902.35	20.33	0.1024
3	720	476	721.65	710.71	98.48	10.74	1.49	0.1952	0.0270	711.26	700.37	10.73	0.1952
4	1520	1036	1444.46	1,405.66	97.31	38.60	2.67	0.1952	0.0135	1,439.45	1400.74	38.55	0.1952
5	2320	1596	2174.98	2,091.25	96.15	83.53	3.84	0.1952	0.0090	2,184.73	2101.11	83.46	0.1952
6	1080	748	1137.70	1,113.34	97.86	24.07	2.12	0.2880	0.0253	1,124.23	1099.96	24.03	0.2880
7	2280	1628	2290.85	2,203.36	96.18	87.20	3.81	0.2880	0.0126	2,287.27	2199.92	87.11	0.2880
8	3480	2508	3471.39	3,281.80	94.54	189.30	5.45	0.2880	0.0083	3,489.33	3299.87	189.22	0.2880
9	1440	1071	1636.14	1,592.46	97.33	43.30	2.65	0.3808	0.0233	1,617.19	1573.65	43.22	0.3808
10	3040	2331	3312.56	3,154.47	95.23	157.71	4.76	0.3808	0.0115	3,305.17	3147.30	157.55	0.3808
11	4640	3591	5040.18	4,696.63	93.18	343.17	6.81	0.3808	0.0076	5,064.22	4720.95	342.95	0.3808

*Further discription of the cases can be found in Table A-3.

Table A-12.

Solution times for the three-dimensional bar,
number of processors=4,
num.time.step=100

CASE	GLOBAL*		MEASURED TIME						COMPUTED TIME			
	num of nodes	num of elem	T_{tot}	T_{cp}		T_{prep}		T_{cm}	T_{tot}	T_{cp}	T_{prep}	T_{cm}
				[s]	[%]	[s]	[%]	[s] [%]	[s]	[s]	[s]	[s]
0	360	204	609.85	606.95	99.52	2.70	0.44	0.2048 0.0336	604.47	601.57	2.70	0.2048
1	760	444	1213.40	1,203.70	99.20	9.50	0.78	0.2048 0.0169	1,212.81	1203.13	9.48	0.2048
2	1160	684	1819.54	1,798.98	98.87	20.36	1.12	0.2048 0.0113	1,825.23	1804.70	20.33	0.2048
3	720	476	1425.31	1,414.18	99.22	10.74	0.75	0.3904 0.0274	1,411.86	1400.74	10.73	0.3904
4	1520	1036	2847.32	2,808.33	98.63	38.60	1.36	0.3904 0.0137	2,840.42	2801.48	38.55	0.3904
5	2320	1596	4281.63	4,197.71	98.04	83.53	1.95	0.3904 0.0091	4,286.07	4202.22	83.46	0.3904
6	1080	748	2244.10	2,219.45	98.90	24.07	1.07	0.5760 0.0257	2,224.53	2199.92	24.03	0.5760
7	2280	1628	4496.71	4,408.93	98.05	87.20	1.94	0.5760 0.0128	4,487.52	4399.83	87.11	0.5760
8	3480	2508	6783.74	6,593.86	97.20	189.30	2.79	0.5760 0.0085	6,789.55	6599.75	189.22	0.5760
9	1440	1071	3218.30	3,174.24	98.63	43.30	1.35	0.7616 0.0237	3,191.28	3147.30	43.22	0.7616
10	3040	2331	6466.78	6,308.31	97.55	157.71	2.44	0.7616 0.0118	6,452.90	6294.59	157.55	0.7616
11	4640	3591	9776.68	9,432.75	96.48	343.17	3.51	0.7616 0.0078	9,785.60	9441.89	342.95	0.7616

*Further discription of the cases can be found in Table A-3.

Table A-13.

Problem parameters for the two-dimensional case

num of proc	CASE	(nnodex)x (nnodey)	GLOBAL		LOCAL		
			N_{nodes}	N_{el}	N_{nd}	N_{up}	$N_{p.el}$
2	0	10x10	100	162	60	50	90
	1	20x10	200	342	110	100	180
	2	40x10	400	702	210	200	360
4	0	10x10	100	162	46	25	64
	1	20x10	200	342	70	50	108
	2	40x10	400	702	120	100	198

Table A-14. - Solution times for the two-dimensional problem*,
(nnodex) x (nnodey) = 10x10,
number of processors = 2

num time step	MEASURED TIME						
	T_{tot}	T_{cp}		T_{prep}		T_{cm}	
		[s]	[%]	[s]	[%]	[s]	[%]
1	0.99	0.12	12.12	0.865	87.37	0.0004	0.0404
10	2.19	1.32	60.27	0.865	39.50	0.0043	0.1963
20	3.61	2.74	75.90	0.865	23.96	0.0086	0.2382
100	14.92	14.01	93.90	0.865	5.80	0.0432	0.2895
1000	142.45	141.15	99.09	0.865	0.61	0.4320	0.3033

* Further discription of the problem can be found in Table A-13,
CASE=0 , num.of.proc=2

Table A-15. - Solution times for the two-dimensional problem*, (nnodex) x (nnodey) = 20x10, number of processors = 2

num time step	MEASURED TIME						
	T_{tot}	T_{cp}		T_{prep}		T_{cm}	
		[s]	[%]	[s]	[%]	[s]	[%]
1	2.06	0.24	11.65	1.817	88.20	0.0007	0.0340
10	4.41	2.59	58.73	1.817	41.20	0.0074	0.1678
20	7.25	5.42	74.76	1.817	25.06	0.0147	0.2028
100	29.97	28.08	93.69	1.817	6.06	0.0736	0.2456
1000	285.00	282.45	99.11	1.817	0.64	0.7360	0.2582

* Further discription of the problem can be found in Table A-13, CASE=1 , num.of.proc=2

Table A-16. - Solution times for the two-dimensional problem*,
(nnodex) x (nnodey) = 40x10,
number of processors = 2

num time step	MEASURED TIME						
	T_{tot}	T_{cp}		T_{prep}		T_{cm}	
		[s]	[%]	[s]	[%]	[s]	[%]
1	5.01	0.24	11.65	4.525	88.20	0.0013	0.0340
10	9.54	2.59	58.73	4.525	41.20	0.0134	0.1678
20	15.03	5.42	74.76	4.525	25.06	0.0269	0.2028
100	60.52	28.08	93.69	4.525	6.06	0.1344	0.2456
1000	572.30	282.45	99.11	4.525	0.64	1.3440	0.2582

* Further discription of the problem can be found in Table A-13,
CASE=2 , num.of.proc=2

Table A-17. - Solution times for the two-dimensional problem*,
(nnodex) x (nnodey) = 10x10,
number of processors = 4

num time step	MEASURED TIME						
	T_{tot}	T_{cp}		T_{prep}		T_{cm}	
		[s]	[%]	[s]	[%]	[s]	[%]
1	0.77	0.08	10.39	0.686	89.09	0.0006	0.0779
10	1.55	0.86	55.48	0.686	44.26	0.0058	0.3742
20	2.40	1.70	70.83	0.686	28.58	0.0115	0.4792
100	9.23	8.49	91.98	0.686	7.43	0.0576	0.6241
1000	86.13	84.87	98.54	0.686	0.80	0.5760	0.6688

* Further discription of the problem can be found in Table A-13,
CASE=0 , num.of.proc=4

Table A-18. - Solution times for the two-dimensional problem*,
(nnodex) x (nnodey) = 20x10,
number of processors = 4

num time step	MEASURED TIME						
	T_{tot}	T_{cp}		T_{prep}		T_{cm}	
		[s]	[%]	[s]	[%]	[s]	[%]
1	1.26	0.15	11.90	1.114	88.41	0.0010	0.0794
10	2.56	1.44	56.25	1.114	43.52	0.0102	0.3984
20	4.01	2.88	71.82	1.114	27.78	0.0205	0.5112
100	15.58	14.36	92.17	1.114	7.15	0.1024	0.6573
1000	145.73	143.59	98.53	1.114	0.76	1.0240	0.7027

* Further discription of the problem can be found in Table A-13, CASE=1 , num.of.proc=4

Table A-19. - Solution times for the two-dimensional problem*, (nnodex) x (nnodey) = 40x10, number of processors = 4

num time step	MEASURED TIME						
	T_{tot}	T_{cp}		T_{prep}		T_{cm}	
		[s]	[%]	[s]	[%]	[s]	[%]
1	2.55	0.27	10.59	2.283	89.53	0.0019	0.0745
10	4.95	2.65	53.54	2.283	46.12	0.0192	0.3879
20	7.79	5.47	70.22	2.283	29.31	0.0384	0.4929
100	30.51	28.04	91.90	2.283	7.48	0.1920	0.6293
1000	286.16	281.96	98.53	2.283	0.80	1.9200	0.6710

* Further discription of the problem can be found in Table A-13, CASE=2 , num.of.proc=4

Table A-20. -Parallel solution times for two-dimensional problems using two processors, T800 transputer[8]

T_{tot}[s]			
number of time steps	(nnodex) x (nnodey)		
	10 x 10	20 x 10	40 x 10
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 A-21. -Solution times for two-dimensional problems for various numbers of processors where the number of nodes per processor is fixed, T800 transputer[8]

T_{tot}[s]			
number of time steps	problem discription		
	10 x 10 num.of.proc=2	20 x 10 num.of.proc=4	40 x 10 num.of.proc=8
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

BIBLIOGRAPHY

BIBLIOGRAPHY

1. Zienkiewicz, O.C., The Finite Element Method. (Fourth Edition, New York: McGraw-Hill Company,1981).

2. Cook, Robert D., Malkus, David S., and Plesha, Michael E., Concepts and Applications of Finite Element Analysis, (Third Edition, New York: John Wiley & Sons).

3. Zienkiewicz, Op. Cit., pp. 464-465.

4. Burnett, David S., Finite Element Analysis, (Reading, Massachusetts: Addison-Wesley Publishing Company, 1987), pp.526-531.

5. Burnett, Op. Cit., pp. 467-482.

6. Pountain, Dick, and May, David, A Tutorial Introduction to OCCAM Programming, (London: BSP Professional Books, 1987).

7. Moss, Larry A., Smith, Todd E., SSME Single Crystal Turbine Blade Dynamics, NASA CR-179644, (Cleveland, Ohio: Lewis Research Center), p.16.

8. Smolinski, Patrick, Transient Finite Element Computations on the Transputers System, NASA CR-185199, (Pittsburgh, Pennsylvania: University of Pittsburgh, February, 1990).

REFRENCES NOT CITED

1. Belytschko, Ted, Hughes, Thomas, Computational Methods for Transient Analysis, Vol. 1, (Elsevier Science Publishers B.V., 1983).

2. Hughes, Thomas, Linear Static & Dynamic Finite Element Analysis. (Englewood Cliff, New Jersey: Prentice- Hall).

3. Janetzke, David C., Efficient Computation of Aerodynamic Influence Coefficients for Analysis on a Transputer Network, NASA TM-103671, (Cleveland, Ohio: Lewis Research Center, February, 1991).

4. Nour-Omid, B., and Park, K.C., "Solving Structural Mechanics Problems on the CalTech Hypercube Machine", Computer Methods in Applied Mechanics, Vol.61, (1987), pp.161-176.

5. Malone, James G., "Automated Mesh Decomposition and Concurrent Finite Element Analysis for Hypercube Multiprocessor Computers", Computer Methods in Applied Mechanics and Engineering, Vol.70, (1988), pp.27-58.

6. Timoshenko, Stephen P., Weaver, William, and Young, Donovan P., Vibration Problems in Engineering. (Fifth Edition, New York: Wiley-Interscience, 1990).

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	January 1993	Final Contractor Report

4. TITLE AND SUBTITLE
Transient Finite Element Computations on a Variable Transputer System

5. FUNDING NUMBERS
WU–505–63–1B
G–NAG3–1152

6. AUTHOR(S)
Patrick J. Smolinski and Ireneusz Lapczyk

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

8. PERFORMING ORGANIZATION REPORT NUMBER
E–7534

9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135–3191

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA CR–189060

11. SUPPLEMENTARY NOTES
Project Manager, David C. Janetzke, Structures Division, NASA Lewis Research Center, (216) 433–6041.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified - Unlimited
Subject Category 39

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)
In this study a parallel program to analyze transient finite element problems was written and implemented on a system of transputer processors. The program uses the explicit time integration algorithm which eliminates the need for equation solving making it more suitable for parallel computations. An interprocessor communication scheme was developed for arbitrary two-dimensional grid processor configurations. Several 3-D problems were analyzed on a system with a small number of processors.

14. SUBJECT TERMS
Finite element; Structural dynamics; Parallel computation; Transputers

15. NUMBER OF PAGES
82

16. PRICE CODE
A05

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	