GRANT/ LANGLEY

35P.

PARALLEL PROCESSORS AND NONLINEAR STRUCTURAL DYNAMICS ALGORITHMS AND SOFTWARE

IN-34279

Principal Investigator: Ted Belytschko

Final Technical Report

March 1, 1986 to October 30, 1986

Department of Civil Engineering
Northwestern University
Evanston, Illinois 60201

NASA Research Grant NAG-1-650

PREFACE

This research was conducted under the direction of Professor Ted Belytschko. Participating research assistants were Noreen Gilbertsen and Bruce Engelmann. The help of Argonne National Laboratory, particularly Dr. James Kennedy, who provided access to several parallel computing machines is also appreciated.

The following paper supported by NASA was accepted for publication:

P. Smolinski and T. Belytschko, "Multi-Time Step Integration Using Nodal Partitions," accepted for publication, International Journal for Numerical Methods in Engineering.

ABSTRACT

A nonlinear structural dynamics program with an element library that exploits parallel processing is under development. The aim is to exploit scheduling-allocation so that parallel processing and vectorization can effectively be treated in a general purpose program. As a byproduct an automatic scheme for assigning time steps was devised. A rudimentary form of the program is complete and has been tested: it shows substantial advantage can be taken of parallelism.

In addition, a stability proof for the subcycling algorithm has been developed.

1. INTRODUCTION

The purpose of this project is to develop a nonlinear structural dynamics computer program with a library of elements which can effectively exploit a computer with concurrent processing capabilities and vectorization. The program will be developed in a form that is readily adaptable to the NICE test bed of NASA Langley.

While substantial study has already been devoted to the treatment of large systems of partial differential equations (PDE) on MIMD (Multiple Instruction - Multiple Data Stream Processors), see Ref. [1] the problems posed in the parallelization of general purpose of finite element computer programs are substantially different. In large PDE systems, the computations associated with each node of a mesh are usually in essense quite similar and most of the concern lies with the spatial partitioning of the mesh so that optimum utilization of the parallel character of the computer is achieved.

On the other hand, in a general purpose finite element program, a major difficulty in parallelization is associated with the large variety of elements and constitutive models which are used in a computation. Thus, the most effective utilization of concurrent processing will usually not be that based on a simple subdivision of the spatial domain, since vectorization of a stream of dissimilar elements is almost impossible. The problem is further compounded by the practical necessity of post-processing the solution for graphic- display purposes in parallel with the computation. The practice of simply dumping all historical variables such as displacements, strains and stresses at all points into a storage device and then processing it subsequent to the calculation is usually unbelievably wasteful in a nonlinear structural dynamics computation because of the large amounts of time required for the communication and the large amounts of storage required.

Thus it can be seen that in a general purpose programs for nonlinear structural dynamics, the assignment - scheduling problem is quite crucial. Obviously, if one processor tends to lag behind the others, the benefits of a parallel architecture are quickly dissipated. As an example of the type of scheduling problem that arises in a typical nonlinear structural dynamics calculation, consider the nonlinear response of an elastic-plastic structure to an impulsive load. Initially the behavior of most of the structure will be elastic. However, as the deformation progresses, plastic response develops in some elements which will slow down the processing substantially. An efficient algorithm should be able to handle these changes in an effective manner without degrading performance.

In this work, an explicit method has been chosen for the time integration. The architecture of an explicit time integration program is such that it can readily be expanded to an implicit time integration program based on an iterative solver such as the conjugate gradient method. This class of algorithms is most readily adapted to concurrent processing machines because the bulk of the computation time is devoted to element level operations, namely, obtaining element nodal forces from element nodal displacements via the strain-displacement equations, the constitutive law, and the volume integration of the semi-discrete divergence of the stress tensor, that is, the computation of the internal nodal forces.

An effective implementation of an explicit time integration scheme in nonlinear structural dynamics requires that different time steps be used on different parts of the mesh. Otherwise, the presence of a few stiff elements in the model will entail the use of a small time step for the entire mesh. Therefore, the explicit time integration procedure in this project was designed so that different time steps could be used on different parts of the

mesh. Furthermore, in conjunction with the vectorization and parallelli-zation, the program is being designed to automatically assign time steps to different parts of the mesh. Previous versions of time step partitions always required the user to select the domains which would use different time steps and for the partition to remain fixed throughout the computation.

Although we had originally planned to implement our algorithms on a partitioned memory machine, initial planning indicated it would be better to take one step at a time and implement the algorithm in a shared memory machine. For this purpose, two computers which are available at Argonne National Laboratory were chosen: the Sequent Balance 8000/21000 and the Alliant. The implementation of parallel processing in these machines at the Argonne center was facilitated substantially by the availability of a macro library of monitors which enables programmers to write portable FORTRAN Code for multi-processors [2]. These monitors are very similar in content to the FORCE monitors developed by Jordan [3], so eventual translation of this program to the NICE test bed of NASA Langley should be readily accomplished.

As a framework of the initial studies, the program WHAMS, which is partially described in Ref. [4] and [5] was chosen. This program contains a 4-node quadrilateral shell element, a beam element, and a spring element so that the allocation problem among different elements could effectively be studied. It is a three dimensional program which treats both geometric and material nonlinearities.

An outline of the report is as follows. Section 2 describes the time integration algorithm. Section 3 gives some sample numerical results, followed by some conclusions. A stability proof for a model problem relevant to the subcycling procedure is given in the Appendix.

2. ALGORITHM DESCRIPTION

Central Difference Method

A finite element model of a nonlinear structure is governed by the following equations:

strain-displacement equations (in rate-form)

$$\underset{\sim}{D} = \underset{\sim}{B}\,\dot{\underset{\sim}{u}}_e \tag{1}$$

constitutive equations (in rate-form)

$$\dot{\underset{\sim}{T}} = \mathcal{J}(\underset{\sim}{T}, \underset{\sim}{D}) \tag{2}$$

momentum equations

$$\ddot{\underset{\sim}{u}} = \underset{\sim}{M}^{-1}\,(\underset{\sim}{f}_{ext} - \underset{\sim}{f}_{int}) \tag{3}$$

$$\underset{\sim}{f}_{int,e} = \int_{\Omega_e} \underset{\sim}{B}^T\,\underset{\sim}{T}\,d\Omega \tag{4}$$

In the above the following nomenclature has been used.

$\underset{\sim}{B}$ ...	strain-rate-velocity matrix
$\underset{\sim}{D}$ ...	velocity strains (strain rates)
$\underset{\sim}{f}_{int}$ ...	internal nodal forces
$\underset{\sim}{f}_{ext}$ ...	external nodal forces
$\underset{\sim}{M}$ ...	mass matrix (assumed diagonal and lumped)
$\underset{\sim}{T}$, $\overset{\nabla}{\underset{\sim}{T}}$ ...	Cauchy stress matrix and its frame invariant rate
$\underset{\sim}{u}$, $\dot{\underset{\sim}{u}}$, $\ddot{\underset{\sim}{u}}$ ...	nodal displacements, velocities and accelerations, respectively

Ω_e ... the domain of element e

$\mathcal{J}$... the constitutive equation (or algorithm)

The subscript e will always indicate element-related variables.

The central difference method uses the following equations to update the nodal variables

$$\underset{\sim}{\dot{u}}^{n+1/2} = \underset{\sim}{\dot{u}}^{n-1/2} + \Delta t\, \underset{\sim}{\ddot{u}}^{n} \tag{5}$$

$$\underset{\sim}{u}^{n+1} = \underset{\sim}{u}^{n} + \Delta t\, \underset{\sim}{\dot{u}}^{n+1/2} \tag{6}$$

In the above, superscripts designate time steps.

An outline of the algorithm for explicit time integration is given in Table 1. As can be seen from the flow chart, the major opportunity for parallelization appears in the loop over the elements. The number of multiplications per element can vary from 50 to the order of 10^3. Here we have a coarse-grained parallelism which is ideal for concurrent processors. However, if the parallelism is exploited on an element level, then the opportunities for any significant vectorization are lost. To exploit vectorization in conjunction with parallelism, it is necessary to arrange the elements in groups. The number of groups should be greater than the number of processors, but the size of each group is limited also by the auxiliary arrays which are needed for vectorization. Furthermore, for an efficient implementation of vectorization, only one type of element can occur in a group.

Table 1

Outline of Explicit Algorithms

1. Initial conditions: $\dot{\underset{\sim}{u}}^{-1/2}$, $\underset{\sim}{u}^{0}$
2. Update nodal velocities and displacements by Eqs. (5) and (6)
3. Loop over all NE elements

   Loop over all NG quadrature points in element

   Compute velocity strains by Eq. (1)

   Evaluate constitutive law, Eq. (2)

   Evaluate $\underset{\sim}{B}^{T}\ \underset{\sim}{T}$ and add to integrand of Eq. (4)

   end of loop over quadrature points

   add $\underset{\sim}{f}_{int,e}$ to total $\underset{\sim}{f}_{int}$ array

   end of loop over elements
4. Compute accelerations by Eq. (3); go to 2

Explicit-Explicit Partition

The concurrent procedure descrubed here is based on an explicit-explicit partitioning procedure, or subcycling procedure, first presented in [9,10]. In this method, the elements are separated into element groups, each of which can be integrated with a different time step subject to the following restrictions:

1. The largest step must be an integer multiple of all time steps.
2. If any node is shared by elements in two different integration groups, the time steps in these groups must be integer multiples of each other.

In adapting this method to a parallel-vector code, it was decided to make the alignment between the element groupings for vectorization and the time

steps partitioning coincident. Thus any element group is automatically integrated by the same time step. The element grouping is performed by a preprocessor.

For the purpose of defining how the explicit-explicit partitioning procedure works, we will define the following variables.

NTGRP: number of groups into which the finite element mesh is subdivided

Δt_G: the time increment for element group G

t_{Mast}: the master time

Δt_{Mast}: the master time increment, which corresponds to the minimum Δt_G among all element groups G

Δt_G: the time increment for element group G.

As stated previously, all element time step increments Δt_G must be integer divisors of Δt_{Mast}. The maximum Δt_G is called Δt_{max}.

The program is designed so that it automatically decides the appropriate nodal time step. This is accomplished by using the largest time increment for any element group connected to the node to update the node. In order to program this algorithm, each node therefore requires two additional words of storage: the nodal time t_N and the time increment used for that node, Δt_N.

The essence of the procedure is as follows. We call the time steps necessary to advance the master clock by Δt_{max} a cycle. Within a cycle, whenever the master clock t_{Mast} is incremented by Δt_{Mast}, all element groups are first checked. If any element group is not ahead of the master time, i.e. if for group G

$$t_G < t_{Mast}$$

that element group is updated. This updating involves the calculation of new velocity strains, stresses and internal forces. The element internal forces are then added into the global internal force matrix.

After all element groups have been updated to time t_{Mast}, the nodal loop for updating velocities and displacements is executed. In this loop, prior to updating the velocities and displacements, the nodal clock t_N is checked. If

$$t_N < t_{Mast}$$

the nodal clock is behind the master clock, so the node is updated. In addition, the nodal clock is updated using the time increment for that node.

The algorithm assumes that a velocity strain formulation is used for all element calculations. When an element needs to be updated, the latest available velocity is used to compute the velocity strain. This means if an element is connected to a node with a larger time step, it uses the same nodal velocity for all intermediate time steps. This corresponds to a constant velocity interpolation or a linear displacement interpolation, which has been shown to be stable. If displacements are needed by an element at an intermediate time step, linear interpolation based on the last cycle displacement with the current velocity as a slope is used.

A flow chart for the procedure is shown in Table 2.

Table 2

Flowchart of Partitioned Explicit Algorithm

1. Set initial condition

$$\underset{\sim}{u}^0 = \underset{\sim}{u}(0) \ , \quad \underset{\sim}{u}^{-1/2} = \dot{\underset{\sim}{u}}(0) \ , \quad n = 0 \ , \quad n2 = 0$$

initial accelerations are assumed to vanish $\ddot{\underset{\sim}{u}}^0 = 0$.

2. Initialize clocks and cycle counters

$t_{Mast} = 0$ master time

$t_G = 0$ for all element groups G

$t_N = 0$ for all nodes N

3. If n = 0 set, nodal time ($\underset{\sim}{f} = 0$) increments Δt_N, subcycle counter n2 = 0

4. Update nodes with clocks behind the master clock, i.e. $t_N < t_{Mast}$

   a. DO N = 1 to NNODE

   b. if $t_N > t_{Mast}$, skip node N

   c. compute accelerations $\ddot{\underset{\sim}{u}}_N^n = \underset{\sim}{M}^{-1} \underset{\sim}{f}$

   d. update nodal velocities: $\dot{\underset{\sim}{u}}_N^{n+1/2} = \dot{\underset{\sim}{u}}_N^{n-1/2} + \Delta t_N \ddot{\underset{\sim}{u}}_N^n$

   e. update nodal displacements: $\underset{\sim}{u}_N^{n+1} = \underset{\sim}{u}_N^n + \Delta t_N \dot{\underset{\sim}{u}}_N^{n+1/2}$

   f. update nodal clock: $t_N \leftarrow t_N + \Delta t_N$

5. Compute internal forces $\underset{\sim}{f}_{int}^{n+1}$

   a. zero $\underset{\sim}{f}_{int}$

   b. if $t_G > t_{Mast}$, loop over element G

c. compute velocity strains: $\underset{\sim}{D}_N^{n+1/2} = \underset{\sim}{B}\ \dot{\underset{\sim}{u}}_N^{n+1/2}$

d. compute velocity strains: $\underset{\sim}{D}_N^{n+1/2} = \underset{\sim}{B}\ \dot{\underset{\sim}{u}}_N^{n+1/2}$

e. update stress: $\underset{\sim}{T}_N^{n+1} = \underset{\sim}{T}_N^{n} + \Delta t_G\ \dot{\underset{\sim}{T}}_N^{n+1/2}$

f. compute element internal forces: $\underset{\sim}{f}_{int,N}^{n+1} + \int_{V_N} \underset{\sim}{B}^T\ T_N^{n+1}\ dV$

g. if n2 = 1, compute stable time increment for element

h. assemble $\underset{\sim}{f}_{int,\ N}^{n+1}$ into $\underset{\sim}{f}_{int}^{n+1}$

6. Update element group clocks t_G and n2

7. Compute $\underset{\sim}{f}^{ext}$

8. If $n2 = n2_{max}$, set new element group time increment Δt_G

3. NUMERICAL RESULTS

At the present time, a version of WHAMS has been developed which includes beam and plate elements and a rudimentary form of parallelization. However, the conversion of the subroutines to take full advantage of vectorization has not been completed nor has the programming of the subcycling been completed.

We will report results for 2 problems which have been solved. The first is a spherical cap problem; the problem description and mesh are shown in Table 3 and Fig. 1, respectively. A total of 91 nodes and 75 elements were used for the one-quarter model. A uniform step load was applied over the cap. The response compouted here is shown in Fig. 2 where it is compared to results given in Ref. [6].

The computer times for various computer and various degrees of concurrency are reported in Table 4. In addition to the concurrency which is incorporated in the program, solution times for concurrency as developed by the compiler are given.

The second problem which has been solved is shown in Fig. 3. In this case, one and two processors were used for the solution. The speedup is going from 1 to 2 processors in this case is 1.63, which is 80% of the theoretical speedup.

Table 3.

Material Properties and Parameters for Spherical Cap Problem

Radius	$r = 22.27$ in
Thickness	$t = 0.41$ in
Angle	$\alpha = 26.67°$
Density	$\rho = 2.45 \times 10^{-4}$ lb-sec^2/in^4
Young's modulus	$E = 1.05 \times 10^7$ psi
Poisson's ratio	$\nu = 0.3$
Yield stress	$\sigma_Y = 2.4 \times 10^4$ psi
Plastic modulus	$E_p = 2.1 \times 10^5$ psi
Pressure load	$P = 600$ psi

Table 4

Solution times for spherical cap problem 1000 time steps

ALLIANT (1 processor)	310.9 sec
ALLIANT (compiler assigned concurrency on 8 processors)	116.5 sec
ALLIANT (8 processors; programmer designed concurrency)	65 sec
VAX 11/780	901.8 sec
IBM 3033	75 sec

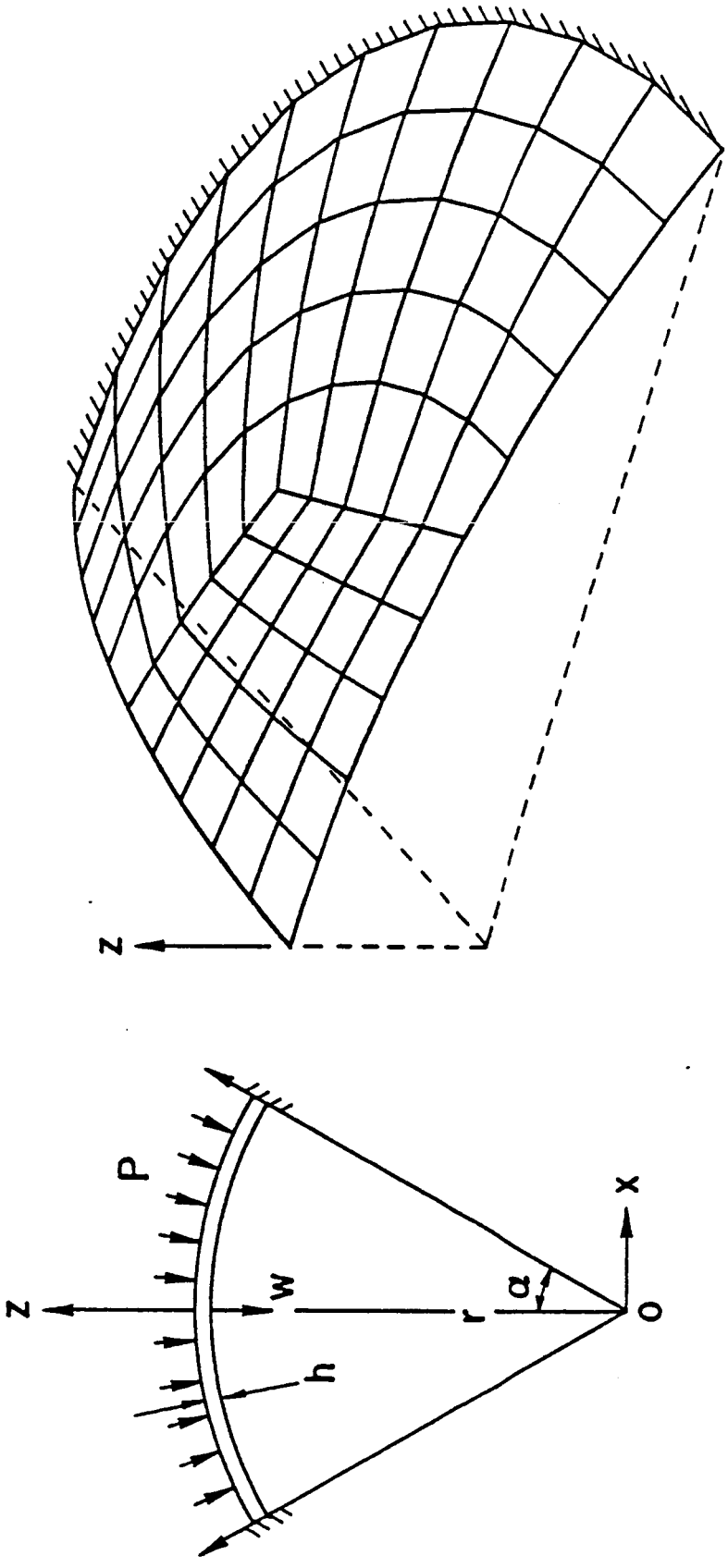

Figure 1. Clamped Spherical Cap and Element Mesh.

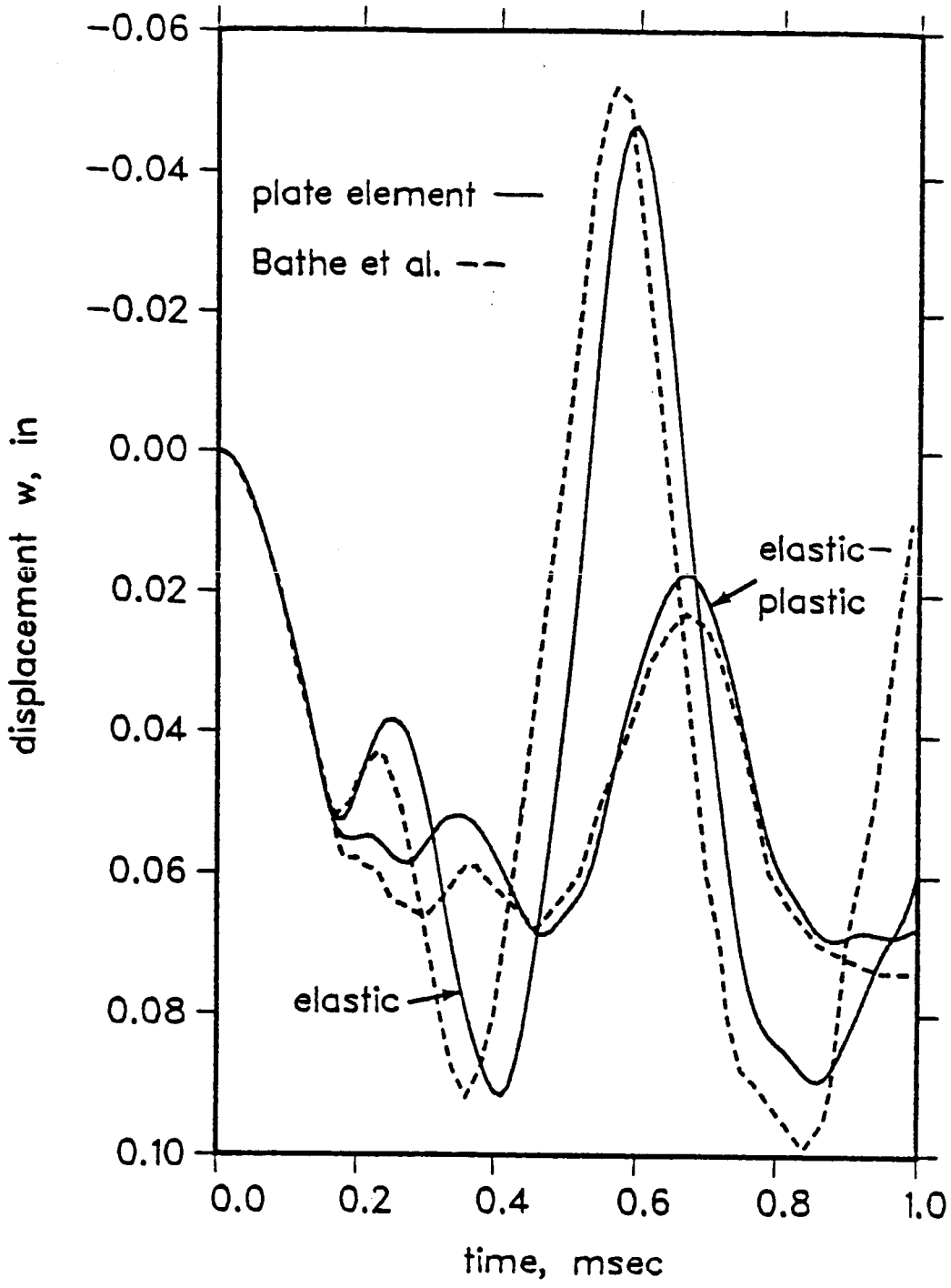

Figure 2. Center Displacement of Spherical Cap for Elastic and Elastic-Plastic Materials.

7 x 19 4 node plate elements

76 beam elements (stiffeners)

960 degrees-of-freedom

1000 time steps

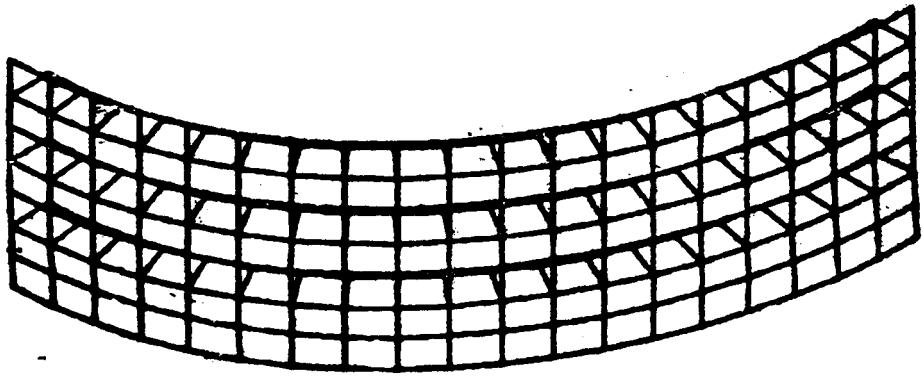

Figure 3. Stiffened Shell Problem.

CONCLUSIONS

A simple algorithm has been developed which permits a partition of time steps, so that different processors can run different groups of elements with different time steps. In addition, the method has been devised so that it can take advantage of vectorization and is quite automatic. A rudimentary form of the program has been completed and tested. It does not yet include vectorization and the complete subcycling algorithm. Several examples run on the Alliant computer show that the resulting algorithm can take relatively good advantage of concurrent processing.

In addition, the stability of these numerical procedures has been studied. A proof of stability has been developed for linear first-order systems when nodal partitions are used.

REFERENCES

1. U. Schendel, "Introduction to Numerical Methods for Parallel Computers," Ellis - Horwood, 1984.

2. E.L. Lusk and R.A. Overbeek, "Use of Monitors in FORTRAN: A Tutorial on the Barrier, Self-scheduling Do-Loop and Ask for Monitors," Report ANL-84-51, Rev. 1, Argonne Natinal Laboratory, Argonne, IL.

3. H.F. Jordan, "Structuring Parallel Algorithms in a MIMD Shared Memory Environment," Parallel Computing, 3(2), 93-110, May 1986.

4. T. Belytschko, L. Schwer and M. Klein, "Large Displacement, Transient Analysis of Space Frames," International Journal for Numerical Methods in Engineering 11, 65-84, 1977.

5. T. Belytschko, J.I. Lin and C.S. Tsay, "Explicit Algorithms for the Nonlinear Dynamics of Shell," Computer Methods in Applied Mechanics and Engineering, 42, 225-251, 1984.

6. K.J. Bathe, E. Ramm and E.L. Wilson, "Finite Element Formulations for Large Deformation Dynamic Analysis," International Journal for Numerical Methods in Engineering, 9, 353-386, 1975.

APPENDIX

MULTI-TIME STEP INTEGRATION USING NODAL PARTITIONING

1. INTRODUCTION

In many engineering models that are composed of non-uniform meshes oftentimes a group of small or stiff elements forces the integration of the remainder of the mesh with a time step much smaller that its stable time step. To alleviate this difficulty subcycling, which uses different time steps in different portions of the mesh, has been developed. This eliminates the need to update the entire mesh with the stable time step of the smallest element, and requires much less additional programming in comparison to implicit-explicit time integration [1-3].

In contrast to implicit-explicit integration, which has been analyzed by a variety of methods [3-5], there has been little done in the stability analysis of subcycling. One reason is that the multiple time steps introduce special difficulties not found in implicit-explicit integration with its single time step. Early stability analyses have employed simplifying assumptions [6] or failed to show that a complete set of real eigenvalues existed [7]. The first rigorous proof of stability is given for a subcycling algorithm which uses the same integration rule in the different partitions [8]. This scheme is modified [9] to use different values of α in the α-algorithm along with different time steps in the different subdomains. The integration schemes in [8-10] are based on an element partition while the algorithm in [7] is based on a nodal partition. One major difference between these two types of partitions is that the nodal partition results in unsymmetric amplification matrices which complicate the analysis.

In this paper a stability analysis of a subcycling method which uses different values of α and a nodal partition is presented. Two features of this method are that no unsymmetric systems need be solved and that a simple procedure is used to establish sufficient conditions for stability. The element eigenvalue inequality theorem is used to bound the critical time step in terms of element eigenvalues.

2. MIXED TIME INTEGRATION PROCEDURE

The matrix equation governing linear diffusion processes is

$$\underset{\sim}{M}\dot{\underset{\sim}{u}} + \underset{\sim}{K}\underset{\sim}{u} = \underset{\sim}{s} \tag{2.1}$$

where using the nomenclature of heat conduction $\underset{\sim}{M}$ is the capacitance matrix, and $\underset{\sim}{K}$ is the conductance matrix. The vectors $\underset{\sim}{u}$ and $\underset{\sim}{s}$ represent the nodal values of the dependent variables and source, respectively. A superposed dot denotes a time derivative. The matrix $\underset{\sim}{M}$ is assumed to be diagonal or lumped and is positive definite, and the matrix $\underset{\sim}{K}$ is symmetric and positive semidefinite. The problem consists of finding a function $\underset{\sim}{u} = \underset{\sim}{u}(t)$ which satisfies governing equation (2.1) and the initial condition

$$\underset{\sim}{u}(t = 0) = \underset{\sim}{u}^{o} \tag{2.2}$$

for all time t, where $\underset{\sim}{u}^{o}$ is a given vector.

To develop the integration algorithm the linear one step integrator is used, which is given by

$$\underset{\sim}{u}^{n+1} = \underset{\sim}{u}^{n} + (1 - \alpha)\Delta t\, \dot{\underset{\sim}{u}}^{n} + \alpha\Delta t\, \dot{\underset{\sim}{u}}^{n+1} \tag{2.3}$$

where $\underset{\sim}{u}^{n} = \underset{\sim}{u}(n\Delta t)$ and Δt is the time step. If the parameter α is equal to zero, the integration is called explicit because $\underset{\sim}{u}^{n+1}$

can be computed from historical data. Explicit integration is only conditionally stable, thus restricting the time step. For $\alpha > 0$, the integration is called implicit with no restriction on Δt if $\alpha \geq \frac{1}{2}$.

For the purpose of introducing the mixed time integration procedure, the vector $\underset{\sim}{u}$ is partitioned as

$$\underset{\sim}{u} = \begin{Bmatrix} \underset{\sim}{u}_A \\ \\ \underset{\sim}{u}_B \end{Bmatrix} \begin{matrix} N_A \text{ rows} \\ \\ N_B \text{ rows} \end{matrix} \tag{2.4}$$

where the nodal groups A and B are integrated with time steps $m\Delta t$ and Δt, respectively, and where m is an integer such that $m \geq 1$. Since with this method $m + 1$ updates are needed to advance the solution one cycle from t to $t + m\Delta t$, a counter k, which is set to zero at the start of each integration cycle and increased by one after every update, is used to keep track of the updates. The integration cycle is developed by writing Eq. (2.3) as

$$\underset{\sim}{u}^{k+1} = \underset{\sim}{u}^{k} + \Delta t \underset{\sim}{W}_1^k \dot{\underset{\sim}{u}}^{k} + \Delta t \underset{\sim}{W}_2^k \dot{\underset{\sim}{u}}^{k+1} \tag{2.5}$$

where

$$\underset{\sim}{W}_1^k = \begin{bmatrix} \underset{\sim}{0} & \underset{\sim}{0} \\ \\ \underset{\sim}{0} & (1 - \alpha_B)\underset{\sim}{I} \end{bmatrix} \qquad \underset{\sim}{W}_2^k = \begin{bmatrix} \underset{\sim}{0} & \underset{\sim}{0} \\ \\ \underset{\sim}{0} & \alpha_B \underset{\sim}{I} \end{bmatrix} \quad \text{for } k \neq m \tag{2.6}$$

$$\underset{\sim}{W}_1^k = \begin{bmatrix} m(1 - \alpha_A)\underset{\sim}{I} & \underset{\sim}{0} \\ \\ \underset{\sim}{0} & \underset{\sim}{0} \end{bmatrix} \qquad \underset{\sim}{W}_2^k = \begin{bmatrix} m\alpha_A \underset{\sim}{I} & \underset{\sim}{0} \\ \\ \underset{\sim}{0} & \underset{\sim}{0} \end{bmatrix} \quad \text{for } k = m \tag{2.7}$$

Here $\underset{\sim}{I}$ is the unit matrix, and the $\underset{\sim}{W}$ matrices are partitioned similar to $\underset{\sim}{u}$.

To begin the integration cycle, equation (2.6) is used in conjunction with Eq. (2.5) to subcycle the solution m times in partition B with a time step Δt. To finish the cycle, nodal partition A is updated with the large time step $m\Delta t$ using Eqs. (2.5) and (2.7). After this process is complete, the solution has been advanced by $m\Delta t$ in both partitions. The parameters α_A and α_B determine the integration algorithm for each partition.

Considering the homogeneous case ($\underset{\sim}{s} = \underset{\sim}{0}$), since this is sufficient to examine stability, equation (2.1) is solved for $\dot{\underset{\sim}{u}}$ and substituted into Eq. (2.5) to yield after some manipulation

$$(\underset{\sim}{M} + \Delta t \underset{\sim}{W}_2^k \underset{\sim}{K})\underset{\sim}{u}^{k+1} = (\underset{\sim}{M} - \Delta t \underset{\sim}{W}_1^k \underset{\sim}{K})\underset{\sim}{u}^k \tag{2.8}$$

or

$$\underset{\sim}{A}_2^k \underset{\sim}{u}^{k+1} = \underset{\sim}{A}_1^k \underset{\sim}{u}^k \tag{2.9}$$

where

$$\underset{\sim}{A}_2^k = \underset{\sim}{M} + \Delta t \underset{\sim}{W}_2^k \underset{\sim}{K} \tag{2.10a}$$

$$\underset{\sim}{A}_1^k = \underset{\sim}{M} - \Delta t \underset{\sim}{W}_1^k \underset{\sim}{K} \tag{2.10b}$$

Using the definitions of $\underset{\sim}{W}$ and the above equations, we find that for k = 0 to m - 1:

$$\underset{\sim}{A}_2^k = \begin{bmatrix} \underset{\sim}{M}_A & \underset{\sim}{0} \\ \Delta t \alpha_B \underset{\sim}{K}_{BA} & \underset{\sim}{M}_B + \Delta t \alpha_B \underset{\sim}{K}_{BB} \end{bmatrix} \tag{2.11a}$$

$$\underset{\sim}{A}_1^k = \begin{bmatrix} \underset{\sim}{M}_A & \underset{\sim}{0} \\ -\Delta t(1 - \alpha_B)\underset{\sim}{K}_{BA} & \underset{\sim}{M}_B - \Delta t(1 - \alpha_B)\underset{\sim}{K}_{BB} \end{bmatrix} \qquad (2.11b)$$

and similarly for k = m if it can be shown that

$$\underset{\sim}{A}_2^k = \begin{bmatrix} \underset{\sim}{M}_A + m\Delta t\alpha_A\underset{\sim}{K}_{AA} & m\Delta t\alpha_A\underset{\sim}{K}_{AB} \\ \underset{\sim}{0} & \underset{\sim}{M}_B \end{bmatrix} \qquad (2.12a)$$

$$\underset{\sim}{A}_1^k = \begin{bmatrix} \underset{\sim}{M}_A - m\Delta t(1 - \alpha_A)\underset{\sim}{K}_{AA} & -m\Delta t(1 - \alpha_A)\underset{\sim}{K}_{AB} \\ \underset{\sim}{0} & \underset{\sim}{M}_B \end{bmatrix} \qquad (2.12b)$$

Note that even though the matrices $\underset{\sim}{A}_2^k$ are not symmetric, no unsymmetric systems need be solved since the partition containing only the capacitance matrix is evaluated prior to solving for the remaining nodal partition. For more details about the implementation of this method see [7] in which a similar method is presented.

3. STABILITY ANALYSIS

To investigate the stability characteristics of the evolution equation (2.9) it is necessary to examine the associated generalized eigenvalue problem, which can be written as

$$\underset{\sim}{A}_1^k \underset{\sim}{x} = \lambda \underset{\sim}{A}_2^k \underset{\sim}{x} \qquad (3.1)$$

Considering first the subcycle update, equation (2.11) is substituted into the above equation and with some rearrangement gives

$$\begin{bmatrix} \underset{\sim}{0} & \underset{\sim}{0} \\ \underset{\sim}{K}_{BA} & \underset{\sim}{K}_{BB} \end{bmatrix} \underset{\sim}{x} = \frac{(1 - \lambda)}{\Delta t(1 - \alpha_B + \Delta t \alpha_B)} \begin{bmatrix} \underset{\sim}{M}_A & \underset{\sim}{0} \\ \underset{\sim}{0} & \underset{\sim}{M}_B \end{bmatrix} \underset{\sim}{x} \qquad (3.2)$$

By defining

$$\mu = \frac{(1 - \lambda)}{\Delta t(1 - \alpha_B + \Delta t \alpha_B)} \qquad (3.3)$$

and partitioning $\underset{\sim}{x}$ as

$$\underset{\sim}{x} = \begin{Bmatrix} \underset{\sim}{y} \\ \underset{\sim}{z} \end{Bmatrix} \qquad (3.4)$$

it follows that

$$\begin{bmatrix} \underset{\sim}{0} & \underset{\sim}{0} \\ \underset{\sim}{K}_{BA} & \underset{\sim}{K}_{BB} \end{bmatrix} \begin{Bmatrix} \underset{\sim}{y} \\ \underset{\sim}{z} \end{Bmatrix} = \mu \begin{bmatrix} \underset{\sim}{M}_A & \underset{\sim}{0} \\ \underset{\sim}{0} & \underset{\sim}{M}_B \end{bmatrix} \begin{Bmatrix} \underset{\sim}{y} \\ \underset{\sim}{z} \end{Bmatrix} \qquad (3.4)$$

where the dimension of the problem is N. Writing out the above equations gives

$$\underset{\sim}{0} = \mu \, \underset{\sim}{M}_A \, \underset{\sim}{y} \qquad (3.5a)$$

$$\underset{\sim}{K}_{BA} \, \underset{\sim}{y} + \underset{\sim}{K}_{BB} \, \underset{\sim}{z} = \mu \, \underset{\sim}{M}_B \, \underset{\sim}{z} \qquad (3.5b)$$

Due to the positive definiteness of $\underset{\sim}{M}_A$, the first equation implies either $\mu = 0$ or $\underset{\sim}{y} = \underset{\sim}{0}$. Assume that for $i = 1$ to N_B, $\underset{\sim}{y} = \underset{\sim}{0}$, while for $i = N_B + 1$ to N, $\mu_i = 0$. We now consider

CASE I $\quad \underset{\sim}{y} = \underset{\sim}{0} \qquad i = 1$ to N_B

The second of the above equations gives

$$\underset{\sim}{K}_{BB}\, \underset{\sim}{z}_i = \mu_i\, \underset{\sim}{M}_B\, \underset{\sim}{z}_i \qquad i = 1 \text{ to } N_B \tag{3.6}$$

Here N_B is the dimension of this reduced problem, obtained by deleting rows and columns from the standard matrices. Note that $\underset{\sim}{K}_{BB}$ and $\underset{\sim}{M}_B$ are symmetric, so that $\underset{\sim}{z}_i$ may be orthogonalized with respect to $\underset{\sim}{K}_{BB}$ and $\underset{\sim}{M}_B$. Also, since $\underset{\sim}{K}_{BB}$ is a constrained version of $\underset{\sim}{K}$, $\underset{\sim}{K}_{BB}$ is not singular and $\underset{\sim}{K}_{BB}^{-1}$ exists. Thus $\underset{\sim}{x}_i$ spans R^{N_B} and is given by

$$\underset{\sim}{x}_i = \begin{Bmatrix} \underset{\sim}{y}_i \\ \\ \underset{\sim}{z}_i \end{Bmatrix} = \begin{Bmatrix} \underset{\sim}{0} \\ \\ \underset{\sim}{z}_i \end{Bmatrix} \qquad i = 1 \text{ to } N_B \tag{3.7}$$

CASE II $\quad \mu_i = 0 \qquad i = N_B + 1, N$

Then the second equation gives

$$\underset{\sim}{K}_{BA}\, \underset{\sim}{y}_i + \underset{\sim}{K}_{BB}\, \underset{\sim}{z}_i = \underset{\sim}{0} \qquad i = N_B + 1, N \tag{3.8}$$

or

$$\underset{\sim}{z}_i = \underset{\sim}{K}_{BB}^{-1}\, \underset{\sim}{K}_{BA}\, \underset{\sim}{y}_i \qquad i = N_B + 1, N \tag{3.9}$$

A set of vectors, $\underset{\sim}{y}_i (i = N_B + 1, N)$, can be choosen which span R^{N_A} and the corresponding $\underset{\sim}{z}_i$ is given by Eq. (3.9). So the eigenvectors can be written as

$$\underset{\sim}{x}_i = \begin{Bmatrix} \underset{\sim}{y}_i \\ \\ \underset{\sim}{z}_i \end{Bmatrix} = \begin{Bmatrix} \underset{\sim}{y}_i \\ \\ -\underset{\sim}{K}_{BB}^{-1} \underset{\sim}{K}_{BA} \underset{\sim}{y}_i \end{Bmatrix} \quad i = N_B + 1 \text{ to } N \qquad (3.10)$$

Since $N = N_A + N_B$ the set of vectors given by Eqs. (3.7) and (3.10) span R^N. Thus, two possible sets of eigenvectors and eigenvalues are given by

$$\underset{\sim}{x}_i = \begin{Bmatrix} \underset{\sim}{0} \\ \\ \underset{\sim}{z}_i \end{Bmatrix}, \qquad \mu_i \neq 0 \quad i = 1 \text{ to } N_B \qquad (3.11a)$$

$$\bar{\underset{\sim}{x}}_i = \begin{Bmatrix} \underset{\sim}{y}_i \\ \\ -\underset{\sim}{K}_{BB}^{-1} \underset{\sim}{K}_{BA} \underset{\sim}{y}_i \end{Bmatrix}, \qquad \mu_i = 0 \quad i = N_B + 1 \text{ to } N \qquad (3.11b)$$

where the bar designates the eigenvectors corresponding to $\mu = 0$. The vectors $\underset{\sim}{x}_i$ are orthogonal to each other and to $\bar{\underset{\sim}{x}}_i$ with respect to $\underset{\sim}{K}$. To show the first fact, $\underset{\sim}{x}_j^T \underset{\sim}{K} \underset{\sim}{x}_i$ is written as

$$\{\underset{\sim}{0}^T, \underset{\sim}{z}_j^T\} \begin{bmatrix} \underset{\sim}{K}_{AA} & \underset{\sim}{K}_{AB} \\ \\ \underset{\sim}{K}_{BA} & \underset{\sim}{K}_{BB} \end{bmatrix} \begin{Bmatrix} \underset{\sim}{0} \\ \\ \underset{\sim}{z}_i \end{Bmatrix} = \underset{\sim}{z}_j^T \underset{\sim}{K}_{BB} \underset{\sim}{z}_i \qquad \begin{matrix} i,j = 1 \text{ to } N_B \\ i \neq j \end{matrix} \qquad (3.12a)$$

$$= \delta_{ij} \mu_i \qquad (3.12b)$$

since $\underset{\sim}{z}_i$ is orthogonal with respect to $\underset{\sim}{K}_{BB}$. The second condition is shown by considering

$$\underset{\sim}{x}_i^T K \bar{\underset{\sim}{x}}_j = \{\underset{\sim}{0}^T,\ \underset{\sim}{z}_i^T\} \begin{bmatrix} \underset{\sim}{K}_{AA} & \underset{\sim}{K}_{AB} \\ \underset{\sim}{K}_{BA} & \underset{\sim}{K}_{BB} \end{bmatrix} \begin{Bmatrix} \underset{\sim}{y}_j \\ -\underset{\sim}{K}_{BB}^{-1}\underset{\sim}{K}_{BA}\underset{\sim}{y}_j \end{Bmatrix} \quad \begin{matrix} i = 1 \text{ to } N_B \\ j = N_B+1 \text{ to } N \end{matrix} \tag{3.13a}$$

$$= \underset{\sim}{z}_i^T \underset{\sim}{K}_{BA} \underset{\sim}{y}_j - \underset{\sim}{z}_i^T \underset{\sim}{K}_{BB} \underset{\sim}{K}_{BB}^{-1} \underset{\sim}{K}_{BA} \underset{\sim}{y}_j \tag{3.13b}$$

$$= 0 \tag{3.13c}$$

Recalling that from Eq. (3.3)

$$\mu_i = \frac{(1 - \lambda_i)}{\Delta t(1 - \alpha_B + \lambda_i \alpha_B)} \qquad i = 1 \text{ to } N$$

or

$$\lambda_i = \frac{1 + \Delta t\ \mu_i(\alpha_B - 1)}{(1 + \Delta t \alpha_B \mu_i)} \qquad i = 1 \text{ to } N \tag{3.13}$$

For the time being, we will assume that stability will be defined as $|\lambda_i| \leq 1$ or from the above equation

$$-1 \leq \frac{1 + \Delta t\ \mu_i(\alpha_B - 1)}{1 + \Delta t\ \alpha_B \mu_i} \leq 1 \tag{3.14}$$

for all i = 1 to N. However, if $\mu_i \geq 0$ then Eq. (3.14) reduces to

$$\Delta t \leq \frac{2}{\mu_i(1 - 2\alpha_B)} \tag{3.15}$$

Note that if $\mu_i = 0$, Eq. (3.13) implies

$$\lambda_i = 1 \tag{3.16}$$

The analysis of the large time step update (k = m) proceeds by substituting Eq. (2.12) into Eq. (3.1) and rearranging to give

$$\begin{bmatrix} \underset{\sim}{K}_{AA} & \underset{\sim}{K}_{AB} \\ \underset{\sim}{0} & \underset{\sim}{0} \end{bmatrix} \begin{Bmatrix} \underset{\sim}{y}' \\ \underset{\sim}{z}' \end{Bmatrix} = \mu' \begin{bmatrix} \underset{\sim}{M}_A & \underset{\sim}{0} \\ \underset{\sim}{0} & \underset{\sim}{M}_B \end{bmatrix} \begin{Bmatrix} y' \\ \underset{\sim}{z}' \end{Bmatrix} \tag{3.17}$$

where

$$\mu' = \frac{(1 - \lambda')}{m\Delta t(1 - \alpha_A + \lambda'\alpha_A)} \tag{3.18}$$

and where the prime designates quantities associated with this update. By arguments similar to those used in the previous update analysis, two possible sets of eigenvectors, which span R^N, and their corresponding eigenvalues can be shown to be

$$\underset{\sim}{x}_i' = \begin{Bmatrix} \underset{\sim}{y}_i' \\ \underset{\sim}{z}_i' \end{Bmatrix} = \begin{Bmatrix} \underset{\sim}{y}_i' \\ \underset{\sim}{0} \end{Bmatrix}, \quad \mu_i' \neq 0 \qquad i = 1 \text{ to } N_A \tag{3.19a}$$

$$\underset{\sim}{\bar{x}}_i' = \begin{Bmatrix} \underset{\sim}{y}_i' \\ \underset{\sim}{z}_i' \end{Bmatrix} = \begin{Bmatrix} -\underset{\sim}{K}_{AA}^{-1}\underset{\sim}{K}_{AB}\underset{\sim}{z}_i' \\ \underset{\sim}{z}_i' \end{Bmatrix}, \quad \mu_i' = 0 \quad i = N_A+1 \text{ to } N \tag{3.19b}$$

where $\underset{\sim}{z}_i'(i = N_A + 1 \text{ to } N)$ are choosen to span R^{N_B} and $\mu_i'(i = 1 \text{ to } N_A)$ is calculated from

$$\underset{\sim}{K}_{AA}\ \underset{\sim}{y}_i' = \mu_i'\ \underset{\sim}{M}_A\ \underset{\sim}{y}_i' \tag{3.20}$$

Also, the set of vectors $\underset{\sim}{x}_i'$ can be shown to be orthogonal to each other and $\underset{\sim}{\bar{x}}_i'$ with respect to $\underset{\sim}{K}$.

From Eq. (3.18) we have that

$$\lambda_i' = \frac{1 + m\Delta t\ \mu_i'(\alpha_A - 1)}{(1 + m\Delta t\alpha_A\mu_i')} \qquad i = 1 \text{ to } N \tag{3.21}$$

Using the stability criterion $|\lambda_i'| \leq 1$ and assuming that $\mu_i' \geq 0$, Eq. (3.21) gives

$$\Delta t \leq \frac{2}{m(1 - 2\alpha_A)\mu_i'} \qquad i = 1 \text{ to } N \tag{3.22}$$

and Eq. (3.21) states that $\lambda_i' = 1$ for $\mu_i' = 0$.

To analyze the updating procedure, the solution at any arbitrary time is expanded in terms of the eigenvectors as

$$\underset{\sim}{d}^k = \beta_i \underset{\sim}{x}_i + \gamma_j \bar{\underset{\sim}{x}}_j \qquad \text{sum on } i,\ j \tag{3.23}$$

Note that the above expansion is valid for either update, however, the range of i and j and the actual vectors will vary depending on whether the subcycle or large time step update are considered.

Substituting the above expression into Eq. (2.9) and taking into account Eq. (3.1), the updated solution can be written as

$$\underset{\sim}{d}^{k+1} = \beta_i\lambda_i \underset{\sim}{x}_i + \gamma_j \bar{\underset{\sim}{x}}_j \qquad \text{sum of } i,\ j \tag{3.24}$$

since $\lambda_j = 1$ for $\bar{\underset{\sim}{x}}_j$. Defining the norm

$$E = \underset{\sim}{d}^T \underset{\sim}{K}\underset{\sim}{d} \tag{3.25}$$

and using the fact that vectors $\underset{\sim}{x}_i$ are orthogonal to each other and to $\bar{\underset{\sim}{x}}_j$ with respect to $\underset{\sim}{K}$, substituting Eq. (3.23) gives

$$\begin{aligned} E^k &= \beta_i\beta_\ell\ \underset{\sim}{x}_i^T\ \underset{\sim}{K}\underset{\sim}{x}_\ell + \gamma_j\gamma_m\ \bar{\underset{\sim}{x}}_j^T\ \underset{\sim}{K}\bar{\underset{\sim}{x}}_m \qquad \text{sum on } i,\ j,\ \ell,\ m \\ &= \beta_i\beta_\ell\mu_i\delta_{ij} + \gamma_j\gamma_m\bar{\underset{\sim}{x}}_j^T\ \underset{\sim}{K}\bar{\underset{\sim}{x}}_m \end{aligned} \tag{3.26a}$$

$$E^k = \beta_i^2 \mu_i + \gamma_j \gamma_m \bar{\underset{\sim}{x}}_j^T \underset{\sim\sim}{K} \bar{\underset{\sim}{x}}_m \tag{3.26b}$$

Similarly, substituting Eq. (3.24) gives

$$E^{k+1} = \beta_i^2 \lambda_i^2 \mu_i + \gamma_j \gamma_m \bar{\underset{\sim}{x}}_j^T \underset{\sim\sim}{K} \bar{\underset{\sim}{x}}_m \quad \text{sum on i, j, m} \tag{3.27}$$

If stability is defined as $|\lambda_i| \leq 1$, as was previously assumed, then it is apparent from the above two equations that $E^{k+1} \leq E^k$. Moreover, since this norm decreases or remains constant for each update, it and thus $\underset{\sim}{d}$, remain bounded for all time.

In determining the critical time step the element eigenvalue theorem will be used which states that

$$\mu_{max} \leq \text{MAX } \mu_{max}^e \qquad \text{for all e} \tag{3.28}$$

where μ_{max} is the eigenvalue of an assemblage of elements subject to arbitrary constraints and μ_{max}^e is the maximum unconstrained eigenvalue for any of the elements [8, 11, 12]. For the subcycle to satisfy the stability condition, the time step must be choosen according to Eq. (3.15) where μ_i is given by Eq. (3.6). Equation (3.6) represents a constrained version of the eigenvalue problem obtained by assembling all the elements which contain a node of partition B so that

$$\mu_{max} \leq \text{MAX } \mu^e \qquad \text{for all e } \epsilon \text{ } S_B \tag{3.29}$$

where S_B is comprised of elements which have a node in partition B. Using the above inequality Eq. (3.15) is satisfied if

$$\Delta t \leq \frac{2}{(1 - 2\alpha_B)\mu_{max}^e} \qquad \text{for all e } \epsilon \text{ } S_B \tag{3.30}$$

Similar arguments can be used for the large time step update to bound μ_i' from Eq. (3.20) as

$$\mu'_{max} \leq \text{MAX } \mu^e_{max} \qquad \text{for all } e \in S_A \tag{3.31}$$

The above inequality can be used in Eq. (3.22) to provide a bound for the critical time step in the form

$$\Delta t \leq \frac{2}{m(1 - 2\alpha_A)\mu^e_{max}} \qquad \text{for all } e \in S_A \tag{3.32}$$

Equations (3.30) and (3.32) give a convenient and easily calculated form for choosing the time steps and time step ratio for the subcycling algorithm.

REFERENCES

(1) T. Belytschko, H.J. Yen and R. Mullen, "Mixed Methods for Time Integration," Comp. Meth. Appl. Mech. Engrg., 17/18, 259-275 (1979).

(2) T.J.R. Hughes and W.K. Liu, "Implicit-Explicit Finite Elements in Transient Analysis: Implementation and Numerical Examples," J. Appl. Mech., 45, 375-378 (1978).

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

(4) K.C. Park, "Partitioned Transient Analysis Procedure for Coupled-Field Problems: Stability Analysis," J. Appl. Mech., 47, 370-376 (1980).

(5) T. Belytschko and R. Mullen, "Stability of Explicit-Implicit Mesh Partitions in Time Integration," Int. J. Numer. Methods Engrg., 12, 1575-1586 (1978).

(6) W.K. Liu and J. Lin, "Stability of Mixed Time Integration Schemes for Transient Thermal Analysis," Numerical Heat Transfer, 5, 211-222 (1982).

(7) T. Belytschko, W.K. Liu and P. Smolinski, "Computational Methods for Analysis of Transient Response," A.S.C.E. Engineering Mechanics Speciality Conf., Purdue University, West Lafayette, Indiana, 23-25, May, 1983.

(8) T. Belytschko, P. Smolinski and W.K. Liu, "Multi-Stepping Implicit-Explicit Procedures in Transient Analysis" in: Innovative Methods for Nonlinear Problems, W.K. Liu, T. Belytschko and K.C. Park, Eds. (Pineridge Press, Swansea, U.K.) 135-153 (1984).

(9) P. Smolinski, T. Belytschko and W.K. Liu, "Stability of Multi-Time Step Partitioned Transient Analysis for First Order Systems of Equations," submitted for publication.

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

(11) B.M. Irons, "Application of a Theorem on Eigenvalues to Finite Element Problems," (CR/132/70) University of Wales, Dept. of Civil Engineering, Swansea, 1970.

(12) D.P. Flanagan and T. Belytschko, "Simultaneous Relaxation in Structural Dynamics," Journal of Engineering Mechanics Division, ASCE, 1039-1055 (1981).

~~ABSTRACT~~

@ NBS A nonlinear structural dynamics program with an element library that exploits parallel processing is under development. The aim is to exploit scheduling-allocation so that parallel processing and vectorization can effectively be treated in a general purpose program. As a byproduct an automatic scheme for assigning time steps was devised. A rudimentary form of the program is complete and has been tested: it shows substantial advantage can be taken of parallelism.

In addition, a stability proof for the subcycling algorithm has been developed.

@ ABA Author.