A GENERAL SOLUTION STRATEGY FOR LARGE SCALE STATIC AND DYNAMIC NONLINEAR FINITE ELEMENT PROBLEMS EMPLOYING THE ELEMENT-BY-ELEMENT FACTORIZATION CONCEPT

Thesis by

Itzhak Levit

In Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy

California Institute of Technology

Pasadena, California

1983

(Submitted September 28, 1982)

To the Memory of Eran

ACKNOWLEDGEMENTS

I wish to extend my deepest gratitude to the following:

to my advisor: Professor T.J.R. Hughes for his assistance and judicious guidance and encouragement throughout this work.

to my colleague: James Winget for his active participation in this research and contribution in the implementation aspects of this work.

to my colleague: Dave Fyhrie for proofreading the manuscript.

to NASA Ames Research Center: for providing their excellent computing services.

to Civil Engineering Laboratory: for providing financial support for this research.

and last but not least, to my family whose unending support and encouragement have brought me this far.

ABSTRACT

It is proposed to solve large-scale finite-element equation systems arising in structural and solid mechanics by way of an element-by-element approximate factorization technique which obviates the need for a global coefficient matrix. The procedure has considerable operation count and I/O advantages over direct elimination schemes and it is easily implemented. Numerical results demonstrate the effectiveness of the method and suggest its potential for the analysis of large-scale systems.

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION

Most problems faced by engineers, in everyday practice, can be solved, in principle, using the finite element method. However, large scale problems either linear or nonlinear, require solution of large equation systems and consequently require considerable amounts of in-core storage and computing time.

Fast core storage is limited, even on the advanced super computers such as the CRAY and STAR and one needs to employ secondary storage devices (disks and tapes) to accommodate the global matrices. This results in more complicated (and thus slower) out of core solvers which in turn causes much more time to be spent on I/O between the fast core and the secondary storage devices than on CPU.

The shortage of fast core storage is most apparent for three dimensional dynamic nonlinear problems. For example consider a rectangular continuum domain with twenty elements along a side having three degrees of freedom per node. This problem requires 3.66×10^7 words of storage and 1.7×10^{10} operations per step per iteration.

For these reasons the research on new finite element techniques is concentrated on the following goals:

i) The reduction of storage requirements.

ii) A reduction in the number of operations required in the solution of the equation systems.

Approximate factorization techniques were first suggested for global operators arising from finite difference schemes. The idea is to replace the global operator by a product of operators each requiring a smaller storage space than the global matrix and which can be factorized more efficiently than the governing global operator.

The ADI technique presented by Douglas and Rachford [D4, D5] approximates the global finite difference operator in two and three dimensional spaces by a product of one dimensional operators each of which is assumed to be effective through part of the time step. Works published by Yanenko and Marchuk [Y1, M3, M4] present an approximation of a global operator arising from finite difference schemes of parabolic and hyperbolic equations by products of operators defined on subdomains of the problem and resulting in unconditionally stable second order accurate (in time step) approximations having a very favourable storage requirement and a relatively small number of operations.

A more recent work in the finite differences area by Beam, Warming and Steger [W4] approximates a global multidimensional operator by a product of one dimensional operators having the storage requirements and operation count of a one dimensional operator and retaining the stability and accuracy of the governing multidimensional operator.

All the methods suggested for finite differences inherit the limitations of being able to handle only regular domains and simple boundary conditions. Approximate factorization algorithms have also been proposed for global operators arising in the finite element method. Trujillo [T2, T3] presented a method in which the global operator is replaced by a product of an upper triangular operator and a lower triangular operator saving the entire factorization stage of the solution but having no storage advantage over the governing operator.

This method was modified by Park [P3] to allow rigid body modes to pass through the approximate operator and was shown to possess some superior numerical properties over the method suggested by Trujillo, but still has no storage advantage over the governing operator.

The element-by-element (EBE) operator split was first proposed by Hughes, Levit and Winget [H11] for finite element schemes in heat conduction problems. The method consists of factorizing the global finite element operator into a product of its natural primitives, namely the element operators. The method was shown to retain the flexibility and versatility of the finite element method and avoids the need for storage and factorization of a global operator.

In this work we present algorithms implementing the element-by-element concept for a general class of linearized equation systems emanating from finite element discretization of continuum mechanics and structural problem. The algorithms are applied to a linearized symmetric positive definite operator which admits element level decomposition and are not otherwise limited to any specific class of problems.

In chapter II a detailed formulation of linear equation systems emanating from finite element formulations of linear and nonlinear problems in elasticity are formulated for both the static and dynamic cases.

In chapter III the dynamic regularization method is employed for solving linear equation systems, an element-by-element approximation algorithm is presented and studied and numerical solution techniques (line searches and quasi-Newton updates) are implemented in conjunction with the EBE algorithms and are shown to improve the numerical characteristics of the suggested schemes.

Chapter IV describes the computer program written for implementing the EBE algorithms and presents numerical examples of dynamic linear and nonlinear

problems solved using the EBE algorithm.

Appendix I includes algorithmic and communication flow charts of the finite element system program written for this work.

CHAPTER II

FORMULATION OF LINEAR EQUATION SYSTEMS IN FINITE ELEMENT ANALYSIS

In this chapter, detailed formulations of linear equation systems in finite element analysis for a variety of problems in structural analysis and continuum mechanics are presented. The emphasis in this chapter is on linear and nonlinear classes of problems in elastostatics and elastodynamics. Although other problems in mechanical engineering, such as plasticity, visco plasticity, heat conduction, fluid mechanics, etc., can be cast into similar formats, they are not treated in this chapter.

2.1. Linear Elastostatics

Let Ω be a bounded region in $\mathbb{R}^{n_{sd}}$, where n_{sd} is the number of space dimensions. Assume Ω has a piecewise smooth boundary Γ. Let $\underset{\sim}{x} = \{x_i\}$, $i = 1, 2, \ldots, n_{sd}$, denote a general point in $\overline{\Omega}$ and let $\underset{\sim}{n} = \{n_i\}$ be the unit outer normal vector to Γ. Further, assume Γ admits the following decomposition:

$$\Gamma = \overline{\Gamma_{g_i} \cup \Gamma_{h_i}} \qquad \text{(no sum)} \quad i = 1, 2, \ldots, n_{sd} \tag{2.1.1}$$

$$\Gamma_{g_i} \cap \Gamma_{h_i} = \phi \qquad \text{(no sum)} \quad i = 1, 2, \ldots, n_{sd} \tag{2.1.2}$$

where Γ_{g_i} and Γ_{h_i} are subsets of Γ, the superposed bar, in (2.1.1), represents set closure and ϕ, in (2.1.2), denotes the empty set.

Further, assume the following functions are given:

$$f_i : \Omega \to \mathbb{R} \qquad \text{(body force)} \tag{2.1.3}$$

$$g_i : \Gamma_{g_i} \to \mathbb{R} \qquad \text{(prescribed displacement)} \tag{2.1.4}$$

$$h_i : \Gamma_{h_i} \to \mathbb{R} \qquad \text{(prescribed traction)} \tag{2.1.5}$$

2.1.1 Strong form of the Boundary Value Problem (B.V.P.)

The mathematical statement of the B.V.P. is as follows:

(S)

Given f_i, g_i, h_i as in (2.1.3) - (2.1.5)

Find:

$$u_i : \overline{\Omega} \to \mathbb{R} \tag{2.1.6}$$

such that:

$$\sigma_{ij,j} + f_i = 0 \quad \text{on} \quad \Omega \tag{2.1.7}$$

$$u_i = g_i \quad \text{on} \quad \Gamma_{g_i} \tag{2.1.8}$$

$$\sigma_{ij}\, n_j = h_i \quad \text{on} \quad \Gamma_{h_i} \tag{2.1.9}$$

where $\underset{\sim}{u}(\underset{\sim}{x}) = \{u_i(\underset{\sim}{x})\}$ is the displacement field vector, $\underset{\sim}{\sigma}(\underset{\sim}{x}) \doteq [\sigma_{ij}(\underset{\sim}{x})]$ is the Cauchy stress tensor, the comma in (2.1.7) denotes spatial differentiation (i.e. $\sigma_{ij,j} = \partial\sigma_{ij}/\partial x_j$) and the summation convention on repeated indices is assumed in force. n_j is the j^{th} component of the unit outward normal vector on Γ_{h_i}. Equation (2.1.7) is the equilibrium equation, (2.1.8) is the prescribed displacement boundary condition, frequently called "g-type" B.C. or "essential" B.C., and equation (2.1.9) is the prescribed

traction boundary condition referred to as "$\mathcal{h}$-type" B.C. or "natural" B.C.

To complete the system of field equations we introduce the strain and stress-strain relations as follows:

$$\varepsilon_{ij} = u_{(i,j)} = (u_{i,j} + u_{j,i})/2 \tag{2.1.10}$$

$$\sigma_{ij} = C_{ijk\ell}\, \varepsilon_{k\ell} \tag{2.1.11}$$

where $\underset{\sim}{\varepsilon} = [\varepsilon_{ij}]$ is the infinitesimal strain tensor defined as the symmetric part of the displacement gradient and $\underset{\sim}{C} = [C_{ijk\ell}]$ is the elastic response tensor. In this work we shall assume that the elastic response tensor $\underset{\sim}{C}$ possess the major symmetry and the two minor symmetries with respect to the indices $ijk\ell$ (e.g. linear elastic solid), namely:

$$C_{ijk\ell} = C_{k\ell ij} = C_{jik\ell} \tag{2.1.12}$$

2.1.2 Weighted Residual Form of the Boundary Value Problem

We define the trial solution spaces $\mathcal{S}_i$ as follows:

$$\mathcal{S}_i = \{u_i \mid u_i = g_i \quad \text{on} \quad \Gamma_{g_i}\} \quad i = 1, 2, \ldots, n_{sd} \tag{2.1.13a}$$

and the variation spaces $\mathcal{V}_i$ as:

$$\mathcal{V}_i = \{w_i \mid w_i = 0 \quad \text{on} \quad \Gamma_{g_i}\} \quad i = 1, 2, \ldots, n_{sd} \tag{2.1.13b}$$

Multiply (2.1.7) by $w_i \in \mathcal{V}_i$, and integrate over the domain Ω. The result, together with (2.1.8) and (2.1.9), gives the following weak form:

$$
(W)\quad \left\{
\begin{array}{l}
\text{Given } f_i,\ g_i \text{ and } h_i \text{ as before} \\
\text{Find } u_i \in \mathcal{S}_i \quad \text{S.T. for every } w_i \in \mathcal{V}_i \\
\qquad a(\underset{\sim}{w},\ \underset{\sim}{u}) = (\underset{\sim}{w},\ \underset{\sim}{f}) + (\underset{\sim}{w},\ \underset{\sim}{h})_\Gamma \\
\qquad a(\underset{\sim}{w},\ \underset{\sim}{u}) = \int_\Omega w_{(i,j)}\ C_{ijk\ell}\ u_{(k,\ell)}\, d\Omega \\
\qquad (\underset{\sim}{w},\ \underset{\sim}{f}) = \int_\Omega w_i f_i\, d\Omega \\
\qquad (\underset{\sim}{w},\ \underset{\sim}{h})_\Gamma = \int_{\Gamma_h} w_i h_i\, d\Gamma = \sum_{i=1}^{n_{sd}} \int_{\Gamma_{h_i}} w_i h_i\, d\Gamma
\end{array}
\right. \qquad (2.1.14)
$$

Remarks :

1. $a(\cdot,\ \cdot)$, $(\cdot,\ \cdot)$ & $(\cdot,\ \cdot)_\Gamma$ are symmetric bilinear forms.
2. If certain smoothness conditions are satisfied then (S) <=> (W).
3. (W) is referred to in mechanics as the "Principle of Virtual Work" and w_i as "Virtual Displacement".

2.1.3 Lubnov-Galerkin Approximation of the B.V.P.

Let $\mathcal{V}_i^h \subset \mathcal{V}_i$, i = 1 , 2 ,..., n_{sd} , be subspaces spanned by a finite number of given linearly independent functions, N_A , A = 1 , 2 ,..., n , such that:

$$
\mathcal{V}_i^h = \{ v_i^h \mid v_i^h = \sum_{A=1}^{n} N_A(\underset{\sim}{x}) d_{Ai} \ ; \quad v_i^h = 0 \quad \text{on } \Gamma_{g_i} \}
$$

$$
i = 1 , 2 ,..., n_{sd} \qquad (2.1.15)
$$

where the d_{Ai}'s are nodal quantities.

Let $g_i^h \in \mathcal{S}_i$ (i.e. $g_i^h = g_i$ on Γ_{g_i}) and define the approximate solution

spaces $\mathcal{S}_i^h$ through:

$$\mathcal{S}_i^h = \{u_i^h \mid u_i^h = v_i^h + g_i^h\} \subset \mathcal{S}_i \quad i = 1, 2, \ldots, n_{sd} \tag{2.1.16}$$

The Bubnov-Galerkin analog of (2.1.14) is:

$$(G)\begin{cases} \text{Given } \underset{\sim}{f},\ \underset{\sim}{g} \text{ and } \underset{\sim}{h} \text{ as before} \\ \text{Find } u_i^h = v_i^h + g_i^h \in \mathcal{S}_i^h \text{ S.T. for every } w_i^h \in \mathcal{V}_i^h \\ a(\underset{\sim}{w}^h, \underset{\sim}{v}^h) = (\underset{\sim}{w}^h, \underset{\sim}{f}) + (\underset{\sim}{w}^h, \underset{\sim}{h})_\Gamma - a(\underset{\sim}{w}^h, \underset{\sim}{g}^h) \end{cases} \tag{2.1.17}$$

Note that in (2.1.17) both $\underset{\sim}{w}^h$ and $\underset{\sim}{v}^h$ are vector functions which can be represented as linear combinations of the same finite set of functions, N_A, $A = 1, 2, \ldots, n$.

2.1.4 Vectorial Notation

To simplify future writing, we shall introduce the strain "vector":

$$\underset{\sim}{\varepsilon}(\underset{\sim}{u}) = \{\varepsilon_I(\underset{\sim}{u})\} = \left[\begin{array}{l} \left\{\begin{array}{c} u_{1,1} \\ u_{2,2} \\ u_{1,2} + u_{2,1} \end{array}\right\} \text{ in } 2\text{-D } (n_{sd} = 2) \\ \left\{\begin{array}{c} u_{1,1} \\ u_{2,2} \\ u_{3,3} \\ u_{2,3} + u_{3,2} \\ u_{3,1} + u_{1,3} \\ u_{1,2} + u_{2,1} \end{array}\right\} \text{ in } 3\text{-D } (n_{sd} = 3) \end{array}\right. \tag{2.1.18}$$

where $u_{i,j} = \partial u_i / \partial x_j$ (i , j = 1 , 2 , 3) and similarly the stress "vector":

$$\underset{\sim}{\sigma} = \{\sigma_I\} = \begin{cases} \begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{Bmatrix} & \text{in 2 - D } (n_{sd} = 2) \\ \begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{23} \\ \sigma_{13} \\ \sigma_{12} \end{Bmatrix} & \text{in 3 - D } (n_{sd} = 3) \end{cases} \tag{2.1.19}$$

Next we shall introduce the matrix form of the fourth order elastic response tensor $\underset{\sim}{C}$:

$$\underset{\sim}{D} = [D_{IJ}] = \begin{cases} \begin{bmatrix} D_{11} & D_{12} & D_{13} \\ & D_{22} & D_{23} \\ \text{symm.} & & D_{33} \end{bmatrix} & \text{in 2 - D } (n_{sd} = 2) \\ \begin{bmatrix} D_{11} & D_{12} & D_{13} & D_{14} & D_{15} & D_{16} \\ & D_{22} & D_{23} & D_{24} & D_{25} & D_{26} \\ & & D_{33} & D_{34} & D_{35} & D_{36} \\ & & & D_{44} & D_{45} & D_{46} \\ & & & & D_{55} & D_{56} \\ \text{symm.} & & & & & D_{66} \end{bmatrix} & \text{in 3 - D } (n_{sd} = 3) \end{cases} \tag{2.1.20}$$

where:

$$D_{IJ} = C_{ijk\ell}$$

The relationship between the (I, J) and (i, j, k, ℓ) indices is given in the following tables:

2 - D ($n_{sd} = 2$)

I / J	i / k	j / ℓ
1	1	1
3	1	2
3	2	1
2	2	2

3 - D ($n_{sd} = 3$)

I / J	i / k	j / ℓ
1	1	1
6	1	2
5	1	3
6	2	1
2	2	2
4	2	3
5	3	1
4	3	2
3	3	3

For the 3 - D isotropic case the elastic response tensor, $C_{ijk\ell}$, takes the following form:

$$C_{ijk\ell} = \mu(\delta_{ik}\,\delta_{j\ell} + \delta_{i\ell}\,\delta_{jk}) + \lambda\,\delta_{ij}\,\delta_{k\ell}$$

where μ and λ are the Lamé moduli and δ_{ij} is the Kronecker delta. For this case the $\underset{\sim}{D}$ matrix is given by:

$$\underset{\sim}{D} = \begin{bmatrix} \lambda+2\mu & \lambda & \lambda & 0 & 0 & 0 \\ & \lambda+2\mu & \lambda & 0 & 0 & 0 \\ & & \lambda+2\mu & 0 & 0 & 0 \\ & & & \mu & 0 & 0 \\ & & & & \mu & 0 \\ & \text{symm.} & & & & \mu \end{bmatrix} \qquad (2.1.22)$$

Finally, we shall introduce the $\underset{\sim}{B}_A(\underset{\sim}{x})$ matrix associated with a given function $N_A(\underset{\sim}{x})$:

$$\underset{\sim}{B}_A = \begin{cases} \begin{bmatrix} N_{A,1} & 0 \\ 0 & N_{A,2} \\ N_{A,2} & N_{A,1} \end{bmatrix} & \text{in } 2-D \ (n_{sd}=2) \\ \begin{bmatrix} N_{A,1} & 0 & 0 \\ 0 & N_{A,2} & 0 \\ 0 & 0 & N_{A,3} \\ 0 & N_{A,3} & N_{A,2} \\ N_{A,3} & 0 & N_{A,1} \\ N_{A,2} & N_{A,1} & 0 \end{bmatrix} & \text{in } 3-D \ (n_{sd}=3) \end{cases} \tag{2.1.23}$$

where $N_{A,i} = \partial N_A(\underset{\sim}{x})/\partial x_i$.

If N_A is an interpolation function, the $\underset{\sim}{B}_A$ matrix represents the relationship between the strain "vector" $\underset{\sim}{\varepsilon}$ at a point $\underset{\sim}{x}_A \in \Omega$ and the approximate displacement field $\underset{\sim}{u}^h$ at that point.

2.1.5. Matrix Finite-Element Form of the B.V.P.

Consider a discretization of Ω into bounded element subdomains Ω^e, $e = 1, 2, \ldots, n_{el}$, where n_{el} is the number of elements. Let Γ^e denote the boundary of Ω^e and assume:

$$\bigcup_e \overline{\Omega}^e = \overline{\Omega} \tag{2.1.24}$$

$$\bigcap_e \Omega^e = \emptyset \tag{2.1.25}$$

Further, assume that $\overline{\Omega}$ and Γ_{g_i} contain η and η_{g_i} nodal points respectively ($\eta_{g_i} \subset \eta$). (η and η_{g_i} are sets.)

Consider interpolation functions which satisfy:

$$N_A(\underset{\sim}{x}) = \begin{cases} 1 & \underset{\sim}{x} = \underset{\sim}{x}_A \\ 0 & \underset{\sim}{x} = \underset{\sim}{x}_B \end{cases} \qquad \begin{matrix} A, B \in \eta \\ A \neq B \end{matrix} \tag{2.1.26}$$

where $\underset{\sim}{x}_A$ and $\underset{\sim}{x}_B$ are position vectors of nodes A and B respectively. Fig. 1 illustrates a 2 - D example of $N_A(\underset{\sim}{x})$. Clearly the N_A, A = 1, 2, ..., n, functions constitute a basis for the Galerkin spaces $\mathcal{S}_i^h$ and $\mathcal{V}_i^h$.

Thus we can write:

$$\begin{cases} v_i^h(\underset{\sim}{x}) = \sum\limits_{A \in \eta - \eta_{g_i}} N_A(\underset{\sim}{x}) d_{iA} \in \mathcal{V}_i^h & i = 1, 2, \ldots, n_{sd} \\ g_i^h(\underset{\sim}{x}) \cong \sum\limits_{A \in \eta_{g_i}} N_A(\underset{\sim}{x}) g_{iA} \in \mathcal{S}_i^h & i = 1, 2, \ldots, n_{sd} \end{cases} \tag{2.1.27}$$

where d_{iA} and g_{iA} are constant nodal values.*

Define the following vector quantities:

$$\begin{cases} \underset{\sim}{v}^h = v_i^h \underset{\sim}{e}_i \\ \underset{\sim}{g}^h = g_i^h \underset{\sim}{e}_i \\ \underset{\sim}{w}^h = w_i^h \underset{\sim}{e}_i \end{cases} \tag{2.1.28}$$

where $\underset{\sim}{e}_i$'s are unit vectors in the i^{th} space dimension direction:

$$\begin{cases} \underset{\sim}{e}_i = \{e_{ij}\} \\ e_{ij} = \delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \end{cases} \qquad i, j = 1, 2, \ldots, n_{sd} \tag{2.1.29}$$

---

* It is assumed that the g-boundary data are approximated by nodal interpolation via the N_A's.

Substituting (2.1.28) into the Galerkin approximation (2.1.17) and using the vectorial definitions of the previous section one obtains the following system of linear equations:

$$\boxed{\underset{\sim}{K}\,\underset{\sim}{d} = \underset{\sim}{F}} \tag{2.1.30}$$

where $\underset{\sim}{K}$ is the stiffness matrix, $\underset{\sim}{d}$ vector of nodal displacement and $\underset{\sim}{F}$ is the load vector.

$\underset{\sim}{K}$ and $\underset{\sim}{F}$ are assembled from the elements contributions via:

$$\begin{aligned} \underset{\sim}{K} &= \mathop{\mathbf{A}}_{e=1}^{n_{e\ell}} \underset{\sim}{k}^e \\ \underset{\sim}{F} &= \mathop{\mathbf{A}}_{e=1}^{n_{e\ell}} \underset{\sim}{f}^e \end{aligned} \tag{2.1.31}$$

where $\mathbf{A}$ is the finite element assembly operator*, $\underset{\sim}{k}^e$ and $\underset{\sim}{f}^e$ are the e^{th} element stiffness matrix and load vector respectively given by:

$$\begin{cases} \underset{\sim}{k}^e = [k^e_{pq}] \qquad 1 \le p\,,\, q \le n_{ee} \\ k^e_{pq} = \underset{\sim i}{e}^T \displaystyle\int_{\Omega^e} \underset{\sim a}{B}^T \underset{\sim}{D} \underset{\sim b}{B}\, d\Omega\, \underset{\sim j}{e} \end{cases} \tag{2.1.32}$$

$$\begin{cases} \underset{\sim}{f}^e = \{f^e_p\} \\ f^e_p = \displaystyle\int_{\Omega^e} N_a\, f_i\, d\Omega + \int_{\Gamma^e \cap \Gamma_{h_i}} N_a\, h_i\, d\Gamma - \sum_{q=1}^{n_{ee}} k^e_{pq}\, g^e_q \end{cases} \tag{2.1.33}$$

where n_{ee} is the number of element equations ($n_{ee} = n_{en} \times n_{ed}$: number

*The finite element assembly operator maps the element local degrees of freedom to the global degrees of freedom and adds the element contribution to the global array.

of element nodes times number of element degrees of freedom per node). a , b are local element node numbers and the relationship between the element equation numbers, (p,q), and node numbers, (a,b), is given by:

$$\begin{cases} p = n_{ed}(a - 1) + i & 1 \leq a , b \leq n_{en} \\ q = n_{ed}(b - 1) + j & 1 \leq i , j \leq n_{ed} \end{cases} \tag{2.1.34}$$

Finally, g_q^e's are defined through:

$$g_q^e = g_{jb}^e = \begin{cases} g_j(\underset{\sim}{x}_b^e) & \text{if } \underset{\sim}{x}_b^e \in \eta_{g_j} \\ 0 & \text{otherwise} \end{cases} \tag{2.1.35}$$

2.2 Linear Elastodynamics

Let Ω , Γ , Γ_{g_i} and Γ_{h_i} be as in section 2.1, and assume the following functions are given:

$$\rho : \Omega \to \mathbb{R} \quad \text{(mass density)} \tag{2.2.1}$$

$$f_i : \Omega \times]0, T[\to \mathbb{R} \tag{2.2.2}$$

$$g_i : \Gamma_{g_i} \times]0, T[\to \mathbb{R} \tag{2.2.3}$$

$$h_i : \Gamma_{h_i} \times]0, T[\to \mathbb{R} \tag{2.2.4}$$

$$u_{oi} : \Omega \to \mathbb{R} \quad \text{(initial displacement)} \tag{2.2.5}$$

$$\dot{u}_{oi} : \Omega \to \mathbb{R} \quad \text{(initial velocity)} \tag{2.2.6}$$

2.2.1 Strong Form of the Initial Boundary Value Problem (I.B.V.P.)

The mathematical statement of the I.B.V.P. is as follows:

(S)

Given ρ, f_i, g_i, h_i, u_{oi} and $\dot{u}_{oi}$ as in (2.2.1) - (2.2.6)

Find:

$$u_i : \overline{\Omega} \times [0, T] \to \mathbb{R} \tag{2.2.7}$$

such that:

$$\rho \ddot{u}_i = \sigma_{ij,j} + f_i \quad \text{on} \quad \Omega \times]0, T[\tag{2.2.8}$$

$$u_i = g_i \quad \text{on} \quad \Gamma_{g_i} \times]0, T[\tag{2.2.9}$$

$$\sigma_{ij} n_j = h_i \quad \text{on} \quad \Gamma_{h_i} \times]0, T[\tag{2.2.10}$$

$$\left.\begin{aligned} u_i(\underset{\sim}{x}, 0) &= u_{oi}(\underset{\sim}{x}) \\ \dot{u}_i(\underset{\sim}{x}, 0) &= \dot{u}_{oi}(\underset{\sim}{x}) \end{aligned}\right\} \quad \text{on} \quad \Omega \tag{2.2.11}$$

where the superposed dot denotes time differentiation in the inertial reference frame (i.e. $\dot{u}_i = \partial u_i / \partial t$) and all the definitions and conventions of section 2.1.1 are enforced.

2.2.2 Weighted Residual Form of the I.B.V.P.

Let $\mathcal{S}_i$ and $\mathcal{V}_i$ be the trial solution spaces* and variation spaces respectively defined through equations (2.1.13a) and (2.1.13b). Multiplying (2.2.8) and (2.2.11) by $w_i \in \mathcal{V}_i$, integrating over Ω and use of the boundary conditions (2.2.9) and (2.2.10) we obtain the following weak form of the I.B.V.P.:

*The trial solution spaces are now time dependent due to (2.2.9).

(W)
$$\left\{\begin{array}{l}
\text{Given } \rho\,,\ f_i\,,\ g_i\,,\ h_i\,,\ u_{oi} \text{ and } \dot{u}_{oi} \text{ as in (2.2.1) - (2.2.6)} \\
\text{Find } u_i : [0,\,T] \to \mathcal{S}_i \text{ S.T. for every } w_i \in \mathcal{V}_i \\
(\underset{\sim}{w},\ \rho\,\ddot{\underset{\sim}{u}}) + a(\underset{\sim}{w},\ \underset{\sim}{u}) = (\underset{\sim}{w},\ \underset{\sim}{f}) + (\underset{\sim}{w},\ \underset{\sim}{h})_\Gamma \qquad (2.2.12) \\
(\underset{\sim}{w},\ \underset{\sim}{u}\,(\underset{\sim}{x},\ 0) - \underset{\sim}{u}_o) = 0 \qquad (2.2.13) \\
(\underset{\sim}{w},\ \dot{\underset{\sim}{u}}(\underset{\sim}{x},\ 0) - \dot{\underset{\sim}{u}}_o) = 0 \qquad (2.2.14)
\end{array}\right.$$

where $a(\cdot,\ \cdot)$, $(\cdot,\ \cdot)$ and $(\cdot,\ \cdot)_\Gamma$ are the same symmetric bilinear forms defined in (2.1.14)

2.2.3 Bubnov Galerkin Approximation of the I.B.V.P.

Let $\mathcal{V}_i^h$ and $\mathcal{S}_i^h$ be the subspaces defined through (2.1.15) and (2.1.16) and use $\mathcal{V}_i^h$ and $\mathcal{S}_i^h$ as approximations for $\mathcal{V}_i$ and $\mathcal{S}_i$ in the weighted residual form to obtain:

(G)
$$\left\{\begin{array}{l}
\text{Given } \rho\,,\ f_i\,,\ g_i\,,\ h_i\,,\ u_{oi} \text{ and } \dot{u}_{oi} \text{ as in (2.2.1) - (2.2.6)} \\
\text{Find:} \\
\quad u_i^h = v_i^h + g_i^h : [0,\,T] \to \mathcal{S}_i^h \text{ S.T. for every } w_i \in \mathcal{V}_i^h \\
(\underset{\sim}{w}^h,\ \rho\,\ddot{\underset{\sim}{v}}^h) + a(\underset{\sim}{w}^h,\ \underset{\sim}{v}^h) = (\underset{\sim}{w}^h,\ \underset{\sim}{f}) + (\underset{\sim}{w}^h,\ \underset{\sim}{h})_\Gamma - \\
\qquad - (\underset{\sim}{w}^h,\ \rho\,\ddot{\underset{\sim}{g}}^h) - a(\underset{\sim}{w}^h,\ \underset{\sim}{g}^h) \qquad (2.2.15) \\
(\underset{\sim}{w}^h,\ \underset{\sim}{v}^h(\underset{\sim}{x},\ 0)) = (\underset{\sim}{w}^h,\ \underset{\sim}{u}_o - \underset{\sim}{g}^h(\underset{\sim}{x},\ 0)) \qquad (2.2.16) \\
(\underset{\sim}{w}^h,\ \dot{\underset{\sim}{v}}^h(\underset{\sim}{x},\ 0)) = (\underset{\sim}{w}^h,\ \dot{\underset{\sim}{u}}_o - \dot{\underset{\sim}{g}}^h(\underset{\sim}{x},\ 0)) \qquad (2.2.17)
\end{array}\right.$$

2.2.4 Semidiscretized Equation of Motion

We proceed as in section 2.1.5 by discretizing Ω into bounded subdomains Ω^e, $e = 1,\ 2,\ \ldots,\ n_{e\ell}$. Let η and η_{g_i} be the same sets as in 2.1.5 and

$N_A(\underset{\sim}{x})$'s be the interpolation functions defined by equation (2.1.26) constructing the basis of the Bubnov-Galerkin spaces $\mathcal{V}_i^h$ and $\mathcal{S}_i^h$.

Thus:

$$\begin{cases} w_i^h(\underset{\sim}{x},\ t) = \displaystyle\sum_{A \in \eta - \eta_{g_i}} N_A(\underset{\sim}{x})\ d_{iA}(t) : [0,\ T] \to \mathcal{V}_i^h \\ g_i^h(\underset{\sim}{x},\ t) = \displaystyle\sum_{A \in \eta_{g_i}} N_A(\underset{\sim}{x})\ g_{iA}(t) : [0,\ T] \to \mathcal{S}_i^h \end{cases} \tag{2.2.18}$$

where $d_{iA}(t)$ and $g_{iA}(t)$ are nodal values at time t.

Define the vector valued functions $\underset{\sim}{w}^h$, $\underset{\sim}{g}^h$ and $\underset{\sim}{w}^h$ through equations (2.1.28) and substitute the above into the Bubnov-Galerkin form of the I.B.V.P. (equations (2.2.15) - (2.2.17)) to obtain the following semidiscretized form:

Given $\underset{\sim}{M}$, $\underset{\sim}{K}$ and $\underset{\sim}{F}(t)$

Find:

$$\underset{\sim}{d} : [0,\ T] \to \mathbb{R}^{n_{sd}}$$

such that:

$$\underset{\sim}{M}\,\ddot{\underset{\sim}{d}}(t) + \underset{\sim}{K}\,\underset{\sim}{d}(t) = \underset{\sim}{F}(t) \tag{2.2.19}$$

$$\underset{\sim}{d}(0) = \underset{\sim}{d}^o\ ;\quad \dot{\underset{\sim}{d}}(0) = \dot{\underset{\sim}{d}}^o \tag{2.2.20}$$

where $\underset{\sim}{M}$ is the mass matrix, $\underset{\sim}{K}$ is the stiffness matrix, $\underset{\sim}{F}(t)$ is the load vector at time t and $\underset{\sim}{d}(t)$ is the vector of nodal displacement at time t. $\underset{\sim}{d}^o$ and $\dot{\underset{\sim}{d}}^o$ emanate from consistent approximations of the initial conditions via:

$$u_{oi}(\underset{\sim}{x}) \cong \sum_{A \in \eta} N_A(\underset{\sim}{x}) d_{iA}^o \tag{2.2.21}$$

$$\dot{u}_{oi}(\underset{\sim}{x}) \cong \sum_{A \in \eta} N_A(\underset{\sim}{x})\ \dot{d}_{iA}^o \tag{2.2.22}$$

The stiffness matrix, $\underset{\sim}{K}$, is defined similarly to the static case and its element components are given by equation (2.1.32).

The mass matrix, $\underset{\sim}{M}$, and the vector of applied forces, $\underset{\sim}{F}(t)$, are defined as follows:

$$\underset{\sim}{M} = \mathop{\mathbf{A}}_{e=1}^{n_{e\ell}} \underset{\sim}{m}^e \quad ; \quad \underset{\sim}{m}^e = [m^e_{pq}]$$

$$m^e_{pq} = \delta_{ij} \int_{\Omega^e} \rho(\underset{\sim}{x})\, N_a(\underset{\sim}{x})\, N_b(\underset{\sim}{x})\, d\Omega \qquad p, q = 1, 2, \ldots, n_{ee} \tag{2.2.23}$$

$$\underset{\sim}{F}(t) = \mathop{\mathbf{A}}_{e=1}^{n_{e\ell}} \underset{\sim}{f}^e(t) \quad ; \quad \underset{\sim}{f}^e_{(t)} = \{f^e_p(t)\}$$

$$f^e_p(t) = \int_{\Omega^e} N_a(\underset{\sim}{x})\, f_i(\underset{\sim}{x}, t)\, d\Omega + \int_{\Gamma^e \cap \Gamma_{h_i}} N_a(\underset{\sim}{x})\, h_i(\underset{\sim}{x}, t)\, d\Gamma$$

$$- \sum_{q=1}^{n_{ee}} [k^e_{pq} g^e_q(t) + m^e_{pq} \ddot{g}^e_q(t)] \qquad p = 1, 2, \ldots, n_{ee} \tag{2.2.24}$$

where $\mathbf{A}$ is the finite element assembly operator. The relationship between the local node numbers, (a,b), and the element equation numbers, (p,q), is given by equation (2.1.34).

The semidiscretized g-type data are defined by:

$$g^e_q(t) = g^e_{jb}(t) = \begin{cases} g_j(\underset{\sim}{x}^e_b, t) & \text{for } \underset{\sim}{x}^e_b \in \eta_{g_i} \\ 0 & \text{otherwise} \end{cases} \tag{2.2.25}$$

and similarly

$$\ddot{g}^e_q(t) = \ddot{g}^e_{jb}(t) = \begin{cases} \ddot{g}_j(\underset{\sim}{x}^e_b, t) & \text{for } \underset{\sim}{x}^e_b \in \eta_{g_i} \\ 0 & \text{otherwise} \end{cases} \tag{2.2.26}$$

We shall introduce the damping matrix $\underset{\sim}{C}$ which represents the observed damping properties of structures. $\underset{\sim}{C}$ does not emanate from classical elasticity theory and usually is written in the so called Rayleigh Damping [Z1] form:

$$\underset{\sim}{C} = a\,\underset{\sim}{M} + b\,\underset{\sim}{K} \tag{2.2.27}$$

where a and b are constants referred to as the external and internal damping coefficients respectively. Including the damping matrix, the semi-discretized I.B.V.P. can be stated as follows:

$$\begin{cases} \text{Given } \underset{\sim}{M},\ \underset{\sim}{C},\ \underset{\sim}{K} \text{ and } \underset{\sim}{F}(t) \text{ as before} \\ \text{Find: } \underset{\sim}{d} : [0, T] \rightarrow \mathbb{R}^{n_{sd}} \\ \text{such that:} \\ \qquad \underset{\sim}{M}\,\ddot{\underset{\sim}{d}}(t) + \underset{\sim}{C}\,\dot{\underset{\sim}{d}}(t) + \underset{\sim}{K}\,\underset{\sim}{d}(t) = \underset{\sim}{F}(t) \qquad (2.2.28) \\ \qquad \underset{\sim}{d}(0) = \underset{\sim}{d}^{o}\ ; \quad \dot{\underset{\sim}{d}}(0) = \dot{\underset{\sim}{d}}^{o} \qquad (2.2.29) \end{cases}$$

Equation (2.2.28) represents a system of ordinary differential equations and (2.2.29) the corresponding initial conditions. From this point on we shall assume that $\underset{\sim}{M}$ is a symmetric positive definite matrix and that $\underset{\sim}{C}$ and $\underset{\sim}{K}$ are symmetric positive semi-definite matrices.

To reduce (2.2.28) to a system of linear algebraic equations we need to discretize the time domain of the problem.

2.2.5 Temporal Discretization

The subject of temporal or time domain discretization for second order systems has been intensively explored since the appearance of the first digital computers. Consequently, a variety of remarkable works have been published

on this subject and many algorithms suggested.

The existing techniques can be classified into two main categories:

1. Single step methods
2. Linear | multi-step methods (L.M.S.)

The first includes all methods for which we can advance the solution to the next time step using only the known state-vector of the previous step and the latter requires state-vectors of more than one previous step.

Well known members of the first category are the Newmark β-method [N1], the Wilson θ-method [W1], the collocation method [H1] and the Hilber, Hughes and Taylor α-method [H2]. As members of the second category we shall mention the Park method [P1] and the Houbolt method [H3].

In this work, for reasons that will become clear in the subsequent sections, the Newmark β-method is used.

2.2.6 The Newmark β-Method

The Newmark family of algorithms is given by the following equations:

$$\underset{\sim}{M}\,\underset{\sim}{a}_{n+1} + \underset{\sim}{C}\,\underset{\sim}{V}_{n+1} + \underset{\sim}{K}\,\underset{\sim}{d}_{n+1} = \underset{\sim}{F}_{n+1} \tag{2.2.30}$$

$$\underset{\sim}{d}_{n+1} = \underset{\sim}{d}_{n} + \Delta t\,\underset{\sim}{V}_{n} + \frac{\Delta t^2}{2}(1-2\beta)\underset{\sim}{a}_{n} + \Delta t^2\beta\,\underset{\sim}{a}_{n+1} \tag{2.2.31}$$

$$\underset{\sim}{V}_{n+1} = \underset{\sim}{V}_{n} + \Delta t(1-\gamma)\underset{\sim}{a}_{n} + \Delta t\,\gamma\,\underset{\sim}{a}_{n+1} \tag{2.2.32}$$

$$\left.\begin{aligned} &\underset{\sim}{d}_{o} = \underset{\sim}{d}^{o} \quad ; \quad \underset{\sim}{V}_{o} = \dot{\underset{\sim}{d}}^{o} \\ &\underset{\sim}{a}_{o} = \underset{\sim}{M}^{-1}\{\underset{\sim}{F}_{o} - \underset{\sim}{C}\,\underset{\sim}{V}_{o} - \underset{\sim}{K}\,\underset{\sim}{d}_{o}\} \end{aligned}\right\} \tag{2.2.33}$$

where $\underset{\sim}{d}_{n+1}$ is the approximation of the spatially discretized displacement vector $\underset{\sim}{d}(t)$ at $t_{n+1} = (n+1)\Delta t$, Δt is a time step, $\underset{\sim}{F}_{n+1}$ is the load

vector $\underset{\sim}{F}(t)$ at t_{n+1}, $\underset{\sim}{V}_{n+1}$ and $\underset{\sim}{a}_{n+1}$ are the approximations of the first and second time derivatives of $\underset{\sim}{d}(t)$, respectively, at t_{n+1} (velocity and acceleration) and β, γ are parameters which control the stability and accuracy of the method.

To study the characteristics of the Newmark family we must reduce (2.2.30) to single degree of freedom (S.D.O.F.) equations via modal decomposition and write them in first order form.

2.2.7 Modal Decomposition and First Order Form of the Equation of Motion

Recall the undamped eigenproblem:

$$[\underset{\sim}{K} - \lambda \underset{\sim}{M}]\underset{\sim}{\Psi} = \underset{\sim}{0} \tag{2.2.34}$$

where λ is an eigenvalue and $\underset{\sim}{\Psi}$ the corresponding eigenvector. The solution of (2.2.34) leads to n_{eq} pairs of eigenvalues and eigenvectors $\{\lambda_\ell, \underset{\sim}{\Psi}_\ell\}_{\ell=1}^{n_{eq}}$. The eigenvectors satisfy the following orthogonality conditions:

$$\underset{\sim}{\Psi}_\ell^T \underset{\sim}{M} \underset{\sim}{\Psi}_m = \delta_{\ell m} \tag{2.2.35}$$

$$\underset{\sim}{\Psi}_\ell^T \underset{\sim}{K} \underset{\sim}{\Psi}_m = \lambda_\ell \delta_{\ell m} \quad \text{(no sum)} \tag{2.2.36}$$

Further, if we assume Rayleigh type damping is present then:

$$\underset{\sim}{C} = a \underset{\sim}{M} + b \underset{\sim}{K} \tag{2.2.37}$$

$$\underset{\sim}{\Psi}_\ell^T \underset{\sim}{C} \underset{\sim}{\Psi}_m = (a + b \lambda_\ell)\delta_{\ell m} \quad \text{(no sum)} \tag{2.2.38}$$

We define the ℓ^{th} modal damping factor ξ_ℓ as:

$$\xi_\ell = (a/\omega_\ell + b \omega_\ell)/2 \tag{2.2.38}$$

Applying modal decomposition to the Newmark equations leads to the following S.D.O.F. form of the algorithm (drop ℓ subscripts):

$$\begin{cases} a_{n+1} + 2\xi\,\omega\,V_{n+1} + \omega^2\,d_{n+1} = F_{n+1} & (2.2.39) \\ d_{n+1} = d_n + \Delta t\,V_n + \frac{\Delta t^2}{2}(1-2\beta)a_n + \Delta t^2\beta\,a_{n+1} & (2.2.40) \\ V_{n+1} = V_n + \Delta t(1-\gamma)a_n + \Delta t\,\gamma\,a_{n+1} & (2.2.41) \\ d_o\,,\ V_o \text{ are given} & \\ a_o = F_o - 2\xi\,\omega\,V_o - \omega^2\,d_o & (2.2.42) \end{cases}$$

where d_n, V_n and a_n are the ℓ^{th} components of the generalized displacement, velocity and acceleration vectors, respectively, at time t_n and F_n is the ℓ^{th} component of the generalized load vector at t_n.

Equations (2.2.39) - (2.2.41) can be written in the following first order form:

$$\underset{\sim}{Y}_{n+1} = \underset{\sim}{A}\,\underset{\sim}{Y}_n + \underset{\sim}{L}_n \qquad (2.2.43)$$

$$\underset{\sim}{Y}_n = \begin{Bmatrix} d_n \\ V_n \end{Bmatrix} \qquad (2.2.44)$$

$$\underset{\sim}{A} = \underset{\sim}{A}_1^{-1}\,\underset{\sim}{A}_2 \qquad (2.2.45)$$

$$\underset{\sim}{A}_1 = \begin{bmatrix} (1 + \Delta t^2\beta\,\omega^2) & (2\,\Delta t^2\beta\,\xi\,\omega) \\ (\Delta t\,\gamma\,\omega^2) & (1 + 2\,\Delta t\gamma\xi\,\omega) \end{bmatrix} \qquad (2.2.46)$$

$$A_2 = \begin{bmatrix} [1 - \frac{\Delta t^2}{2}(1-2\beta)\omega^2] & \Delta t[1 - \Delta t(1-2\beta)\xi\,\omega] \\ [-\Delta t(1-\gamma)\omega^2] & [1 - 2\,\Delta t(1-\gamma)\xi\,\omega] \end{bmatrix} \qquad (2.2.47)$$

$$\underset{\sim}{L}_n = \underset{\sim}{A}_1^{-1} \begin{Bmatrix} \frac{\Delta t^2}{2}[(1 - 2\beta)F_n + 2\beta F_{n+1}] \\ \Delta t[(1 - \gamma)F_n + \gamma F_{n+1}] \end{Bmatrix} \tag{2.2.48}$$

where $\underset{\sim}{A}$ is the algorithmic amplification matrix and $\underset{\sim}{L}_n$ is the load vector.

2.2.8 Accuracy Analysis of the Newmark Family of Algorithms

To analyze the accuracy of an algorithm some accuracy measure is needed. We define the "local truncation error vector", $\underset{\sim}{\tau}(t_n)$, as follows

$$\underset{\sim}{\tau}(t_n) = \underset{\sim}{Y}(t_{n+1}) - \underset{\sim}{A}\,\underset{\sim}{Y}(t_n) - \underset{\sim}{L}(t_n) \tag{2.2.49}$$

$$\underset{\sim}{Y}(t_n) = \begin{Bmatrix} d(t_n) \\ V(t_n) \end{Bmatrix}$$

$\underset{\sim}{\tau}(t_n)$ measures how the exact solution of the S.D.O.F. equation:

$$\ddot{d}(t) + 2\xi\,\omega\,\dot{d}(t) + \omega^2\,d(t) = F(t) \tag{2.2.50}$$

fits the algorithmic approximation given by equation (2.2.43). If we further assume that the exact solution of (2.2.50) satisfies (2.2.43) at t_n, then $\underset{\sim}{\tau}(t_n)$ measures the local error accumulated by the algorithm between step n and step n+1.

Expanding $\underset{\sim}{Y}(t_{n+1})$ about $\underset{\sim}{Y}(t_n)$ using a Taylor series with remainder and use of equation (2.2.50) and time derivatives of (2.2.50) together with (2.2.49) leads to:

$$|\underset{\sim}{\tau}(t_n)| = C\,\Delta t^{k+1} \tag{2.2.51}$$

where C is some constant and k is the order of accuracy of the algorithm.

One obtains the following results for the Newmark family:

$$\begin{aligned} &k = 2 \quad \text{for} \quad \gamma = \tfrac{1}{2} \\ &k = 1 \quad \text{for} \quad \gamma \neq \tfrac{1}{2} \quad , \quad \gamma \in [0, 1] \end{aligned} \tag{2.2.52}$$

So the Newmark algorithms are second order accurate for $\gamma = \frac{1}{2}$ and first order accurate otherwise.

2.2.9 Stability Analysis of the Newmark Family of Algorithms

For stability analysis we consider the homogeneous version of equation (2.2.43) (i.e. $\underset{\sim}{L}_n = \underset{\sim}{0}$) .

The stability criterion is:

$$\rho(\underset{\sim}{A}) \leq 1 \tag{2.2.53}$$

where $\rho(\underset{\sim}{A})$ is the spectral radius of the amplification matrix $\underset{\sim}{A}$ defined through:

$$\rho(\underset{\sim}{A}) = \max_{i=1,\ldots,m} |\lambda_i(\underset{\sim}{A})| \tag{2.2.54}$$

where $\lambda_i(\underset{\sim}{A})$ is the i^{th} eigenvalue of $\underset{\sim}{A}$ and m is the dimension of $\underset{\sim}{A}$. For the Newmark family, $m = 2$, and for the case $\lambda_1(\underset{\sim}{A}) = \lambda_2(\underset{\sim}{A})$ we require that $\rho(\underset{\sim}{A})$ be strictly less than 1. Hughes [H4] obtains the following stability conditions for the Newmark algorithms:

$$\begin{aligned} &2\beta \geq \gamma \geq \tfrac{1}{2} && \text{unconditional stability} \\ &\begin{cases} \gamma \geq \tfrac{1}{2} \\ 0 \leq 2\beta < \gamma \end{cases} && \text{conditional stability} \end{aligned} \tag{2.2.55}$$

$$\Omega_{crit} = \omega \Delta t_{crit} = \frac{\xi(\gamma - \frac{1}{2}) + [\gamma/2 - \beta + \xi^2(\gamma - \frac{1}{2})^2]^{\frac{1}{2}}}{(\gamma/2 - \beta)}$$

2.2.10 Some Commonly Used Members of the Newmark Family

(i) Trapezoidal rule ($\beta = \frac{1}{4}$, $\gamma = \frac{1}{2}$): An implicit second order accurate unconditionally stable algorithm.

(ii) Linear acceleration ($\beta = \frac{1}{6}$, $\gamma = \frac{1}{2}$): An implicit second order accurate algorithm with conditional stability. For the undamped case

$$\Omega_{crit} = 2\sqrt{3}$$

(iii) Central Differences ($\beta = 0$, $\gamma = \frac{1}{2}$): An explicit second order accurate algorithm. The stability limit in the undamped case is:

$$\Omega_{crit} = 2$$

Since this algorithm is explicit (i.e. no system of linear equation need be solved to advance the solution from one step to the next if the mass matrix is diagonal and only external damping is present) it is used efficiently to analyze transient problems when relatively small time steps are required.

2.2.11 Dahlquist's Theorem

To conclude this section about time integration algorithms and justify the use of the Newmark Algorithms in this work, we shall quote Dahlquist's theorem:

1. An explicit A-stable L.M.S. method does not exist.
2. A third order accurate A-stable L.M.S. method does not exist.
3. The second order accurate A-stable L.M.S. method with the smallest error constant is the Trapezoidal Rule.

Remark:

The notion of A-stable method (see [H4]) is slightly different from the notion of spectral stability used to analyze the algorithms in this work, spectral stability being more restrictive.

2.2.12 Residual Form of the I.B.V.P.

Eliminating $\underset{\sim}{d}_{n+1}$ and $\underset{\sim}{V}_{n+1}$ in equation (2.2.30) using (2.2.31) and (2.2.32) results in the following equation:

$$\boxed{\underset{\sim}{M}^* \, \Delta\underset{\sim}{a}_n = \underset{\sim}{R}_n} \tag{2.2.56}$$

where:

$$\Delta\underset{\sim}{a}_n = \underset{\sim}{a}_{n+1} - \underset{\sim}{a}_n \tag{2.2.57}$$

$$\underset{\sim}{M}^* = \underset{\sim}{M} + \Delta t \, \gamma \, \underset{\sim}{C} + \Delta t^2 \beta \, \underset{\sim}{K}$$

$$\underset{\sim}{R}_n = \underset{\sim}{F}_{n+1} - [\underset{\sim}{M} + \Delta t \, \underset{\sim}{C} + \frac{\Delta t^2}{2} \, \underset{\sim}{K}]\underset{\sim}{a}_n - [\underset{\sim}{C} + \Delta t \, \underset{\sim}{K}]\underset{\sim}{V}_n - \underset{\sim}{K} \, \underset{\sim}{d}_n \tag{2.2.59}$$

Thus, for every time step, a set of linear algebraic equations, given by (2.2.56), must be solved in order to advance the solution to the next time step. $\underset{\sim}{a}_{n+1}$ is calculated using (2.2.57) then $\underset{\sim}{V}_{n+1}$ and $\underset{\sim}{d}_{n+1}$ are obtained from equations (2.2.31) and (2.2.32).

2.2.13 Implicit-Explicit Algorithm

There are two classes of algorithms for dynamic problems: implicit and explicit. Implicit algorithms are usually numerically stable and allow the solution to be advanced in relatively large time steps but the cost per time step is high and the storage requirements tend to increase dramatically with

the size of the mesh. Explicit algorithms, on the other hand, are inexpensive per step and require very small storage; however, stability requirements dictate very small time steps.

In everyday structural engineering and solid mechanics problems large finite element models are employed with different mesh refinements at various regions of the same problem and neither method is efficient. Hughes and Liu [H5, H6] suggest the following algorithm which combines both implicit and explicit methods by defining part of the mesh as a region of implicit elements and the rest as a region of explicit elements.

The algorithm is formulated in predictor-corrector fashion and the superscripts E and I denote contributions from explicit and implicit elements respectively.

$$\left.\begin{aligned} \tilde{\underset{\sim}{d}}_{n+1} &= \underset{\sim}{d}_n + \Delta t\, \underset{\sim}{V}_n + \frac{\Delta t^2}{2}(1-2\beta)\underset{\sim}{a}_n \\ \tilde{\underset{\sim}{V}}_{n+1} &= \underset{\sim}{V}_n + \Delta t(1-\gamma)\underset{\sim}{a}_n \end{aligned}\right\} \text{predictor} \qquad (2.2.60),\ (2.2.61)$$

$$\underset{\sim}{M}^*\, \underset{\sim}{a}_{n+1} = \underset{\sim}{F}^*_{n+1} \qquad \text{solution phase} \qquad (2.2.62)$$

$$\left.\begin{aligned} \underset{\sim}{d}_{n+1} &= \tilde{\underset{\sim}{d}}_{n+1} + \Delta t^2 \beta\, \underset{\sim}{a}_{n+1} \\ \underset{\sim}{V}_{n+1} &= \tilde{\underset{\sim}{V}}_{n+1} + \Delta t\, \gamma\, \underset{\sim}{a}_{n+1} \end{aligned}\right\} \text{corrector} \qquad (2.2.63),\ (2.2.64)$$

where:

$$\underset{\sim}{M}^* = (\underset{\sim}{M}^*)^E + (\underset{\sim}{M}^*)^I \qquad (2.2.65)$$

$$(\underset{\sim}{M}^*)^E = \underset{\sim}{M}^E \qquad (2.2.66)$$

$$(\underset{\sim}{M}^*)^I = \underset{\sim}{M}^I + \Delta t\, \gamma\, \underset{\sim}{C}^I + \Delta t^2 \beta\, \underset{\sim}{K}^I \qquad (2.2.67)$$

$$\underset{\sim}{F}^{*}_{n+1} = \underset{\sim}{F}_{n+1} - \underset{\sim}{C}\,\tilde{\underset{\sim}{V}}_{n+1} - \underset{\sim}{K}\,\tilde{\underset{\sim}{d}}_{n+1} \tag{2.2.68}$$

Stability analysis of the above algorithm leads to the following results (see [H6]). For:

$$\gamma \geq \tfrac{1}{2}\,, \quad \beta = (\gamma + \tfrac{1}{2})^2/4 \tag{2.2.69}$$

unconditional stability is achieved in the undamped case for the implicit elements.

And

$$\Omega_{crit} = \omega\,\Delta t_{crit} = \frac{[(\xi^2 + 2\gamma)^{\frac{1}{2}} - \xi]}{\gamma} \tag{2.2.70}$$

for the explicit elements.

2.3 Nonlinear Elastostatics

Two classes of nonlinearity are treated in this section, the first involves nonlinearity due to material behavior and the second includes nonlinear effects due to finite deformations.

For the purpose of the second class of problems rate constitutive equations are presented and incremental linearized stress-strain equations are derived.

2.3.1 Nonlinear Material: Mathematical Statement of the Problem

Assumptions:

1. Small strains
2. No initial stresses
3. Nonlinearity enters only through material behavior

Further, we shall assume that there exists a strain-energy potential function

$\phi(\underset{\sim}{\varepsilon})$, scalar valued, such that:

$$\sigma_{ij}(\underset{\sim}{\varepsilon}) = \partial \phi(\underset{\sim}{\varepsilon})/\partial \varepsilon_{ij} \tag{2.3.1}$$

$$C_{ijk\ell}(\underset{\sim}{\varepsilon}) = \partial \sigma_{ij}(\underset{\sim}{\varepsilon})/\partial \varepsilon_{k\ell} = \frac{\partial^2 \phi(\underset{\sim}{\varepsilon})}{\partial \varepsilon_{ij} \partial \varepsilon_{k\ell}} = C_{k\ell ij}(\underset{\sim}{\varepsilon}) \tag{2.3.2}$$

Let Ω , Γ_{g_i} and Γ_{h_i} be as in section 2.1. Then the mathematical statement of the problem is as follows:

(S)
Given f_i , g_i and h_i as in (2.1.3) - (2.1.5)

Find:

$$u_i : \overline{\Omega} \to \mathbb{R} \tag{2.3.3}$$

such that:

$$\sigma_{ij,j} + f_i = 0 \tag{2.3.4}$$

$$u_i = g_i \quad \text{on} \quad \Gamma_{g_i} \tag{2.3.5}$$

$$\sigma_{ij} n_j = h_i \quad \text{on} \quad \Gamma_{h_i} \tag{2.3.6}$$

where all the definitions and conventions are the same as in section 2.1.1 except for the Cauchy Stress tensor which is given by equation (2.3.1).

2.3.2 Nonlinear Material: Weak and Bubnov-Galerkin Forms

The weighted residual and Bubnov-Galerkin forms of the nonlinear material B.V.P. are same as in sections 2.1.2 and 2.1.3 with $\eta(\underset{\sim}{w}, \underset{\sim}{u})$ replacing $a(\underset{\sim}{w}, \underset{\sim}{u})$ where:

$$\eta(\underset{\sim}{w}, \underset{\sim}{u}) = \int_{\Omega} w_{(i,j)} \sigma_{ij}(\underset{\sim}{\varepsilon}) d\Omega \tag{2.3.7}$$

2.3.3 Nonlinear Material: Matrix Form

We proceed using the vectorial notation of section 2.1.4 and the definitions of section 2.1.5 to obtain the following matrix form of the nonlinear B.V.P.:

$$\underset{\sim}{N}(\underset{\sim}{d}) = \underset{\sim}{F} \tag{2.3.7}$$

where $\underset{\sim}{N}(\underset{\sim}{d})$ is the nonlinear vector of internal forces and $\underset{\sim}{F}$ is the external load vector.

$$\underset{\sim}{N}(\underset{\sim}{d}) = \mathop{\mathbf{A}}_{e=1}^{n_{e\ell}} \underset{\sim}{n}^e(\underset{\sim}{d}^e) \tag{2.3.8}$$

$$\underset{\sim}{n}^e(\underset{\sim}{d}^e) = \{n_p^e(\underset{\sim}{d}^e)\} \tag{2.3.9}$$

$$n_p^e(\underset{\sim}{d}^e) = \underset{\sim}{e}_i^T \int_{\Omega^e} \underset{\sim}{B}_a^T \underset{\sim}{\sigma}(\underset{\sim}{\varepsilon})\, d\Omega \tag{2.3.10}$$

$$\underset{\sim}{F} = \mathop{\mathbf{A}}_{e=1}^{n_{e\ell}} \underset{\sim}{f}^e \tag{2.3.11}$$

$$\underset{\sim}{f}^e = \{f_p^e\} \tag{2.3.12}$$

$$f_p^e = \int_{\Omega^e} N_a\, f_i\, d\Omega + \int_{\Gamma^e \cap \Gamma_{h_i}} N_a\, h_i\, d\Gamma \tag{2.3.13}$$

$$p = n_{ed}\,(a - 1) + i \tag{2.3.14}$$

A set of nonlinear algebraic equations in the form of (2.3.7) cannot be solved directly and some iterative solver is required. In the next section a Newton-Raphson method is applied to equation (2.3.7) and an iterative algorithm is formulated.

2.3.4 The Newton-Raphson Method (N.R.)

The Newton-Raphson method is based on local linearization of the non-linear algebraic equations. When applied to equation (2.3.7) it yields the following algorithm:

$$\begin{cases} \boxed{\underset{\sim}{DN}(\underset{\sim}{d}_{(i)})\ \Delta\underset{\sim}{d}_{(i)} = \underset{\sim}{R}_{(i)}} & (2.3.15) \\ \underset{\sim}{d}_{(i+1)} = \underset{\sim}{d}_{(i)} + \Delta\underset{\sim}{d}_{(i)} & (2.3.16) \\ \underset{\sim}{d}_{(0)} = \underset{\sim}{0} & (2.3.17) \end{cases}$$

where (i) is the iteration index, $\underset{\sim}{DN}(\underset{\sim}{d}_{(i)})$ is the local tangent stiffness at iteration i, $\Delta\underset{\sim}{d}_{(i)}$ is the nodal displacement increment at the i^{th} iteration and $\underset{\sim}{R}_{(i)}$ is the residual load vector.

The matrix forms of $DN(d_i)$ and $\mathbf{R}_i$ are given by:

$$\underset{\sim}{DN}(\underset{\sim}{d}_{(i)}) = \mathop{\mathbf{A}}_{e=1}^{n_{e\ell}} Dn^e(d^e_{(i)}) \qquad (2.3.18)$$

$$Dn^e(\underset{\sim}{d}^e_{(i)}) = [Dn^e_{pq}(d^e_{(i)})] \qquad (2.3.19)$$

$$Dn^e_{pq}(\underset{\sim}{d}^e_{(i)}) = \partial\, n^e_p(\underset{\sim}{d}^e_{(i)})/\partial\, d^e_{q(i)} =$$

$$= \underset{\sim}{e}_i^T \int_{\Omega^e} \underset{\sim}{B}_a^T \frac{\partial\, \underset{\sim}{\sigma}(\underset{\sim}{\varepsilon}_{(i)})}{\partial\, \underset{\sim}{\varepsilon}_{(i)}} \underset{\sim}{B}_b\, d\Omega\, \underset{\sim}{e}_j =$$

$$= \underset{\sim}{e}_i^T \int_{\Omega^e} \underset{\sim}{B}_a^T\, \underset{\sim}{C}(\underset{\sim}{\varepsilon}_{(i)})\, \underset{\sim}{B}_b\, d\Omega\, \underset{\sim}{e}_j \qquad (2.3.20)$$

$$\underset{\sim}{R}_{(i)} = \underset{\sim}{F} - \underset{\sim}{N}(\underset{\sim}{d}_{(i)}) \qquad (2.3.21)$$

Termination of iterative procedures, such as the N.R. method, depends upon some convergence criteria such as:

$$||\underset{\sim}{R}_{(i)}|| \leq \varepsilon_1 \, ||\underset{\sim}{R}_{(0)}|| \tag{2.3.22}$$

$$||\Delta\underset{\sim}{d}_{(i)}|| \leq \varepsilon_2 \, ||\underset{\sim}{d}_{(i+1)}|| \tag{2.3.23}$$

where ε_1 and ε_2 are some small prescribed values.

It should be mentioned here that the N.R. method is rarely applied in its present form and more efficient techniques such as modified N.R., incremental N.R. and quasi-N.R. updates are commonly used. These will be discussed in chapter III.

2.3.5 A Class of Rate-type Constitutive Equations for Inviscid Materials

Consider the following rate-type constitutive equations:

$$\sigma^*_{ij} = \dot{\sigma}_{ij} - S_{ijkl} \, \dot{u}_{[k,l]} - C^*_{ijkl} \, \dot{u}_{(k,l)} \tag{2.3.24}$$

where:

$$\sigma^*_{ij} = C_{ijkl} \, \dot{u}_{(k,l)} \tag{2.3.25}$$

$$S_{ijkl} = (\sigma_{il}\delta_{jk} + \sigma_{jl}\delta_{ik} - \sigma_{ik}\delta_{jl} - \sigma_{jk}\delta_{il})/2 \tag{2.3.26}$$

$$\dot{u}_{(k,l)} = (\dot{u}_{k,l} + \dot{u}_{l,k})/2 \tag{2.3.27}$$

$$\dot{u}_{[k,l]} = (\dot{u}_{k,l} - \dot{u}_{l,k})/2 \tag{2.3.28}$$

Here $\dot{u}_{(k,l)}$ and $\dot{u}_{[k,l]}$ are the symmetric and skew-symmetric parts, respectively, of the velocity gradient and the superposed dot denotes time

differentiation.

σ^*_{ij} is called the "objective stress rate" and is required to transform as a tensor under rigid body motion. The second term in the R.H.S. of (2.3.24) ($S_{ijkl}\,\dot{u}_{[k,l]}$) is the rotational part of the objective stress rate and is determined by the tensor rotation requirement or "objectivity" (for details consult [T1, B1]). The numerical aspects of this term are discussed in a paper by Hughes and Winget [H7]. The third term in the right-hand side of equation (2.3.24) ($C^*_{ijkl}\,\dot{u}_{(k,l)}$) is required to be objective but is otherwise arbitrary. In this work we shall employ the Truesdell Stress Rate where C^*_{ijkl} is given by:

$$C^*_{ijkl} = \hat{C}_{ijkl} = -\sigma_{ij}\,\delta_{kl} + (\sigma_{il}\,\delta_{jk} + \sigma_{jl}\,\delta_{ik} + \sigma_{ik}\,\delta_{jl} + \sigma_{jk}\,\delta_{il})/2 \tag{2.3.29}$$

For further details on this subject the reader is urged to consult [H8, H9].

The tensors S_{ijkl} and C^*_{ijkl} possess the following symmetries:

$$S_{ijkl} = S_{jikl} = -S_{ijlk} \tag{2.3.30}$$

$$\hat{C}_{ijkl} = \hat{C}_{jikl} = \hat{C}_{ijlk} \tag{2.3.31}$$

Combining (2.3.24), (2.3.25) and (2.3.29) we can write the rate constitutive equation as follows:

$$\dot{\sigma}_{ij} = \overline{C}_{ijkl}\,\dot{u}_{(k,l)} + S_{ijkl}\,\dot{u}_{[k,l]} \tag{2.3.32}$$

$$\overline{C}_{ijkl} = C_{ijkl} + \hat{C}_{ijkl} \tag{2.3.33}$$

Note that in (2.3.32), $\overline{C}_{ijkl}$ accounts for material nonlinear response while S_{ijkl} accounts for finite rotations.

Equation (2.3.32) can be approximated by an incremental version using the following approximations:

$$\dot{\underset{\sim}{\sigma}} \cong \frac{\Delta \underset{\sim}{\sigma}}{\Delta t} \tag{2.3.34}$$

$$\dot{\underset{\sim}{u}} \cong \frac{\Delta \underset{\sim}{u}}{\Delta t} \tag{2.3.35}$$

combining the above with equation (2.3.32) we obtain:

$$\Delta\sigma_{ij} = \overline{C}_{ijkl}\, \Delta u_{(k,l)} + S_{ijkl}\, \Delta u_{[k,l]} \tag{2.3.36}$$

The reason for introducing equations of this form is that it allows simple generalization to elastic-plastic phenomena. We note that all (hyper-elastic) constitutive equations can be put in the form (2.3.32) by way of time differentiation.

2.3.6 Linearized Variational Equations

Define a scalar valued functional, f, through:

$$f(\underset{\sim}{\sigma}, \underset{\sim}{u}) = \int_{\Omega} w_{i,j}\, \sigma_{ij}\, d\Omega \tag{2.3.37}$$

where Ω is the current configuration of the domain. Both Ω and $\partial/\partial x_j$ depend on $\underset{\sim}{u}$.

The directional derivative of f is defined by:

$$\frac{df(\underset{\sim}{\sigma} + \varepsilon\, \Delta\underset{\sim}{\sigma}, \underset{\sim}{u} + \varepsilon\, \Delta\underset{\sim}{u})}{d\varepsilon} \tag{2.3.38}$$

where ε is some small parameter. Evaluating (2.3.38) at $\varepsilon=0$ leads to the following linearized variational equation:

$$\int_{\Omega} w_{i,j}(\Delta\sigma_{ij} + \sigma_{ij}\, \Delta u_{k,k} - \Delta u_{j,k}\, \sigma_{ki})d\Omega = \int_{\Omega} w_i\, \rho_i\, d\Omega + \int_{\Gamma_h} w_i\, h_i\, d\Gamma - \int_{\Omega} w_{i,j}\, \sigma_{ij}\, d\Omega \tag{2.3.39}$$

Use the incremental constitutive equation (2.3.36) to obtain:

$$\int_{\Omega} w_{i,j}\ d_{ijkl}\ \Delta u_{k,l}\ d\Omega = \int_{\Omega} w_i\ f_i\ d\Omega + \int_{\Gamma_h} w_i\ h_i\ d\Gamma - \int_{\Omega} w_{i,j}\ \sigma_{ij}\ d\Omega \qquad (2.3.40)$$

where:

$$d_{ijkl} = C_{ijkl} + \sigma_{jl}\ \delta_{ik} \qquad (2.3.41)$$

Remarks

1. The $\sigma_{jl}\ \delta_{ik}$ term in (2.3.41) is the initial stress stiffness.
2. d_{ijkl} possesses the major symmetry $(d_{ijkl} = d_{klij})$ which yields a symmetric tangent stiffness matrix.
3. The right hand side of (2.3.40) is the same as in the materially non-linear small deformation except for Ω and Γ being in the current configuration.

For further details on derivation of linearized variational equations see [H8, H9, H10].

2.3.7 Matrix Counterparts of the Incremental Constitutive Equation

Equation (2.3.36) can be written in the following matrix form:

$$\underset{\sim}{\Delta\sigma} = \overline{\underset{\sim}{C}}\ \underset{\sim}{\Delta\gamma} + \underset{\sim}{S}\ \underset{\sim}{\Delta\theta} \qquad (2.3.42)$$

where

$$\underset{\sim}{\Delta\sigma} = \{\Delta\sigma_I\} \qquad (2.3.43)$$

$$\Delta\sigma_I = \Delta\sigma_{ij} \qquad (2.3.44)$$

$$\overline{\underset{\sim}{C}} = \underset{\sim}{C} + \hat{\underset{\sim}{C}} \qquad (2.3.45)$$

$$\underset{\sim}{C} = [C_{IJ}] \qquad (2.3.46)$$

$$C_{IJ} = C_{ijkl} \tag{2.3.47}$$

$$\hat{\underset{\sim}{C}} = [\hat{C}_{IJ}] \tag{2.3.48}$$

$$\hat{C}_{IJ} = \hat{C}_{ijkl} \tag{2.3.49}$$

$$\Delta\underset{\sim}{\gamma} = \sum_{a=1}^{n_{en}} \underset{\sim}{B}_a^{\gamma}\, \Delta\underset{\sim}{d}_a^{e} = \begin{Bmatrix} \Delta u^{h}_{1,1} \\ \Delta u^{h}_{1,2} + \Delta u^{h}_{2,1} \\ \Delta u^{h}_{2,2} \\ \Delta u^{h}_{3,3} \\ \Delta u^{h}_{2,3} + \Delta u^{h}_{3,2} \\ \Delta u^{h}_{1,3} + \Delta u^{h}_{3,1} \end{Bmatrix} \tag{2.3.50}$$

$$\Delta\underset{\sim}{\theta} = \sum_{a=1}^{n_{en}} \underset{\sim}{B}_a^{\theta}\, \Delta\underset{\sim}{d}_a^{e} = \begin{Bmatrix} \Delta u^{h}_{1,2} - \Delta u^{h}_{2,1} \\ \Delta u^{h}_{2,3} - \Delta u^{h}_{3,2} \\ \Delta u^{h}_{3,1} - \Delta u^{h}_{1,3} \end{Bmatrix} \tag{2.3.51}$$

$$\underset{\sim}{B}_a = \begin{bmatrix} \underset{\sim}{B}_a^{\gamma} \\ \underset{\sim}{B}_a^{\theta} \end{bmatrix} \tag{2.3.52}$$

$$\underset{\sim}{B}_a^{\gamma} = \underline{\text{3-D}} \begin{bmatrix} N_{a,1} & 0 & 0 \\ N_{a,2} & N_{a,1} & 0 \\ 0 & N_{a,2} & 0 \\ 0 & 0 & N_{a,3} \\ 0 & N_{a,3} & N_{a,2} \\ N_{a,3} & 0 & N_{a,1} \end{bmatrix}, \quad \underline{\text{2-D}} \begin{bmatrix} N_{a,1} & 0 \\ N_{a,2} & N_{a,1} \\ 0 & N_{a,2} \end{bmatrix} \tag{2.3.53}$$

$$\underset{\sim}{B}_a^{\theta} = \underset{\text{3-D}}{\begin{bmatrix} N_{a,2} & -N_{a,1} & 0 \\ 0 & N_{a,3} & -N_{a,2} \\ N_{a,1} & 0 & -N_{a,3} \end{bmatrix}}, \underset{\text{2-D}}{\begin{bmatrix} N_{a,2} & -N_{a,1} \end{bmatrix}} \qquad (3.2.54)$$

and the relationship between the indices, (IJ) , and the indices, (ijkl) , is given in the following table:

I / J	i / k	j / l
1	1	1
2	1	2
3	2	2
4	3	3
5	2	3
6	3	1

2.3.8 Large Deformations: Matrix Form

The matrix form emanating from the Bubnov-Galerkin approximation of the weak form (equation (2.3.40)) can be cast into the following N.R. framework:

$$\begin{cases} \underset{\sim}{DN}(\underset{\sim}{d}_{(i)})\ \Delta\underset{\sim}{d}_{(i)} = \underset{\sim}{R}_{(i)} & (2.3.52) \\ \underset{\sim}{d}_{(i+1)} = \underset{\sim}{d}_{(i)} + \Delta\underset{\sim}{d}_{(i)} & (2.3.53) \\ \underset{\sim}{d}_{(0)} = \underset{\sim}{0} & (2.3.54) \end{cases}$$

where $\underset{\sim}{R}_{(i)}$ is the same residual load vector defined in section 2.3.4 except for being integrated over the updated geometry. $\underset{\sim}{DN}(\underset{\sim}{d}_{(i)})$ is given by:

$$\underset{\sim}{DN}(\underset{\sim}{d}_{(i)}) = \mathop{\mathbf{A}}_{e=1}^{n_{el}} \underset{\sim}{Dn}^{e}(\underset{\sim}{d}^{e}_{(i)}) \tag{2.3.55}$$

$$\underset{\sim}{Dn}^{e}(\underset{\sim}{d}^{e}_{(i)}) = [Dn^{e}_{pq}(\underset{\sim}{d}^{e}_{(i)})] \tag{2.3.56}$$

$$Dn^{e}_{pq} = \underset{\sim}{e}_{i}^{T} \int_{\Omega^{e}} \underset{\sim}{B}_{a}^{T} \underset{\sim}{D} \underset{\sim}{B}_{b} \, d\Omega \, \underset{\sim}{e}_{j} \tag{2.3.57}$$

$$\underset{9 \times 9}{\underset{\sim}{D}} = \begin{bmatrix} \underset{6 \times 6}{\underset{\sim}{C}} & \underset{6 \times 3}{\underset{\sim}{0}} \\ \underset{3 \times 6}{\underset{\sim}{0}} & \underset{3 \times 3}{\underset{\sim}{0}} \end{bmatrix} + \underset{\sim}{T} \quad \text{(in 3-D)} \tag{2.3.58}$$

$$\underset{\sim}{T} = \begin{bmatrix}
\sigma_1 & \frac{\sigma_2}{2} & 0 & 0 & 0 & \frac{\sigma_6}{2} & \frac{\sigma_2}{2} & 0 & -\frac{\sigma_6}{2} \\
 & \frac{\sigma_1+\sigma_3}{4} & \frac{\sigma_2}{2} & 0 & \frac{\sigma_6}{4} & \frac{\sigma_5}{4} & \frac{\sigma_3-\sigma_1}{4} & \frac{\sigma_6}{4} & -\frac{\sigma_5}{4} \\
 & & \sigma_3 & 0 & \frac{\sigma_5}{2} & 0 & -\frac{\sigma_2}{2} & \frac{\sigma_5}{2} & 0 \\
 & & & \sigma_4 & \frac{\sigma_5}{2} & \frac{\sigma_6}{2} & 0 & -\frac{\sigma_5}{2} & \frac{\sigma_6}{2} \\
 & & & & \frac{\sigma_3+\sigma_4}{4} & \frac{\sigma_2}{4} & -\frac{\sigma_6}{4} & \frac{\sigma_4-\sigma_3}{4} & \frac{\sigma_2}{4} \\
 & & & & & \frac{\sigma_4+\sigma_1}{4} & \frac{\sigma_5}{4} & -\frac{\sigma_2}{4} & \frac{\sigma_1-\sigma_4}{4} \\
 & \text{symm.} & & & & & \frac{\sigma_1+\sigma_3}{4} & -\frac{\sigma_6}{4} & -\frac{\sigma_5}{4} \\
 & & & & & & & \frac{\sigma_3+\sigma_4}{4} & -\frac{\sigma_2}{4} \\
 & & & & & & & & \frac{\sigma_4+\sigma_1}{4}
\end{bmatrix} \tag{2.3.59}$$

(in 3-D)

2.3.9 Large Rotations

Hughes and Winget [H7] suggest the following constitutive algorithm which allows large rotations within a single increment.

$$\begin{cases} \underset{\sim}{\sigma}_{(i+1)} = \overline{\underset{\sim}{\sigma}}_{(i+1)} + \Delta\underset{\sim}{\sigma}_{(i)} & (2.3.60) \\ \Delta\underset{\sim}{\sigma}_{(i)} = \overline{\underset{\sim}{C}}\, \Delta\underset{\sim}{\gamma}_{(i+\alpha)} & (2.3.61) \\ \overline{\underset{\sim}{\sigma}}_{(i+1)} = \underset{\sim}{Q}\, \underset{\sim}{\sigma}_{(i)}\, \underset{\sim}{Q}^{T} & (2.3.62) \\ \underset{\sim}{Q} = \underset{\sim}{I} + (\underset{\sim}{I} - \alpha\, \Delta\underset{\sim}{\theta}_{(i+\alpha)})^{-1}\, \Delta\underset{\sim}{\theta}_{(i+\alpha)} & (2.3.63) \end{cases}$$

where $\Delta\underset{\sim}{\gamma}_{(i+\alpha)}$ and $\Delta\underset{\sim}{\theta}_{(i+\alpha)}$ are in the α configuration of the body (i.e. $\underset{\sim}{x}_{(i+\alpha)} = \underset{\sim}{x}_{(i)} + \alpha\, \Delta\underset{\sim}{x}_{(i)}$ for every $\underset{\sim}{x} \in \Omega$). Hughes and Winget show that for $\alpha = \frac{1}{2}$ and $|\Delta\underset{\sim}{\theta}| < 180^{o}$ then $\underset{\sim}{Q}$ is a proper orthogonal matrix and the algorithm is second order accurate in $\Delta\underset{\sim}{\theta}$.

2.4 Nonlinear Elastodynamics

In this section the nonlinear elastostatics theory, presented in section 2.3, is generalized for the nonlinear elastodynamic case.

We shall show that most quantities emanating from the nonlinear I.B.V.P. are similar to those previously defined for the nonlinear B.V.P., except for being time-dependent, and that the only additional terms are due to inertia forces and damping effects.

We shall formulate the semi-discrete nonlinear algebraic equations, and, as in the linear case, we shall use the Newmark β-method as the time integrator. Finally, an implicit-explicit algorithm written in N.R. fashion will be presented.

2.4.1 Mathematical Statement of the Nonlinear Initial Boundary Value Problem of Elastodynamics

Let Ω , Γ , Γ_{g_i} and Γ_{h_i} be as in section 2.1 and assume the following scalar valued functions ρ , f_i , g_i , h_i , u_{oi} and $\dot{u}_{oi}$ are given as in (2.2.1)-(2.2.6).

The strong form of the nonlinear I.B.V.P. is:

(S)
Find:

$$u_i: \ \overline{\Omega} \times [0, T] \rightarrow \mathbb{R} \tag{2.4.1}$$

such that:

$$\rho \ddot{u}_i = \sigma_{ij,j} + f_i \quad \text{on} \quad \Omega \times]0,T[\tag{2.4.2}$$

$$u_i = g_i \quad \text{on} \quad \Gamma_{g_i} \times]0,T[\tag{2.4.3}$$

$$\sigma_{ij} n_j = h_i \quad \text{on} \quad \Gamma_{h_i} \times]0,T[\tag{2.4.4}$$

$$\left.\begin{aligned} u_i(\underset{\sim}{x}, 0) &= u_{oi}(\underset{\sim}{x}) \\ \dot{u}_i(\underset{\sim}{x}, 0) &= \dot{u}_{oi}(\underset{\sim}{x}) \end{aligned}\right\} \quad \text{for every } \underset{\sim}{x} \in \Omega_o \tag{2.4.5, 2.4.6}$$

Here Ω , $\overline{\Omega}$, Γ , Γ_{g_i} and Γ_{h_i} are in current geometry (the deformed configuration at time t) and Ω_o is the initial geometry (the undeformed configuration at time t=0).

2.4.2 Weighted Residual Form of the Nonlinear I.B.V.P. of Elastodynamics

Let $\mathcal{S}_i$ and $\mathcal{V}_i$, $i = 1, 2, \ldots, n_{sd}$, be the trial solution spaces and variation spaces, respectively, given by equations (2.1.3a) and (2.1.3b). Proceeding as in section 2.2.2 the following weak form of the nonlinear I.B.V.P. is obtained:

(W)

Given ρ, f_i, g_i, h_i, u_{oi} and $\dot{u}_{oi}$ as before

Find:

$$u_i : [0, T] \rightarrow \mathcal{S}_i \tag{2.4.7}$$

such that for every $w_i \in \mathcal{V}_i$

$$(\underset{\sim}{w}, \rho\ddot{\underset{\sim}{u}}) + \eta(\underset{\sim}{w}, \underset{\sim}{u}) = (\underset{\sim}{w}, \underset{\sim}{f}) + (\underset{\sim}{w}, \underset{\sim}{h})_\Gamma \tag{2.4.8}$$

$$\left.\begin{aligned} (\underset{\sim}{w}, \underset{\sim}{u}(\underset{\sim}{x}, 0) - \underset{\sim}{u}_o) &= 0 \\ (\underset{\sim}{w}, \dot{\underset{\sim}{u}}(\underset{\sim}{x}, 0) - \dot{\underset{\sim}{u}}_o) &= 0 \end{aligned}\right\} \text{ on } \Omega_o \tag{2.4.9}$$

Here $(\cdot, \cdot)$ and $(\cdot, \cdot)_\Gamma$ are the same symmetric bilinear forms defined in section 2.1.2 except for being integrated over the current domain and $\eta(\cdot, \cdot)$ is the same as in the nonlinear B.V.P. case (equation (2.3.7)).

The Bubnov-Galerkin approximation of the weak-form leads to the same results as in section 2.2.3 with $\eta(\cdot, \cdot)$ replacing $a(\cdot, \cdot)$.

2.4.3 Semidiscrete Nonlinear Equation of Motion

Following the same recipe as in section 2.2.4, the following semi-discrete system is obtained:

$$\underset{\sim}{M}\,\ddot{\underset{\sim}{d}}(t) + \underset{\sim}{N}(\underset{\sim}{d}(t)) = \underset{\sim}{F}(t) \tag{2.4.10}$$

$$\underset{\sim}{d}(0) = \underset{\sim}{d}^o \; ; \; \dot{\underset{\sim}{d}}(0) = \dot{\underset{\sim}{d}}^o \tag{2.4.11}$$

where $\underset{\sim}{N}(\underset{\sim}{d}(t))$ is the internal force vector given in section 2.3.4 (integrated over the updated geometry) and $\underset{\sim}{F}(t)$ is the same as in the linear case.

The mass matrix $\underset{\sim}{M}$ is given by:

$$\underset{\sim}{M} = \mathop{\mathbf{A}}_{e=1}^{n_{e\ell}} \underset{\sim}{m}^e \tag{2.4.12}$$

$$\underset{\sim}{m}^e = [m^e_{pq}] \tag{2.4.13}$$

$$m^e_{pq} = \delta_{ij} \int_{\Omega^e} \rho\, N_a N_b \, d\Omega = \delta_{ij} \int_{\Omega^e_o} \rho\, J\, N_a N_b \, d\Omega_o = \delta_{ij} \int_{\Omega^e_o} \rho_o\, N_a N_b \, d\Omega_o \tag{2.4.14}$$

Thus, the mass matrix is integrated once and for all over the initial configuration as in the linear case.

To include damping effects in the system we replace $\underset{\sim}{N}(\underset{\sim}{d})$ by:

$$\underset{\sim}{N} = \underset{\sim}{N}(\underset{\sim}{d}, \underset{\sim}{V}) \tag{2.4.15}$$

The consistent tangent stiffness, $\underset{\sim}{K}_T$, and tangent damping, $\underset{\sim}{C}_T$, are defined as follows:

$$\underset{\sim}{K}_T(\underset{\sim}{d}, \underset{\sim}{V}) = \frac{\partial \underset{\sim}{N}}{\partial \underset{\sim}{d}}(\underset{\sim}{d}, \underset{\sim}{V}) \tag{2.4.16}$$

$$\underset{\sim}{C}_T(\underset{\sim}{d}, \underset{\sim}{V}) = \frac{\partial \underset{\sim}{N}}{\partial \underset{\sim}{V}}(\underset{\sim}{d}, \underset{\sim}{V}) \tag{2.4.17}$$

2.4.4 The Newmark Method

Applying the Newmark method, described in section 2.2.6, to the nonlinear I.B.V.P. leads to the following system of nonlinear algebraic equations:

$$\underset{\sim}{M}\,\underset{\sim}{a}_{n+1} + \underset{\sim}{N}(\underset{\sim}{d}_{n+1}, \underset{\sim}{V}_{n+1}) = \underset{\sim}{F}_{n+1} \tag{2.4.18}$$

$$\underset{\sim}{d}_{n+1} = \underset{\sim}{d}_n + \Delta t\, \underset{\sim}{V}_n + \frac{\Delta t^2}{2}(1 - 2\beta)\underset{\sim}{a}_n + \Delta t^2 \beta\, \underset{\sim}{a}_{n+1} \tag{2.4.19}$$

$$\underset{\sim}{V}_{n+1} = \underset{\sim}{V}_n + \Delta t(1 - \gamma)\underset{\sim}{a}_n + \Delta t\, \gamma\, \underset{\sim}{a}_{n+1} \tag{2.4.20}$$

$$\underset{\sim}{d}_o = \underset{\sim}{d}^o \tag{2.4.21}$$

$$\underset{\sim}{V}_o = \dot{\underset{\sim}{d}}^o \tag{2.4.22}$$

$$\underset{\sim}{a}_o = \underset{\sim}{M}^{-1}(\underset{\sim}{F}_o - \underset{\sim}{N}(\underset{\sim}{d}_o, \underset{\sim}{V}_o)) \tag{2.4.23}$$

As in the nonlinear static case, equations (2.4.18)-(2.4.23) cannot be solved directly and for each time step an iterative solver is required. In the following sections a predictor multi-corrector algorithm is presented and generalized to include implicit-explicit formulation.

2.4.5 Predictor Multi-Corrector Algorithm

We shall assume that our mesh consists of implicit and explicit elements denoted by I and E superscripts respectively and further, we shall assume that the internal force vector, $\underset{\sim}{N}$, admits the following decomposition:

$$\underset{\sim}{N}(\underset{\sim}{d}, \underset{\sim}{V}) = \underset{\sim}{N}^I(\underset{\sim}{d}, \underset{\sim}{V}) + \underset{\sim}{N}^E(\underset{\sim}{d}, \underset{\sim}{V}) \tag{2.4.24}$$

Let (i) be an iteration index and n be a time step number. Consider the following predictor multi-corrector scheme:

$$\left.\begin{aligned} \tilde{\underset{\sim}{d}}_{n+1} &= \underset{\sim}{d}_n + \Delta t\, \underset{\sim}{V}_n + \frac{\Delta t^2}{2}(1 - 2\beta)\underset{\sim}{a}_n \qquad (2.4.25)\\ \tilde{\underset{\sim}{V}}_{n+1} &= \underset{\sim}{V}_n + \Delta t(1 - \gamma)\underset{\sim}{a}_n \qquad (2.4.26) \end{aligned}\right\} \text{ Predictor}$$

$$\left.\begin{aligned} \underset{\sim}{d}_{n+1}^{(i+1)} &= \underset{\sim}{\tilde{d}}_{n+1} + \Delta t^2 \beta\, \underset{\sim}{a}_{n+1}^{(i+1)} = \underset{\sim}{d}_{n+1}^{(i)} + \Delta \underset{\sim}{d} \\ \underset{\sim}{V}_{n+1}^{(i+1)} &= \underset{\sim}{\tilde{V}}_{n+1} + \Delta t\, \gamma\, \underset{\sim}{a}_{n+1}^{(i+1)} = \underset{\sim}{V}_{n+1}^{(i)} + \Delta \underset{\sim}{V} \end{aligned}\right\} \text{Corrector} \qquad (2.4.27),\ (2.4.28)$$

$$\underset{\sim}{a}_{n+1}^{(i+1)} = \underset{\sim}{a}_{n+1}^{(i)} + \Delta \underset{\sim}{a} \qquad (2.4.29)$$

Solving equations (2.4.27) and (2.4.28) for $\Delta \underset{\sim}{d}$ and $\Delta \underset{\sim}{V}$ we obtain:

$$\Delta \underset{\sim}{d} = \Delta t^2 \beta\, \Delta \underset{\sim}{a} \qquad (2.4.30)$$

$$\Delta \underset{\sim}{V} = \Delta t\, \gamma\, \Delta \underset{\sim}{a} \qquad (2.4.31)$$

Using the above approximations, $\underset{\sim}{d}_{n+1}^{(i+1)}$, $\underset{\sim}{V}_{n+1}^{(i+1)}$ and $\underset{\sim}{a}_{n+1}^{(i+1)}$ can be written in terms of $\Delta \underset{\sim}{a}$ and we can define the following vector valued function, $\underset{\sim}{f}$:

$$\underset{\sim}{f}(\Delta \underset{\sim}{a}) = \underset{\sim}{F}_{n+1} - \underset{\sim}{M}\, \underset{\sim}{a}_{n+1}^{(i+1)} - \underset{\sim}{N}^{I}(\underset{\sim}{d}_{n+1}^{(i+1)}, \underset{\sim}{V}_{n+1}^{(i+1)}) - \underset{\sim}{N}^{E}(\underset{\sim}{d}_{n+1}^{(i+1)}, \underset{\sim}{V}_{n+1}^{(i+1)}) = \underset{\sim}{0} \qquad (2.4.32)$$

where superscripts I and E denote implicit and explicit parts, respectively, and $\underset{\sim}{f}$, is the so-called "kick" force or "out-off-balance" force. Linearizing (2.4.32) via:

$$\underset{\sim}{f}(\underset{\sim}{0}) + \frac{\partial \underset{\sim}{f}}{\partial \Delta \underset{\sim}{a}}(\underset{\sim}{0})\, \Delta \underset{\sim}{a} = \underset{\sim}{0} \qquad (2.4.33)$$

and combining the linearized equation with the predictor-corrector equations, we obtain the following algorithm :

$$\left.\begin{aligned} \underset{\sim}{d}_{n+1}^{(0)} &= \underset{\sim}{\tilde{d}}_{n+1} = \underset{\sim}{d}_{n} + \Delta t\, \underset{\sim}{V}_{n} + \frac{\Delta t^2}{2}(1 - 2\beta)\underset{\sim}{a}_{n} \\ \underset{\sim}{V}_{n+1}^{(0)} &= \underset{\sim}{\tilde{V}}_{n+1} = \underset{\sim}{V}_{n} + \Delta t(1 - \gamma)\underset{\sim}{a}_{n} \end{aligned}\right\} \text{Predictor phase} \qquad (2.4.34),\ (2.4.35)$$

$$\boxed{\underset{\sim}{M}^* \, \Delta\underset{\sim}{a} = \underset{\sim}{R}_{n+1}^{(i)}} \qquad \text{Solution phase} \tag{2.4.36}$$

$$\left.\begin{aligned} \underset{\sim}{a}_{n+1}^{(i+1)} &= \underset{\sim}{a}_{n+1}^{(i)} + \Delta\underset{\sim}{a} && (2.4.37)\\ \underset{\sim}{V}_{n+1}^{(i+1)} &= \underset{\sim}{V}_{n+1}^{(i)} + \Delta t\, \gamma\, \Delta\underset{\sim}{a} && (2.4.38)\\ \underset{\sim}{d}_{n+1}^{(i+1)} &= \underset{\sim}{d}_{n+1}^{(i)} + \Delta t^2 \beta\, \Delta\underset{\sim}{a} && (2.4.39) \end{aligned}\right\} \quad \text{Corrector phase}$$

where:

$$\underset{\sim}{M}^* = \underset{\sim}{M} + \Delta t\, \gamma\, \underset{\sim}{C}_T^I + \Delta t^2 \beta\, \underset{\sim}{K}_T^I \tag{2.4.40}$$

$$\underset{\sim}{C}_T^I = \frac{\partial \underset{\sim}{N}^I}{\partial \underset{\sim}{V}} \left(\underset{\sim}{d}_{n+1}^{(i)} \, , \, \underset{\sim}{V}_{n+1}^{(i)}\right) \tag{2.4.41}$$

$$\underset{\sim}{K}_T^I = \frac{\partial \underset{\sim}{N}^I}{\partial \underset{\sim}{d}} \left(\underset{\sim}{d}_{n+1}^{(i)} \, , \, \underset{\sim}{V}_{n+1}^{(i)}\right) \tag{2.4.42}$$

$$\underset{\sim}{R}_{n+1}^{(i)} = \underset{\sim}{F}_{n+1} - \underset{\sim}{M}\, \underset{\sim}{a}_{n+1}^{(i)} - \underset{\sim}{N}\left(\underset{\sim}{d}_{n+1}^{(i)} \, , \, \underset{\sim}{V}_{n+1}^{(i)}\right) \tag{2.4.43}$$

As for the nonlinear static case, we need some convergence criteria to terminate the iterative solution.

Two candidate criteria are:

$$||\underset{\sim}{R}_{n+1}^{(i)}|| \leq \varepsilon_1 \, ||\underset{\sim}{R}_o|| \tag{2.4.44}$$

$$||\Delta\underset{\sim}{a}|| \leq \varepsilon_2 \, ||\underset{\sim}{a}_{n+1}^{(i+1)}|| \tag{2.4.45}$$

where ε_1 and ε_2 are some small prescribed values and $||\cdot||$ denotes the Euclidian norm.

2.5 Final Remarks

In this chapter we formed some linear algebraic equation systems associated with various areas of finite element applications in elasticity. We should emphasize that other classes of problems in mechanical engineering can be cast into similar frame-work of finite element formulation and lead to similar linear algebraic systems.

All results of this chapter lead to linear systems of the form:

$$\underset{\sim}{A}\,\underset{\sim}{x} = \underset{\sim}{b} \tag{2.4.46}$$

and a summary of the definitions of $\underset{\sim}{A}$, $\underset{\sim}{x}$ and $\underset{\sim}{b}$ for the cases treated in this chapter are given in table 1.

CHAPTER III

PARABOLIC REGULARIZATION USING ELEMENT-BY-ELEMENT ALGORITHMS

In this chapter, a solution technique for linear algebraic systems (i.e. $\underset{\sim}{A}\,\underset{\sim}{x} = \underset{\sim}{b}$) via a parabolic regularization method is derived, element-by-element approximation algorithms are presented and studied and, finally, improved numerical solution schemes are introduced and implemented with the preceding theory.

3.1 Parabolic Regularization Method

The problem in hand is to solve

$$\underset{\sim}{A}\,\underset{\sim}{x} = \underset{\sim}{b} \tag{3.1.1}$$

We shall assume, throughout this chapter, that $\underset{\sim}{A}$ is a given constant, symmetric positive-definite matrix and $\underset{\sim}{b}$ is a given vector.

There are two main methods of transforming a static problem, as (3.1.1), into a dynamic one:

1. Dynamic relaxation method
2. Parabolic regularization method

The first method transform the static problem to a second order problem:

$$\begin{cases} \text{Find the steady-state solution of:} \\ \underset{\sim}{M}\,\ddot{\underset{\sim}{x}} + \underset{\sim}{C}\,\dot{\underset{\sim}{x}} + \underset{\sim}{A}\,\underset{\sim}{x} = \underset{\sim}{F}(\tau) = \underset{\sim}{b}\,u(\tau) & (3.1.2) \\ \underset{\sim}{x}(\tau = 0) = \dot{\underset{\sim}{x}}(\tau = 0) = \underset{\sim}{0} & (3.1.3) \end{cases}$$

where $\underset{\sim}{M}$ and $\underset{\sim}{C}$ are an artificial mass matrix and damping matrix, respectively, τ is the relaxation time variable, $u(\tau)$ is a step function and the superposed dots denote time differentiation (i.e. $\dot{\underset{\sim}{x}} = \frac{\partial x}{\partial \tau}$) . For further information on the dynamic relaxation method consult [O1, D1, W2, W3, B2].

The parabolic regularization method transforms the static problem into a first order parabolic problem as follows:

$$\begin{cases} \text{Find the steady-state solution of:} \\ \underset{\sim}{M}\,\dot{\underset{\sim}{x}} + \underset{\sim}{A}\,\underset{\sim}{x} = \underset{\sim}{F}(\tau) = \underset{\sim}{b}\,u(\tau) & (3.1.4) \\ \underset{\sim}{x}(\tau = 0) = \underset{\sim}{0} & (3.1.5) \end{cases}$$

where $\underset{\sim}{M}$ is a pseudo-mass matrix. We shall choose $\underset{\sim}{M}$ to be a diagonal positive-definite matrix and the choice of this matrix will be discussed later in this chapter. The asymptotic solution of equation (3.1.4) (i.e. $\underset{\sim}{x}(\tau \to \infty)$) satisfies:

$$\underset{\sim}{x}(\tau \to \infty) \to \underset{\sim}{A}^{-1}\,\underset{\sim}{b} \qquad (3.1.6)$$

In other words, the steady-state solution of the regularized problem (3.1.4) converges to the solution of the static problem (3.1.1).

To numerically solve a first order parabolic equation, such as (3.1.4), we need to employ some time integration algorithm. In the next section a suitable family of integrators is presented.

3.1.1 The Generalized α-Method

The generalized α-method is given by the following equations:

$$\begin{cases} \underset{\sim}{M}\,\dot{\underset{\sim}{x}}_{n+1} + \underset{\sim}{A}\,\underset{\sim}{x}_{n+1} = \underset{\sim}{F}_{n+1} & (3.1.7) \\ \underset{\sim}{x}_{n+1} = \underset{\sim}{x}_n + \Delta\tau(1-\alpha)\dot{\underset{\sim}{x}}_n + \Delta\tau\,\alpha\,\dot{\underset{\sim}{x}}_{n+1} & (3.1.8) \end{cases}$$

where the n subscript is a step number (i.e. $\underset{\sim}{x}_n \cong \underset{\sim}{x}(\tau_n) = \underset{\sim}{x}(n\,\Delta\tau)$) and α is a parameter which controls the stability and accuracy of the method.

Eliminating $\dot{\underset{\sim}{x}}_{n+1}$ from the equations, the following form of the α-method is obtained:

$$(\underset{\sim}{M} + \alpha\,\Delta\tau\,\underset{\sim}{A})\underset{\sim}{x}_{n+1} = (\underset{\sim}{M} - (1-\alpha)\Delta\tau\,\underset{\sim}{A})\underset{\sim}{x}_n + \Delta\tau\,\underset{\sim}{F}_{n+\alpha} \tag{3.1.9}$$

where:

$$\underset{\sim}{F}_{n+\alpha} \cong \underset{\sim}{F}(\tau_{n+\alpha}) = (1-\alpha)\underset{\sim}{F}_n + \alpha\,\underset{\sim}{F}_{n+1} \tag{3.1.10}$$

$$\tau_{n+\alpha} = \tau_n + \alpha\,\Delta\tau \tag{3.1.11}$$

Before exploring the stability of the α-method, we must reduce equation (3.1.9) to a single-degree-of-freedom (S.D.O.F.) equation by modal decomposition.

Repeating the process of section (2.2.6) with $\det(\underset{\sim}{A} - \lambda\,\underset{\sim}{M})$ as the characteristic equation, the following S.D.O.F. equation is obtained:

$$(1 + \alpha\,\Delta\tau\,\lambda)\overline{x}_{n+1} = [1 - (1-\alpha)\Delta\tau\,\lambda]\overline{x}_n + \overline{F}_{n+\alpha} \tag{3.1.12}$$

where the superposed bar denotes generalized quantity and λ is an eigenvalue.

3.1.2. Stability Analysis of the α-Method

For stability analysis the homogeneous version of (3.1.12) (i.e. $\overline{F}_{n+\alpha} = 0$) is considered.

Define

$$\overline{x}_{n+1} = \eta(\alpha, \Delta\tau)\overline{x}_n \tag{3.1.13}$$

where $\eta(\cdot, \cdot)$ is the amplification function.

For stability we require that

$$|\eta| \leq 1 \tag{3.1.14}$$

From (3.1.12) and (3.1.13) we obtain:

$$\eta(\alpha, \Delta\tau) = \frac{1 - (1-\alpha)\,\Delta\tau\lambda}{1 + \alpha\,\Delta\tau\lambda} \tag{3.1.15}$$

Applying the stability condition to η, the following results are obtained:

$$\begin{cases} 1 \geq \alpha \geq \frac{1}{2} & \text{unconditional stability} \\ 0 \leq \alpha < \frac{1}{2} & \Delta\tau_{crit} = \dfrac{2}{(1-2\alpha)\lambda_{max}} \end{cases} \tag{3.1.16}$$

3.1.3 Accuracy Analysis of the α-Method

The local truncation error, (see section 2.2.7), is defined as follows:

$$T(\tau_n) = y(\tau_{n+1}) - \eta(\alpha, \Delta\tau)\, y(\tau_n) - L(\tau_n) \tag{3.1.17}$$

Here $y(\tau)$ is the solution of:

$$\dot{y}(\tau) + \lambda\, y(\tau) = \overline{F}(\tau) \tag{3.1.18}$$

and $L(\tau_n)$ is given by:

$$L(\tau_n) = \frac{1}{1 + \alpha\,\Delta\tau\lambda}\ \overline{F}(\tau_n) \tag{3.1.19}$$

$T(\tau_n)$ measures how well the exact solution (i.e. $y(\tau)$) satisfies the discretized algorithmic equation (equation (3.1.12)). In addition, if we

assume that $y(\tau)$ satisfies the algorithmic equation at τ_n, then $T(\tau_n)$ measures the "local" error introduced by the algorithmic approximation between step n and step n+1.

Expanding $y(\tau_{n+1})$ about $y(\tau_n)$ by a Taylor series with remainder, use of equation (3.1.18) and its derivatives w.r.t τ yields the following results:

$$T(\tau_n) = C(\Delta\tau)^{k+1} \tag{3.1.20}$$

where C is some constant determined by the remainder term of the Taylor expansion, and k is defined to be the order of accuracy of the algorithm.

One obtains the following results for k :

$$\begin{cases} k = 2 & \text{if } \alpha = \tfrac{1}{2} \\ k = 1 & \text{if } \alpha \neq \tfrac{1}{2}, \ \alpha \in [0, 1] \end{cases} \tag{3.1.21}$$

The generalized α-method is second order accurate in τ for $\alpha = \frac{1}{2}$ (trapezoidal rule) and first order accurate in τ otherwise.

3.1.4 Numerical Dissipation

The exact solution of the homogeneous version of equation (3.1.18) is:

$$y(\tau_n) = \bar{x}_0 e^{-\lambda n \Delta\tau} \tag{3.1.22}$$

The algorithmic solution of the homogeneous problem leads to:

$$\bar{x}_n = \eta^n(\alpha, \Delta\tau)\, \bar{x}_0 \triangleq e^{-\lambda^* n \Delta\tau}\, \bar{x}_0 \tag{3.1.23}$$

where λ^* is the algorithmic eigenvalue.

Solving (3.1.23) for λ^*, we obtain:

$$\lambda^* = -\frac{1}{\Delta\tau} \ln \eta(\alpha, \Delta\tau) \tag{3.1.24}$$

Define:

$$\Omega \equiv \Delta\tau\lambda \tag{3.1.25}$$

$$\Omega^* \equiv \Delta\tau\lambda^* \tag{3.1.26}$$

Then:

$$\Omega^* = -\ln\left[\frac{1 - (1-\alpha)\Omega}{1 + \alpha\,\Omega}\right] \tag{3.1.27}$$

Define the algorithm dissipation ratio, ξ^*, as:

$$\xi^* = \frac{\Omega^*}{\Omega} \tag{3.1.28}$$

Figure 2 shows the algorithmic amplification factor, η, plotted vs. log (Ω) for various values of α.

Clearly, the algorithm dissipation increases as α approaches one (i.e. Backward Euler).

3.1.5 Operator Approximations

Equation (3.1.9) can be written in the following form:

$$\underset{\sim}{M}^{\frac{1}{2}}\,\underset{\sim}{x}_{n+1} = \underset{\sim}{\hat{A}}(\alpha, \Delta\tau)\underset{\sim}{M}^{\frac{1}{2}}\,\underset{\sim}{x}_n + \Delta\tau\,\underset{\sim}{\hat{B}}\,\underset{\sim}{M}^{-\frac{1}{2}}\,\underset{\sim}{F}_{n+\alpha} \tag{3.1.29}$$

where:

$$\hat{\underset{\sim}{C}} = \underset{\sim}{M}^{-\frac{1}{2}} \underset{\sim}{A} \underset{\sim}{M}^{-\frac{1}{2}} \tag{3.1.30}$$

$$\hat{\underset{\sim}{B}}(\alpha, \Delta\tau) = [\underset{\sim}{1} + \alpha\Delta\tau \hat{\underset{\sim}{C}}]^{-1} \tag{3.1.31}$$

$$\hat{\underset{\sim}{A}}(\alpha, \Delta\tau) = \hat{\underset{\sim}{B}}(\alpha, \Delta\tau)[\underset{\sim}{1} - (1 - \alpha)\Delta\tau \hat{\underset{\sim}{C}}] \tag{3.1.32}$$

Consider the following approximations:

$$\tilde{\underset{\sim}{A}}(\alpha, \Delta\tau) = \hat{\underset{\sim}{A}}(\alpha, \Delta\tau) + 0(\Delta\tau^{\ell}) \tag{3.1.33}$$

$$\tilde{\underset{\sim}{B}}(\alpha, \Delta\tau) = \hat{\underset{\sim}{B}}(\alpha, \Delta\tau) + 0(\Delta\tau^{m}) \tag{3.1.34}$$

Assume equation (3.1.29) emanates from a generalized α-method which is k^{th} order accurate.

By writing the local truncation error with the approximate operators $\tilde{\underset{\sim}{A}}$ and $\tilde{\underset{\sim}{B}}$ replacing the exact algorithmic operators $\hat{\underset{\sim}{A}}$ and $\hat{\underset{\sim}{B}}$ and requiring that the overall order of accuracy be retained, yields the following relations:

$$\begin{aligned} \ell &= k + 1 \\ m &= k \end{aligned} \tag{3.1.35}$$

which means that the approximation to the $\hat{\underset{\sim}{A}}$ operator should be one order higher than the order of accuracy of the original equations and that the approximation of the $\hat{\underset{\sim}{B}}$ operator should be of the same order as the original equations so as to retain the order of accuracy of the governing algorithm.

In the following sections element-by-element approximations of the global operators are presented and studied.

3.2 Element-By-Element Algorithms

Assume that the $\underset{\sim}{A}$ matrix of equation (3.1.1) admits the following decomposition:

$$\underset{\sim}{A} = \sum_{e=1}^{n_{e\ell}} \underset{\sim}{A}^e = \mathop{\mathbf{A}}_{e=1}^{n_{e\ell}} \underset{\sim}{a}^e \tag{3.2.1}$$

where $\underset{\sim}{A}^e$ and $\underset{\sim}{a}^e$ are the e^{th} element contributions to $\underset{\sim}{A}$ in global and local degrees of freedom, respectively, and $\mathbf{A}$ is the finite element assembly operator.

Define:

$$\hat{\underset{\sim}{C}}^e \equiv \underset{\sim}{M}^{-\frac{1}{2}} \underset{\sim}{A}^e \underset{\sim}{M}^{-\frac{1}{2}} \tag{3.2.2}$$

$$\hat{\underset{\sim}{B}}^e \equiv (\underset{\sim}{1} + \alpha\Delta\tau\, \hat{\underset{\sim}{C}}^e)^{-1} \tag{3.2.3}$$

$$\hat{\underset{\sim}{A}}^e \equiv \hat{\underset{\sim}{B}}^e(\underset{\sim}{1} - (1-\alpha)\Delta\tau\, \hat{\underset{\sim}{C}}^e) \tag{3.2.4}$$

We shall assume that $\underset{\sim}{A}^e$, $e = 1, 2, \ldots, n_{e\ell}$, are symmetric positive semi-definite matrices.

Next we shall present element-by-element algorithms based on element decomposition of $\underset{\sim}{A}$.

3.2.1 One-Pass Element-By-Element Algorithm

Consider the following operator approximations:

$$\tilde{\underset{\sim}{A}}(\alpha, \Delta\tau) = \prod_{e=1}^{n_{e\ell}} \hat{\underset{\sim}{A}}^e(\alpha, \Delta\tau) \tag{3.2.5}$$

$$\tilde{\underset{\sim}{B}}(\alpha, \Delta\tau) = \prod_{e=1}^{n_{e\ell}} \hat{\underset{\sim}{B}}^e(\alpha, \Delta\tau) \tag{3.2.6}$$

where π is the product operator.

Substituting $\tilde{\underset{\sim}{A}}$ for $\hat{\underset{\sim}{A}}$ and $\tilde{\underset{\sim}{B}}$ for $\hat{\underset{\sim}{B}}$ in equation (3.1.29) we obtain the following one-pass element-by-element algorithm :

$$\underset{\sim}{M}^{\frac{1}{2}}\,\underset{\sim}{x}_{n+1} = \left[\prod_{e=1}^{n_{e\ell}} \underset{\sim}{\hat{A}}^{e}(\alpha,\ \Delta\tau)\right] \underset{\sim}{M}^{\frac{1}{2}}\,\underset{\sim}{x}_{n} + \Delta\tau \left[\prod_{e=1}^{n_{e\ell}} \underset{\sim}{\hat{B}}^{e}(\alpha,\ \Delta\tau)\right] \underset{\sim}{M}^{-\frac{1}{2}}\,\underset{\sim}{F}_{n+\alpha} \qquad (3.2.7)$$

Next, the stability and accuracy of the suggested algorithm will be studied.

3.2.2 One-Pass Element-By-Element: Stability Analysis

Consider the homogeneous problem (i.e. set $\underset{\sim}{F}_{n+\alpha} = \underset{\sim}{0}$ in (3.2.7)) and define:

$$\underset{\sim}{y}_0 \equiv \underset{\sim}{M}^{\frac{1}{2}}\,\underset{\sim}{x}_n \qquad (3.2.8)$$

Now consider the following sequence of problems:

$$\underset{\sim}{y}_e = \underset{\sim}{\hat{A}}^{e}\,\underset{\sim}{y}_{e-1} \qquad e = 1\ ,\ 2\ ,\ \ldots,\ n_{e\ell} \qquad (3.2.9)$$

Each problem in (3.2.9) corresponds to the generalized α scheme with $\underset{\sim}{1}$ for the mass matrix (capacity matrix in the context of heat conduction case) and $\underset{\sim}{\hat{C}}^{e}$ for the $\underset{\sim}{A}$ array (conductivity matrix in the context of heat conduction) therefore, using the stability condition for the generalized α-method (see section 3.1.2), we have:

For $\alpha \geq \frac{1}{2}$

$$\underset{\sim}{y}_e^{T} \cdot \underset{\sim}{y}_e \leq \underset{\sim}{y}_{e-1}^{T} \cdot \underset{\sim}{y}_{e-1} \qquad e = 1\ ,\ 2\ ,\ \ldots,\ n_{e\ell} \qquad (3.2.10)$$

Repeated use of the above in (3.2.9) leads to:

For $\alpha \geq \frac{1}{2}$

$$\underset{\sim}{y}_{n_{e\ell}}^{T} \cdot \underset{\sim}{y}_{n_{e\ell}} \leq \underset{\sim}{y}_0^{T} \cdot \underset{\sim}{y}_0 \qquad (3.2.11)$$

Substituting for $\underset{\sim}{y}_{n_{e\ell}}$ and $\underset{\sim}{y}_0$ we get:

For $\alpha \geq \frac{1}{2}$

$$(\underset{\sim}{x}_{n+1},\ \underset{\sim}{M}\,\underset{\sim}{x}_{n+1}) \leq (\underset{\sim}{x}_n,\ \underset{\sim}{M}\,\underset{\sim}{x}_n) \qquad (3.2.12)$$

Conclusion: the element-by-element approximation retains the stability range of the governing generalized α-scheme.

3.2.3 One-Pass Element-By-Element: Accuracy Analysis

In section 3.1.5 we showed that if the order of accuracy, k, of the governing algorithm is to be retained the approximation to the $\hat{\underset{\sim}{A}}$ operator should be of order $(k + 1)$ and the approximation for the $\hat{\underset{\sim}{B}}$ operator should be of order k. Therefore, for generalized α-method (say trapezoidal rule)

$$\ell = k + 1 = 3 \quad ; \quad m = k = 2 \tag{3.2.13}$$

To find the order of approximation we must expand the operators using a Taylor series in $\Delta\tau$ as follows

$$\hat{\underset{\sim}{A}}(\tfrac{1}{2}, \Delta\tau) = \underset{\sim}{1} - \Delta\tau\, \hat{\underset{\sim}{C}} + \frac{\Delta\tau^2}{2}\, \hat{\underset{\sim}{C}}^2 + O(\Delta\tau^3) \tag{3.2.14}$$

$$\hat{\underset{\sim}{B}}(\tfrac{1}{2}, \Delta\tau) = \underset{\sim}{1} - \frac{\Delta\tau}{2}\, \hat{\underset{\sim}{C}} + O(\Delta\tau^2) \tag{3.2.15}$$

$$\tilde{\underset{\sim}{A}}(\tfrac{1}{2}, \Delta\tau) = \prod_{e=1}^{n_{e\ell}} \hat{\underset{\sim}{A}}^e(\tfrac{1}{2}, \Delta\tau) = \underset{\sim}{1} - \Delta\tau\, \hat{\underset{\sim}{C}} + \frac{\Delta\tau^2}{2}\, \hat{\underset{\sim}{C}}^2 +$$

$$+ \frac{\Delta\tau^2}{2} \sum_{e=1}^{n_{e\ell}} \sum_{f=e+1}^{n_{e\ell}} (\hat{\underset{\sim}{C}}^e\, \hat{\underset{\sim}{C}}^f - \hat{\underset{\sim}{C}}^f\, \hat{\underset{\sim}{C}}^e) + O(\Delta\tau^3) \tag{3.2.16}$$

$$\tilde{\underset{\sim}{B}}(\tfrac{1}{2}, \Delta\tau) = \prod_{e=1}^{n_{e\ell}} \hat{\underset{\sim}{B}}^e(\tfrac{1}{2}, \Delta\tau) = \underset{\sim}{1} - \frac{\Delta\tau}{2}\, \hat{\underset{\sim}{C}} + O(\Delta\tau^2) \tag{3.2.17}$$

Clearly, $\tilde{\underset{\sim}{B}}(\tfrac{1}{2}, \Delta\tau)$ satisfies the accuracy condition $(\tilde{\underset{\sim}{B}} = \hat{\underset{\sim}{B}} + O(\Delta\tau^2))$. However, $\tilde{\underset{\sim}{A}}(\tfrac{1}{2}, \Delta\tau)$ has an additional second order term which violates the accuracy

requirements for this term (unless the element matrices, $\hat{\underset{\sim}{C}}^e$'s commute, which is not the case in general).

The one-pass element-by-element algorithm is, therefore, only first order accurate (even for the trapezoidal rule). Note that the additional term in the $\tilde{\underset{\sim}{A}}(\frac{1}{2}, \Delta\tau)$ expansion is a skew-symmetric term (the $\frac{\Delta\tau^2}{2} \sum_{e=1}^{n_{el}} \sum_{f=e+1}^{n_{el}}$ $(\hat{\underset{\sim}{C}}^e \hat{\underset{\sim}{C}}^f - \hat{\underset{\sim}{C}}^f \hat{\underset{\sim}{C}}^e)$ term). Next we shall introduce a symmetric two-pass element-by-element algorithm which is second order accurate.

3.2.4 Two-Pass Element-By-Element Algorithm

Consider the following operator approximations:

$$\tilde{\underset{\sim}{A}}(\alpha, \Delta\tau) = \left[\prod_{e=n_{el}}^{1} \hat{\underset{\sim}{A}}^e(\alpha, \Delta\tau/2)\right]\left[\prod_{e=1}^{n_{el}} \hat{\underset{\sim}{A}}^e(\alpha, \Delta\tau/2)\right] \tag{3.2.18}$$

$$\tilde{\underset{\sim}{B}}(\alpha, \Delta\tau) = \left[\prod_{e=n_{el}}^{1} \hat{\underset{\sim}{B}}^e(\alpha, \Delta\tau/2)\right]\left[\prod_{e=1}^{n_{el}} \hat{\underset{\sim}{B}}^e(\alpha, \Delta\tau/2)\right] \tag{3.2.19}$$

Substituting the above approximations into equation (3.1.29), we obtain the following algorithm:

$$\begin{aligned} \underset{\sim}{M}^{\frac{1}{2}} \underset{\sim}{x}_{n+1} = {} & \prod_{e=n_{el}}^{1} \hat{\underset{\sim}{A}}^e(\alpha, \Delta\tau/2) \prod_{e=1}^{n_{el}} \hat{\underset{\sim}{A}}^e(\alpha, \Delta\tau/2)\underset{\sim}{M}^{\frac{1}{2}} \underset{\sim}{x}_n + \\ & + \Delta\tau \prod_{e=n_{el}}^{1} \hat{\underset{\sim}{B}}^e(\alpha, \Delta\tau/2) \prod_{e=1}^{n_{el}} \hat{\underset{\sim}{B}}^e(\alpha, \Delta\tau/2)\underset{\sim}{M}^{-\frac{1}{2}} \underset{\sim}{F}_{n+\alpha} \end{aligned} \tag{3.2.20}$$

In the next sections we shall present the stability and accuracy analyses of the two-pass algorithm.

3.2.5 Two-Pass Element-By-Element: Stability Analysis

Following the same arguments as for the one-pass element-by-element algorithm with $\underset{\sim}{1}$ as the capacity matrix and $\frac{1}{2}\,\hat{\underset{\sim}{C}}^e$ as the conductivity matrix leads to the following stability condition:

$$\begin{cases} \text{For} \quad \alpha \geq \frac{1}{2} & \text{unconditionally stable} \\ \\ \quad 0 \leq \alpha < \frac{1}{2} & \text{conditionally stable} \end{cases} \tag{3.2.21}$$

So the two-pass element-by-element possesses the same stability region as the governing α-scheme.

3.2.6 Two-Pass Element-By-Element: Accuracy Analysis

Expanding the two-pass operator approximations in Taylor series yield:

$$\tilde{\underset{\sim}{B}}(\tfrac{1}{2}, \Delta\tau) = \prod_{e=n_{e\ell}}^{1} \hat{\underset{\sim}{B}}^e(\tfrac{1}{2}, \Delta\tau/2) \prod_{e=1}^{n_{e\ell}} \hat{\underset{\sim}{B}}^e(\tfrac{1}{2}, \Delta\tau/2) = \underset{\sim}{1} - \frac{\Delta\tau}{2}\,\hat{\underset{\sim}{C}} +$$

$$+ \; O(\Delta\tau^2) = \hat{\underset{\sim}{B}}(\tfrac{1}{2}, \Delta\tau) + O(\Delta\tau^2) \tag{3.2.22}$$

$$\tilde{\underset{\sim}{A}}(\tfrac{1}{2}, \Delta\tau) = \prod_{e=n_{e\ell}}^{1} \hat{\underset{\sim}{A}}^e(\tfrac{1}{2}, \Delta\tau/2) \prod_{e=1}^{n_{e\ell}} \hat{\underset{\sim}{A}}^e(\tfrac{1}{2}, \Delta\tau/2) = \Big[\underset{\sim}{1} - \frac{\Delta\tau}{2}\,\hat{\underset{\sim}{C}} +$$

$$+ \frac{\Delta\tau^2}{8}\,\hat{\underset{\sim}{C}}^2 + \frac{\Delta\tau^2}{8} \sum_{e=n_{e\ell}}^{1} \sum_{f=e-1}^{1} (\hat{\underset{\sim}{C}}^e\,\hat{\underset{\sim}{C}}^f - \hat{\underset{\sim}{C}}^f\,\hat{\underset{\sim}{C}}^e) + O(\Delta\tau^3)\Big] \cdot$$

$$\cdot \Big[\underset{\sim}{1} - \frac{\Delta\tau}{2}\,\hat{\underset{\sim}{C}} + \frac{\Delta\tau^2}{8}\,\hat{\underset{\sim}{C}}^2 + \frac{\Delta\tau^2}{8} \sum_{e=1}^{n_{e\ell}} \sum_{f=e+1}^{n_{e\ell}} (\hat{\underset{\sim}{C}}^e\,\hat{\underset{\sim}{C}}^f - \hat{\underset{\sim}{C}}^f\,\hat{\underset{\sim}{C}}^e) + O(\Delta\tau^3)\Big] =$$

$$= \underset{\sim}{1} - \Delta\tau\,\hat{\underset{\sim}{C}} + \frac{\Delta\tau^2}{2}\,\hat{\underset{\sim}{C}}^2 + O(\Delta\tau^3) \tag{3.2.23}$$

so:

$$\tilde{\underset{\sim}{A}}(\tfrac{1}{2},\ \Delta\tau) = \hat{\underset{\sim}{A}}(\tfrac{1}{2},\ \Delta\tau) + O(\Delta\tau^3) \tag{3.2.24}$$

$$\tilde{\underset{\sim}{B}}(\tfrac{1}{2},\ \Delta\tau) = \hat{\underset{\sim}{B}}(\tfrac{1}{2},\ \Delta\tau) + O(\Delta\tau^2) \tag{3.2.25}$$

Thus the two-pass element-by-element algorithm is second order accurate when employed together with trapezoidal rule and first order accurate otherwise.

For implementation of element-by-element algorithms in transient heat conduction analysis and generalization of the theory for the nonlinear case see [H11].

3.2.7 Implementation of Element-By-Element Algorithms

We shall assume that the solution at the n^{th} step (i.e. $\underset{\sim}{x}_n$) is known and for advancing the solution to the next step either equation (3.2.7) or (3.2.20) is to be solved.

The following algorithmic flowchart shows the solution procedure:

1. $\underset{\sim}{x}_{n+1} \leftarrow \underset{\sim}{M}^{\frac{1}{2}}\ \underset{\sim}{x}_n$

   $\underset{\sim}{F}_{n+\alpha} \leftarrow \Delta\tau\ \underset{\sim}{M}^{-\frac{1}{2}}\ \underset{\sim}{F}_{n+\alpha}$

2. Loop on number of passes 1, N_{pass} (1 or 2)

3. Loop on element $\begin{cases} e = 1,\ n_{el} & 1^{st} \text{ pass} \\ e = n_{el},\ 1 & 2^{nd} \text{ pass} \end{cases}$

   Localize: $\underset{\sim}{x}_{n+1} \rightarrow \underset{\sim}{x}^e$

   $\underset{\sim}{F}_{n+\alpha} \rightarrow \underset{\sim}{F}^e$

   Get $\hat{\underset{\sim}{A}}^e(\alpha,\ \Delta\tau/N_{pass})$, $\hat{\underset{\sim}{B}}^e(\alpha,\ \Delta\tau/N_{pass})$

   Solve: $\underset{\sim}{x}^e \leftarrow \hat{\underset{\sim}{A}}^e\ \underset{\sim}{x}^e$

   $\underset{\sim}{F}^e \leftarrow \hat{\underset{\sim}{B}}^e\ \underset{\sim}{F}^e$

   Globalize: $\underset{\sim}{x}_{n+1} \leftarrow \underset{\sim}{x}^e$

   $\underset{\sim}{F}_{n+\alpha} \leftarrow \underset{\sim}{F}^e$

4. $\underset{\sim}{x}_{n+1} \leftarrow \underset{\sim}{M}^{-\frac{1}{2}} (\underset{\sim}{x}_{n+1} + \underset{\sim}{F}_{n+\alpha})$

The localizing step in 3. means that entries in global d.o.f. are mapped to local d.o.f. (i.e. element d.o.f.) and the globalizing step in 3 is the reverse process.

Note that the solution algorithm does not involve any global matrices. The connectivity among the elements is introduced only through the localization and globalization of the solution vectors.

3.2.8 Steady-State Solution of First-Order Parabolic Equations

As shown in section 3.1 the solution of the linear static problem can be found as the steady-state solution of a first order parabolic equation.

In seeking for an asymptotic solution we want the solution algorithm to possess the following properties:

i) algorithmic dissipation: this causes the transient response to decay quickly to the steady-state.

ii) homogeneous convergence: a narrow spectrum of eigenvalues of the regularized problem.

iii) stability: a large stable $\Delta\tau$ step will allow the solution to reach steady-state rapidly.

In view of properties (i) and (iii) we should select an integration algorithm unconditionally stable with the maximum possible numerical dissipation. The obvious candidate, among the generalized-α family, is the Backward-Euler method, $\alpha = 1$.

In the next sections we shall formulate the element-by-element algorithms within the Backward-Euler framework, discuss the choice of the regularization mass matrix and introduce a numerical criterion for maximum allowable $\Delta\tau$.

3.2.9 The Backward-Euler Method

For $\alpha = 1$, equation (3.1.9) becomes:

$$(\underset{\sim}{M} + \Delta\tau\ \underset{\sim}{A})\underset{\sim}{x}_{n+1} = \underset{\sim}{M}\ \underset{\sim}{x}_n + \Delta\tau\ \underset{\sim}{b} \tag{3.2.26}$$

which can be written in residual form as:

$$(\underset{\sim}{1} + \Delta\tau\ \hat{\underset{\sim}{C}})\underset{\sim}{M}^{\frac{1}{2}}\ \Delta\underset{\sim}{x} = \Delta\tau\ \underset{\sim}{M}^{-\frac{1}{2}}\ \underset{\sim}{R}_n \tag{3.2.27}$$

$$\underset{\sim}{x}_{n+1} = \underset{\sim}{x}_n + \Delta\underset{\sim}{x} \tag{3.2.28}$$

where:

$$\underset{\sim}{R}_n = \underset{\sim}{b} - \underset{\sim}{A}\ \underset{\sim}{x}_n \tag{3.2.29}$$

The one-pass element-by-element algorithm emanating from the above equation is:

$$\underset{\sim}{M}^{\frac{1}{2}}\ \Delta\underset{\sim}{x} = \Delta\tau \prod_{e=1}^{n_{el}} \hat{\underset{\sim}{A}}_e(1, \Delta\tau)\underset{\sim}{M}^{-\frac{1}{2}}\ \underset{\sim}{R}_n \tag{3.2.30}$$

$$\underset{\sim}{x}_{n+1} = \underset{\sim}{x}_n + \Delta\underset{\sim}{x} \tag{3.2.31}$$

where:

$$\hat{\underset{\sim}{A}}_e(1, \Delta\tau) = (\underset{\sim}{1} + \Delta\tau\ \hat{\underset{\sim}{C}}^e)^{-1} \tag{3.2.32}$$

To modify the above algorithm for the two-pass form replace the π product in (3.2.30) by double π product and $\hat{\underset{\sim}{A}}^e(1, \Delta\tau)$ by $\hat{\underset{\sim}{A}}^e(1, \Delta\tau/2)$. Note that the governing algorithm (Backward-Euler) is only first order accurate in $\Delta\tau$ and such are both its one-pass and two-pass element-by-element analogs.

However, our experience with the method indicates that the constant term in the local truncation error, is much smaller for the two-pass algorithm. The reader should note that the residual formulation presented in equations (3.2.27) - (3.2.29) differs from the algorithms presented in sections 3.2.1 and 3.2.3 and [H11] which are unconditionally stable algorithms. For the residual formulation of the element-by-element algorithm unconditional stability cannot be guaranteed without further embellishment. This is dealt with shortly.

3.2.10 The Regularization Mass Matrix

As stated in section 3.2.7, we would like the spectrum of participating modes to be as narrow as possible, in other words we would like to minimize the ratio:

$$\frac{\lambda_{max}}{\lambda_{min}} \rightarrow 1 \tag{3.2.33}$$

where λ_{max} and λ_{min} are the maximum and minimum eigenvalues of:

$$\det|\underset{\sim}{A} - \lambda \underset{\sim}{M}| = 0 \tag{3.2.34}$$

Remark: Since $\underset{\sim}{A}$ is symmetric positive definite and $\underset{\sim}{M}$ diagonal positive definite all λ_i , i = 1 , 2 ,..., n_{eq}, are positive real quantities.

There are several methods in the engineering literature for treating this problem. The first one, which is commonly used in dynamic relaxation schemes, is to choose the element lumped mass matrices in such a way that the transit time for information transfer for d.o.f. i to adjacent and similar degrees of freedom is constant in all the elements which, in turn, leads to a Courant-Friedricks-Lewy [C1] type condition.

This method was first proposed by Welsh [W2] and implemented for finite-element analysis by Key [K1]. Explicit expressions for several structural type

elements, such as beams, plates and shells, were derived by Cassell [C2, C3] for finite difference schemes. However, the first approach is very cumbersome for structures containing several types of elements [see U1]. Another method is based on application of Gerschgorin's theorem [S1]. This theorem states that: "Every eigenvalue of $\hat{\underset{\sim}{C}}$ lies in at least one of the circles $\mathscr{C}_1$, $\mathscr{C}_2$, ..., $\mathscr{C}_{n_{eq}}$, where $\mathscr{C}_i$ is centered at the diagonal entry $\hat{C}_{ii}$ and has radius $r_i = \sum_{j \neq i}^{n_{eq}} |\hat{C}_{ij}|$ ". The method suggested by Cassell [C2] consists of scaling all the rows of $\hat{\underset{\sim}{C}}$ such that the absolute sum along every row is identical and then estimating the spectum of eigenvalues using the Gerschgorin theorem.

The third approach was proposed by Papadakakis [P2]. He assumes

$$M_{ij} = m_i \, \delta_{ij} \; ; \quad m_i = |\hat{C}_{ii}| \quad \text{(no sum)} \tag{3.2.35}$$

The simplicity of the last approach makes it a favorable candidate and this is the method we chose to adopt in this work.

After we select the regularization mass matrix we must calculate the maximum allowable $\Delta\tau$ step, using accuracy considerations, as will be shown in the next section.

3.2.11 Numerical Criterion for Choosing the Maximum Regularized Problem Step Size

Our iterative solution scheme for the regularized problem is as follows:

$$\begin{cases} \underset{\sim}{x}_0 = \underset{\sim}{0} & (3.2.36) \\ \underset{\sim}{M}^{\frac{1}{2}} \Delta\underset{\sim}{x} = \Delta\tau \, \tilde{\underset{\sim}{A}}(1, \Delta\tau) \underset{\sim}{M}^{-\frac{1}{2}} \underset{\sim}{R}_n & (3.2.37) \\ \underset{\sim}{x}_{n+1} = \underset{\sim}{x}_n + \Delta\underset{\sim}{x} & (3.2.38) \end{cases}$$

where:

$$\tilde{\underset{\sim}{A}}(1, \Delta\tau) = \prod_{e=1}^{n_{e\ell}} \hat{\underset{\sim}{A}}^{e}(1, \Delta\tau) = \prod_{e=1}^{n_{e\ell}} (1 + \Delta\tau\ \hat{\underset{\sim}{C}}^{e})^{-1} \tag{3.2.39}$$

$$\underset{\sim}{R}_n = \underset{\sim}{b} - \underset{\sim}{A}\ \underset{\sim}{x}_n \tag{3.2.40}$$

To account for the two-pass algorithm proceed as in section 3.2.8. A convergence criterion for the algorithm is satisfied when $\Delta\underset{\sim}{x} \longrightarrow \underset{\sim}{0}$. This can occur in one of two cases:

a) $\underset{\sim}{R}_n \longrightarrow \underset{\sim}{0} \iff \underset{\sim}{x}_n \longrightarrow \underset{\sim}{A}^{-1}\underset{\sim}{b}$

b) $\tilde{\underset{\sim}{A}}(1, \Delta\tau)$ is an ill-conditioned array, in this case $\tilde{\underset{\sim}{A}}$ possesses one or more eigenvalues, say $\lambda_i(\tilde{\underset{\sim}{A}})$, $i = 1, 2, \ldots, k \leq n_{eq}$, such that:

$$\varepsilon_1 \leq \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \cdots \leq \lambda_k \leq \varepsilon_2$$

where ε_1 and ε_2 are some small positive constants. Let $\underset{\sim}{\Psi}_i$ be an eigenvector corresponding to $\lambda_i(\tilde{\underset{\sim}{A}})$. Then it is possible that the convergence criterion will be satisfied when the vector $\underset{\sim}{M}^{-\frac{1}{2}}\ \underset{\sim}{R}_n$ converges to a linear combination of the low modes:

$$\underset{\sim}{M}^{-\frac{1}{2}}\ \underset{\sim}{R}_n \underset{n \to \infty}{\longrightarrow} \sum_{i=1}^{k} C_i\ \underset{\sim}{\Psi}_i \tag{3.2.41}$$

where C_i are some constants (not all zeroes). In this case:

$$||\underset{\sim}{M}^{\frac{1}{2}}\ \Delta\underset{\sim}{x}|| \underset{n \to \infty}{\longrightarrow} ||\Delta\tau\ \tilde{\underset{\sim}{A}} \sum_{i=1}^{k} C_i\ \underset{\sim}{\Psi}_i|| = O(\Delta\tau\ \varepsilon_2) \tag{3.2.42}$$

Clearly if ε_2 is small enough, the convergence criterion for $\Delta\underset{\sim}{x}$ will be satisfied.

The smallest eigenvalue of $\tilde{A}(1, \frac{\Delta\tau}{N_{pass}})$ can be controlled by the $\Delta\tau$ step.

To see an example of such ill-conditioned behavior consider the linear static test problem shown in figure 3 (case 1). The three degrees of freedom linear structure was subjected to one static loading case. Figure 4. shows a plot of the displacement at node no. 3 vs. the number of iterations for two-$\Delta\tau$ steps. In this case the regularization mass was taken as the diagonal entries of the stiffness matrix and the asymptotic results were virtually exact for $\Delta\tau \leq 1$; however, when we continued to increase $\Delta\tau$, more and more pathological behavior was observed.

When using a CDC6600 computer (60 bit machine) ill-conditioned behavior started when the minimum eigenvalue of the regularized problem was $\leq 10^{-5}$.

We suggest the following empirical criterion for determining the maximum $\Delta\tau$ step as follows:

$$\lambda_{min}(\tilde{A}) \geq \varepsilon \qquad (3.2.43)$$

where ε is machine-dependent constant defined, once-and-for-all, for the machine in hand as follows:

machine word-length (bits)	ε
60 - 64	10^{-4}
32	10^{-2}
16	10^{-1}

Next we shall present a numerical procedure for calculating $\Delta\tau$.

3.2.12 Numerical Procedure for $\Delta\tau$ Calculation

As a first estimation of $\Delta\tau$ we shall use an approximation for the smallest eigenvalue of the exact operator $\hat{\underset{\sim}{A}}$.

Recall:

$$\hat{\underset{\sim}{A}}(1, \Delta\tau) = (\underset{\sim}{1} + \Delta\tau\ \hat{\underset{\sim}{C}})^{-1} \tag{3.2.44}$$

Assume that $\hat{\underset{\sim}{C}}$ is defined using a regularization mass equal to the diagonal of $\underset{\sim}{A}$. Applying Gerschgorin's theorem to $\hat{\underset{\sim}{C}}$ we obtain:

$$\lambda_{max}(\hat{\underset{\sim}{C}}) \leq 1 + r_{max}(\hat{\underset{\sim}{C}}) \tag{3.2.45}$$

where:

$$r_{max}(\hat{\underset{\sim}{C}}) = \max_{i=1}^{n_{eq}} \sum_{\substack{j=1 \\ j\neq i}}^{n_{eq}} |\hat{C}_{ij}| \tag{3.2.46}$$

Thus:

$$\lambda_{min}(\hat{\underset{\sim}{A}}) \geq \frac{1}{1 + \Delta\tau(1 + r_{max}(\hat{\underset{\sim}{C}}))} \tag{3.2.47}$$

Then $\Delta\tau$ can be calculated from:

$$\Delta\tau = \frac{1}{1 + r_{max}}(\hat{\underset{\sim}{C}}) \left[\frac{1}{\varepsilon} - 1\right] \tag{3.2.48}$$

where ε is the machine-dependent tolerance discussed in the previous section.

The algorithm for the above calculation is:

1. $\underset{\sim}{r} \leftarrow \underset{\sim}{0}$

2. loop on elements $e = 1, n_{el}$

   Get $\hat{\underset{\sim}{C}}^e = \underset{\sim}{M}^{-\frac{1}{2}} \underset{\sim}{A}^e \underset{\sim}{M}^{-\frac{1}{2}}$

   loop on elements d.o.f.'s $i = 1, n_{ee}$

   $$r_i^e \leftarrow \sum_{\substack{j=1 \\ j \neq i}}^{n_{ee}} |\hat{C}_{ij}^e|$$

   assemble $\underset{\sim}{r} \leftarrow \underset{\sim}{r} + \underset{\sim}{r}^e$

3. $$r_{max} \leftarrow \max_i^{n_{eq}} r_i$$

4. $\Delta\tau \leftarrow (1/\varepsilon - 1)/(1 + r_{max})$

The next step is to check whether the smallest eigenvalue of the element-by-element approximate operator $\tilde{\underset{\sim}{A}}(1, \Delta\tau)$ satisfies:

$$\lambda_{min}(\tilde{\underset{\sim}{A}}) \geq \varepsilon \tag{3.2.49}$$

This can be done using Rayleigh's quotient:

$$\lambda_{min}(\tilde{\underset{\sim}{A}}) \leq \frac{\underset{\sim}{v}^T \tilde{\underset{\sim}{A}} \underset{\sim}{v}}{\underset{\sim}{v}^T \underset{\sim}{v}} \leq \lambda_{max}(\tilde{\underset{\sim}{A}}) \tag{3.2.50}$$

$$\text{for every } \underset{\sim}{v} \in \mathbb{R}^{n_{eq}}$$

In this section we showed how to calculate the $\Delta\tau$ step. Unfortunately, in many cases the limits imposed on the $\Delta\tau$ step dictate many iterations using straightforward N.R. before the steady-state solution is reached.

To overcome this difficulty, we shall introduce, in the last sections of this chapter, some numerical methods which allow us to reach the steady-state solution in only a few iterations regardless of step size.

3.3 Quasi-Newton Updates

The methods that shall be described in the next sections were originally developed for improving the convergence of nonlinear problems of the form:

$$\underset{\sim}{N}(\underset{\sim}{x}) = \underset{\sim}{b} \tag{3.3.1}$$

where $\underset{\sim}{N}(\underset{\sim}{x})$ is a given nonlinear vector valued function of $\underset{\sim}{x}$ and $\underset{\sim}{b}$ is a given constant vector.

Writing (3.3.1) in N.R. format, yields:

$$D\,\underset{\sim}{N}(\underset{\sim}{x}_{(i)})\,\Delta\underset{\sim}{x} = \underset{\sim}{R}_{(i)} \triangleq \underset{\sim}{b} - \underset{\sim}{N}(\underset{\sim}{x}_{(i)}) \tag{3.3.2}$$

$$\underset{\sim}{x}_{(i+1)} = \underset{\sim}{x}_{(i)} + \Delta\underset{\sim}{x} \tag{3.3.3}$$

where $D\,\underset{\sim}{N}(\underset{\sim}{x}_{(i)})$ is the consistent tangent stiffness [H10] and $\underset{\sim}{R}_{(i)}$ the residual load vector (see section 2.3.4 for details).

In the context of the element-by-element solution of the regularized problem the analog of $D\,\underset{\sim}{N}(\underset{\sim}{x}_{(i)})$ is the $\tilde{\underset{\sim}{A}}(1, \Delta\tau)$ operator and the analog of the internal force vector, $\underset{\sim}{N}(\underset{\sim}{x}_{(i)})$, is $\underset{\sim}{A}\,\underset{\sim}{x}_n$.

The basic idea of the following methods is to improve the solution, $\underset{\sim}{x}(I_{iter} + 1)$, using information based on current and historical data (i.e. $\Delta\underset{\sim}{x}$, $\underset{\sim}{x}_{(i)}$ and $\underset{\sim}{R}_{(i)}$, $i = 1, 2, \ldots, I_{iter}$).

3.3.1 Line Searches

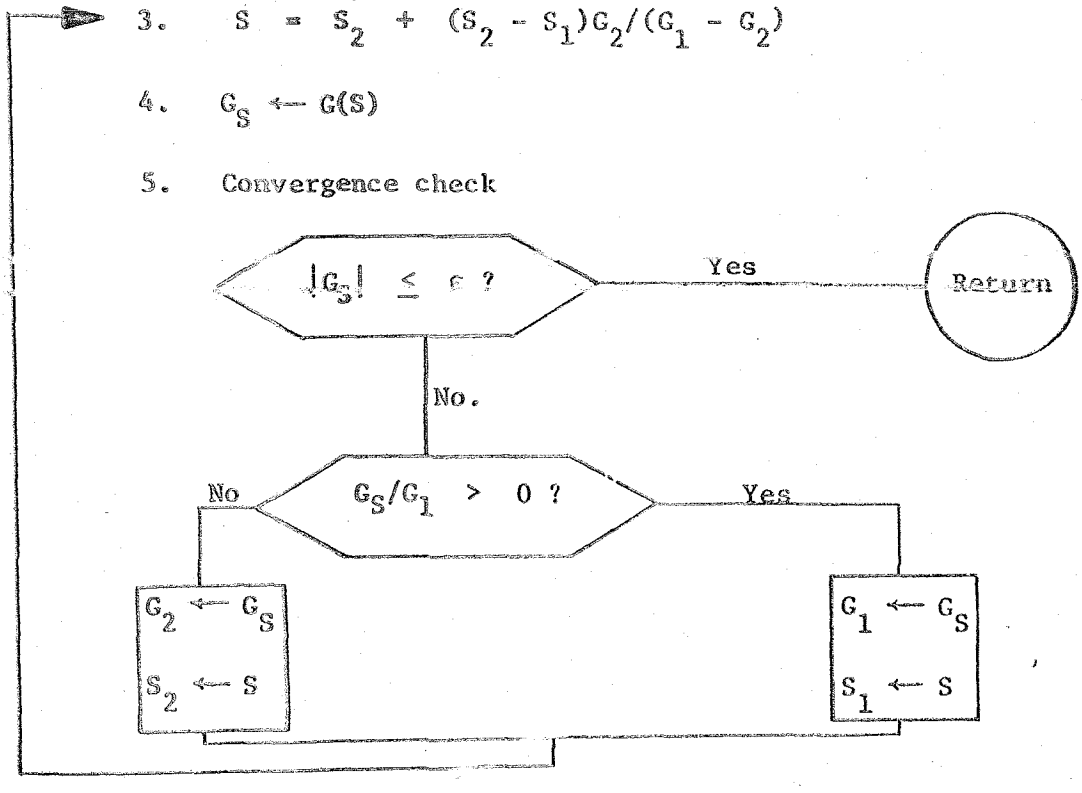
The idea of this method is to scale $\Delta\underset{\sim}{x}$ by a constant S such that the residual load vector will become perpendicular to $\Delta\underset{\sim}{x}$ (i.e. minimizing the out-of-balance force in the $\Delta\underset{\sim}{x}$ direction). For this purpose, a search

function, $G(S)$, is defined as follows:

$$G(S) = \Delta\underset{\sim}{x}^T \cdot (\underset{\sim}{b} - \underset{\sim}{N}(\underset{\sim}{x}_{(i)} + S\,\Delta\underset{\sim}{x})) \qquad (3.3.4)$$

The algorithm for implementing the search is:

1. $S_1 \leftarrow 0$; $S_2 \leftarrow 1$
2. $G_1 \leftarrow G(0)$
   $G_2 \leftarrow G(1)$
3. $S = S_2 + (S_2 - S_1)G_2/(G_1 - G_2)$
4. $G_S \leftarrow G(S)$
5. Convergence check



Note that step 3 is merely a linear interpolation to the zero point. It should be mentioned that some other versions with slight variations of the above algorithm exist. As an example we shall mention the Illinois algorithm [24] which bisects the initial search interval (i.e. the [0,1] interval) before starting the search to get a better closure on the zero point.

In our case, $\underset{\sim}{N}(x)$ is a linear vector function, (i.e $\underset{\sim}{A}\ \underset{\sim}{x}_n$), therefore the line search converges in a single iteration to the zero point. For the linear case, $G(S)$ takes the following form:

$$G(S) = \Delta\underset{\sim}{x}^T \cdot [\underset{\sim}{b} - \underset{\sim}{A}(\underset{\sim}{x}_n + S\ \Delta\underset{\sim}{x})] \tag{3.3.5}$$

or:

$$G(S) = \Delta\underset{\sim}{x}^T\ \underset{\sim}{R}_n - S\ \Delta\underset{\sim}{x}^T\ \underset{\sim}{A}\ \Delta\underset{\sim}{x} \tag{3.3.6}$$

requiring $G(S) = 0$ leads to:

$$S = \frac{\Delta\underset{\sim}{x}^T\ \underset{\sim}{R}_n}{\Delta\underset{\sim}{x}^T\ \underset{\sim}{A}\ \Delta\underset{\sim}{x}} \tag{3.3.7}$$

One should notice that the search parameter S can be viewed as minimum value of the total potential function $P(S)$ in $\Delta\underset{\sim}{x}$ direction namely:

$$\underset{\sim}{x}_{n+1} = \underset{\sim}{x}_n + S\ \Delta\underset{\sim}{x} \tag{3.3.8}$$

$$P(S) = \underset{\sim}{x}_{n+1}^T\ (\underset{\sim}{b} - \frac{1}{2}\ \underset{\sim}{A}\ \underset{\sim}{x}_{n+1}) \tag{3.3.9}$$

Minimizing the total potential with respect to S leads to the same result given by (3.3.7). Thus for $\underset{\sim}{A}$ symmetric positive definite the stability of the overall algorithm is not compromised if we employ a line search in each solution iteration. The one directional line search minimizes the total potential energy in a one dimensional subspace spanned by $\Delta\underset{\sim}{x}$. However, improved results can be obtained if we minimize the potential energy in a higher order subspace. Next we shall present a two directional search which we found to be most successful when implemented with the element-by-element algorithms.

3.3.2 Two-Directional Search

Our experiments with the element-by-element algorithm indicate a very good low mode behavior and a lack of high mode accuracy. Thus we would like to introduce the high mode accuracy by introducing a second search direction which is accurate for high modes. Fortunately such a direction is readily available without need of any additional computation. We refer to the Jacobi iterative technique [Z1], a method which is very accurate in high modes. In fact, one can show that the amplification factor of the Jacobi method at the n^{th} iteration is given by:

$$\left(1 - \frac{\lambda(\underset{\sim}{A})}{\lambda_{max}(\underset{\sim}{A})}\right)^n \tag{3.3.10}$$

Clearly the amplification is of order ε^n in the high modes and $(1 - \varepsilon)^n$ for the low modes where $\varepsilon << 1$.

The Jacobi method can be employed with the preceding theory by simply replacing the left hand side operator of equation (3.2.27) (namely $\frac{1}{\Delta\tau}\ \underset{\sim}{M}^{\frac{1}{2}}(\underset{\sim}{1} + \Delta\tau\ \underset{\sim}{C})\underset{\sim}{M}^{\frac{1}{2}}$) by an identity matrix. Thus we propose to use $\Delta\underset{\sim}{x}$ and $\underset{\sim}{R}_n$ as search directions and to minimize the total potential function in these directions:

$$\underset{\sim}{x}_{n+1} = \underset{\sim}{x}_n + S_1\ \Delta\underset{\sim}{x} + S_2\ \underset{\sim}{R}_n \tag{3.3.11}$$

$$P(S_1, S_2) = \underset{\sim}{x}^T_{n+1}(\underset{\sim}{b} - \frac{1}{2}\ \underset{\sim}{A}\ \underset{\sim}{x}_{n+1}) \tag{3.3.12}$$

$$\frac{\partial P}{\partial S_\alpha} = 0 \qquad \alpha = 1, 2 \tag{3.3.13}$$

which yield:

$$\begin{bmatrix} (\Delta\underset{\sim}{x}^T \underset{\sim}{A}\ \Delta\underset{\sim}{x}) & (\underset{\sim n}{R}^T \underset{\sim}{A}\ \Delta\underset{\sim}{x}) \\ (\underset{\sim n}{R}^T \underset{\sim}{A}\ \Delta\underset{\sim}{x}) & (\underset{\sim n}{R}^T \underset{\sim}{A}\ \underset{\sim n}{R}) \end{bmatrix} \begin{Bmatrix} S_1 \\ S_2 \end{Bmatrix} = \begin{Bmatrix} (\Delta\underset{\sim}{x}^T\ \underset{\sim}{b}) - (\underset{\sim n}{x}^T\ \Delta\underset{\sim}{x}) \\ (\underset{\sim n}{R}^T\ \underset{\sim}{b}) - (\underset{\sim n}{x}^T\ \underset{\sim n}{R}) \end{Bmatrix} \tag{3.3.14}$$

The line-search methods are very efficient and usually cause a fast convergence for "well behaved" nonlinear problem. However, since the search is done in a direction calculated using an approximation to the tangent matrix, $D\underset{\sim}{N}$, or in the case of element-by-element approximation, $\tilde{\underset{\sim}{A}}(1, \Delta\tau)$, this direction may not point to the neighborhood of the final solution point and several iterations may be required for convergence.

In the next sections we shall introduce two quasi-Newton methods which aim to find a better search direction for the line-search algorithm.

3.3. Broyden Updates

The method proposed by Broyden [D3] is to replace the operator approximation, $\tilde{\underset{\sim}{A}}(1, \Delta\tau)$, by $\underset{\sim}{A}^*(1, \Delta\tau)$ given by:

$$\underset{\sim}{A}^{*-1}(1, \Delta\tau) = (\underset{\sim}{1} + \underset{\sim N}{C}\ \Delta\underset{\sim N}{x}^T) \cdot (\underset{\sim}{1} + \underset{\sim N-1}{C}\ \Delta\underset{\sim}{x}^T_{N-1}) \cdot \ldots$$

$$\ldots \cdot (\underset{\sim}{1} + \underset{\sim 1}{C}\ \Delta\underset{\sim 1}{x}^T) \cdot \tilde{\underset{\sim}{A}}^{-1}(1, \Delta\tau) \tag{3.3.15}$$

where N is the number of updates in each iteration (step) and the $\underset{\sim i}{C}$, $i=1, 2, \ldots, N$, are vectors defined by $\Delta\underset{\sim}{x}$ and previous states as will be shown later.

The computation algorithm is as follows (implemented with the element-by-element formulation):

1. $\Delta \underset{\sim}{x} \leftarrow \Delta\tau \, \underset{\sim}{M}^{-\frac{1}{2}} \, \tilde{\underset{\sim}{A}}(1, \Delta\tau) \underset{\sim}{M}^{-\frac{1}{2}} \, \underset{\sim}{R}_n(\underset{\sim}{x})$

2. Line search to obtain S

3. $\Delta \bar{\underset{\sim}{x}}_0 \leftarrow \Delta\tau \, M^{-\frac{1}{2}} \, \tilde{\underset{\sim}{A}}(1, \Delta\tau) \underset{\sim}{M}^{-\frac{1}{2}} \, \underset{\sim}{R}_n(\underset{\sim}{x}_n + S\Delta \underset{\sim}{x})$

4. Loop on no. of updates $k = 1, 2, \ldots, N-1$
   $\Delta \bar{\underset{\sim}{x}}_k \leftarrow \Delta \bar{\underset{\sim}{x}}_{k-1} + (\Delta \underset{\sim}{x} \cdot \Delta \bar{\underset{\sim}{x}}_{k-1}) \cdot \underset{\sim}{C}_k$

5. $\underset{\sim}{C}_N \leftarrow [(1 - S)\Delta \underset{\sim}{x} - \Delta \bar{\underset{\sim}{x}}_{N-1}] \, / \, [\Delta \underset{\sim}{x} \cdot (\Delta \bar{\underset{\sim}{x}}_{N-1} - \Delta \underset{\sim}{x})]$

6. $\Delta \underset{\sim}{x}_N \leftarrow \Delta \bar{\underset{\sim}{x}}_{N-1} + (\Delta \underset{\sim}{x} \cdot \Delta \bar{\underset{\sim}{x}}_{N-1}) \underset{\sim}{C}_N$

7. Store $\{\Delta \underset{\sim}{x}_N, \underset{\sim}{C}_N\}$

8. Line search in $\Delta \underset{\sim}{x}_N$ direction

9. $\Delta \underset{\sim}{x} \leftarrow S\Delta \underset{\sim}{x}_N$

Remarks:

1. The pair of vectors $\{\underset{\sim}{C}_k, \Delta \underset{\sim}{x}_k\}$ may be stored on a secondary storage device.
2. Only a fixed number of updates are performed in each iteration and the earliest data are discarded as new data are obtained

The Broyden method being a rank one update, yields a nonsymmetric approximation $\underset{\sim}{A}^*(1, \Delta\tau)$ even in cases where $\tilde{\underset{\sim}{A}}(1, \Delta\tau)$ is symmetric (e.g. the two-pass element-by-element).

Next we shall introduce a rank two update, the BFGS method, which produces a symmetric approximation whenever $\tilde{\underset{\sim}{A}}(1, \Delta\tau)$ is symmetric.

3.3.4 BFGS Updates

The method suggested by Broyden-Fletcher-Goldfarb-Shanno [M1] is based on the following rank-two update:

$$\underset{\sim}{\tilde{A}}^{*-1}(1, \Delta\tau) = (\underset{\sim}{1} + \underset{\sim N}{V} \underset{\sim N}{W}^T)(\underset{\sim}{1} + \underset{\sim N-1}{V} \underset{\sim N-1}{W}^T)\ldots.(\underset{\sim}{1} + \underset{\sim 1}{V} \underset{\sim 1}{W}^T) \cdot$$

$$\cdot \underset{\sim}{\tilde{A}}^{-1}(1, \Delta\tau) \cdot (\underset{\sim}{1} + \underset{\sim 1}{W} \underset{\sim 1}{V}^T)\ldots.(\underset{\sim}{1} + \underset{\sim N}{W} \underset{\sim N}{V}^T) \qquad (3.3.16)$$

where N is the number of updates and $\underset{\sim k}{V}$, $\underset{\sim k}{W}$ are the update pairs which will be defined later.

For the element-by-element algorithms, the BFGS method is implemented using the following algorithm:

1. $\Delta\underset{\sim}{x} \leftarrow \Delta\tau \, \underset{\sim}{M}^{-\frac{1}{2}} \, \underset{\sim}{\tilde{A}}(1, \Delta\tau) \underset{\sim}{M}^{-\frac{1}{2}} \, \underset{\sim n}{R}(\underset{\sim}{x})$

2. Line search in $\Delta\underset{\sim}{x}$ direction to get S

3. $\underset{\sim N}{V} \leftarrow \dfrac{1}{\Delta\underset{\sim}{x} \cdot \underset{\sim n}{R}} \, \Delta\underset{\sim}{x}$

   $\underset{\sim N}{W} \leftarrow \underset{\sim n}{R}(\underset{\sim}{x} + S\Delta\underset{\sim}{x}) - (1 - S^{\frac{1}{2}})\underset{\sim n}{R}(\underset{\sim}{x})$

   Store $\{\underset{\sim N}{V}, \underset{\sim N}{W}\}$

4. $\underset{\sim N}{\bar{R}} \leftarrow \underset{\sim n}{R}(\underset{\sim}{x} + S\Delta\underset{\sim}{x})$

5. Loop $k = N, N-1, \ldots, 1$

   $\underset{\sim k-1}{\bar{R}} \leftarrow \underset{\sim k}{\bar{R}} + (\underset{\sim k}{V} \cdot \underset{\sim k}{\bar{R}}) \underset{\sim k}{W}$

6. $\Delta\underset{\sim 0}{\bar{x}} \leftarrow \Delta\tau \, \underset{\sim}{M}^{-\frac{1}{2}} \, \underset{\sim}{\tilde{A}}(1, \Delta\tau) \underset{\sim}{M}^{-\frac{1}{2}} \, \underset{\sim 0}{\bar{R}}$

7. Loop $k = 1, N$

8. $\Delta\bar{x}_{k} \leftarrow \bar{x}_{k-1} + (W_{k} \cdot \Delta\bar{x}_{k-1})V_{k}$

9. Line search in $\Delta\bar{x}_{N}$ direction

10. $x \leftarrow x + S\Delta\bar{x}_{N}$

11. If more iterations are required repeat steps 3. - 10.

The two quasi-Newton update methods, Broyden and BFGS, require essentially the same storage and data handling capabilities; however, operationswise, the BFGS update is a more expensive method to use.

In the next section of this chapter we shall present numerical examples using both methods and attempt to draw some conclusions regarding the practical implementation of quasi-Newton updates.

3.3.5 Quasi-Newton Updates: Examples

We solved a test problem, shown in figure 3, as a three d.o.f. linear dynamic problem under one dynamic loading case. The structure shown was solved with two different stiffness and nodal mass configurations denoted "case 2" and "case 3", in figure 3.

For case 2 the eigenvalues of the associated dynamic problem (i.e. det $(K - \omega^2 M) = 0$) were:

$$\begin{cases} \omega_1 = .352 \\ \omega_2 = 2.378 \\ \omega_3 = 5.180 \end{cases}$$

and the central difference critical time step was $\Delta t_{crit} = .386$.

The problem was solved using time step $\Delta t = 1.$ (about 3 times Δt_{crit}) and for each time step the regularized problem was formed and solved using only one iteration and a single quasi-Newton update. The results were compared to those of an implicit global trapezoidal algorithm. Figures 5a and 5b show the displacement time history for node 3 obtained using the one-pass element-by-element algorithm together with Broyden and BFGS updates. Figures 5c and 5d show the results obtained using the two-pass element-by-element algorithm with the same updates.

The results obtained show excellent convergence in all cases (only a single update was needed) for this problem which indicates that whenever the participating modes are in a narrow band (only one order of magnitude apart in this case) the one-pass element-by-element together with the Broyden update is the economical method.

For case 3 the eigenvalues were:

$$\begin{cases} \omega_1 = .0576 \\ \omega_2 = 12.26 \\ \omega_3 = 141.60 \end{cases}$$

Note that in this problem the eigenvalues have four orders of magnitude spread. The explicit critical time step was $\Delta t_{crit} = .0141$.

Here, again, we solved the problem using $\Delta t = 1.$ (70 times Δt_{crit}) and used the two-pass element-by-element algorithm together with 3 quasi-Newton updates per step.

Figures 6a and 6b show the displacement time history for node 3 obtained using Broyden and BFGS updates compared with the results of the governing implicit algorithm (the trapezoidal rule).

The results obtained using the Broyden updates did not converge after three updates; however, three BFGS updates were virtually exact.

The conclusion is to adopt the BFGS method as the updating procedure and to use the two-pass element-by-element algorithm whenever we expect a wide band (over 4 orders of magnitude) of participating modes.

The quasi-Newton updates cannot be implemented in conjunction with the two directional search technique introduced in section 3.3.2. We shall present a new class of updates, which we shall refer to as complementary updates, enabling us to use the new search technique.

3.3.6 Complementary Updates

The quasi-Newton updates above satisfy the following conditions:

$$\text{i)} \quad \underset{\sim}{A}^{*-1}\Delta\underset{\sim}{x} = \Delta\underset{\sim}{R}$$

$$\text{ii)} \quad \underset{\sim}{A}^{*}\,\underset{\sim}{z} = \tilde{\underset{\sim}{A}}\,\underset{\sim}{z} \quad \text{for every} \quad \underset{\sim}{z} \perp \Delta\underset{\sim}{x}$$

$$\text{iii)} \quad \underset{\sim}{A}^{*} = \tilde{\underset{\sim}{A}} \quad \text{if} \quad \underset{\sim}{R} = \underset{\sim}{0}$$

Condition (i) is the secant equation condition, (ii) implies that the previous approximation will not be changed in any direction but the updating direction $\Delta\underset{\sim}{x}$ and the last condition implies convergence of the operator approximation as the solution converges.

The quasi-Newton updates use the solution increment, $\Delta\underset{\sim}{x}$, to update the operator. We shall use the complementary part of the solution vector, namely $\underset{\sim}{x}_n$, to update the operator and replace condition (ii) by:

$$\text{ii)} \quad \underset{\sim}{A}^{*-1}\,\underset{\sim}{x}_n = \hat{\underset{\sim}{A}}^{-1}\,\underset{\sim}{x}_n$$

where $\hat{\underset{\sim}{A}}$ is the exact operator. The following are rank one and rank two updates

emanating from the above conditions:

Rank one update:

$$\begin{cases} \underset{\sim}{A}^{*-1} = \tilde{\underset{\sim}{A}}^{-1}(\underset{\sim}{1} + \underset{\sim}{V}\,\underset{\sim}{W}^T) \\ \underset{\sim}{W} = -\dfrac{1}{||\underset{\sim}{x}_n||^2}\,\underset{\sim}{x}_n \\ \underset{\sim}{V} = \underset{\sim}{x}_n - \tilde{\underset{\sim}{A}}\,\hat{\underset{\sim}{A}}^{-1}\,\underset{\sim}{x}_n \end{cases}$$

Rank two update:

$$\begin{cases} \underset{\sim}{A}^{*-1} = (\underset{\sim}{1} + \underset{\sim}{V}\,\underset{\sim}{W}^T)\,\tilde{\underset{\sim}{A}}^{-1}\,(\underset{\sim}{1} + \underset{\sim}{W}\,\underset{\sim}{V}^T) \\ \underset{\sim}{W} = \underset{\sim}{x}_n \\ \underset{\sim}{V} = -\dfrac{1}{\alpha(\underset{\sim}{x}_n^T\,\underset{\sim}{b})}\,[\alpha\,\underset{\sim}{b} - \hat{\underset{\sim}{A}}^{-1}\,\underset{\sim}{x}_n] \\ \alpha = \left(\dfrac{\underset{\sim}{x}_n^T\,\hat{\underset{\sim}{A}}^{-1}\,\underset{\sim}{x}_n}{\underset{\sim}{x}_n^T\,\underset{\sim}{b}}\right)^{1/2} \end{cases}$$

Clearly the implementation of the above updates is identical to the implementation of the corresponding rank one and rank two quasi-Newton updates described in sections 3.3.3 and 3.3.4. An important advantage of the complementary updates is that there is only one update per iteration unlike the quasi-Newton updates which accumulate.

$\underset{\sim}{x}_n$ may be looked upon as any given vector and may include the two directional line search as part of its construction.

In the numerical implementation and examples chapter of this work a comparison of the results obtained using the above methods to those of the classical quasi-Newton techniques is presented in conjunction with element-by-element algorithms and the Jacobi algorithm.

3.4 Substructures

In some practical problems, parts of the mesh can be treated as "superelements" having internal nodes and boundary nodes common with neighboring superelements.

In general, such substructures aim to cut down the computational cost of large scale problems. The idea can be viewed as a physical method for mathematical matrix partitioning, where each superelement is a submatrix of the global problem having a small band width.

In regular finite element formulations, each substructure is solved separately and the global solution is obtained by solving compatibility equations along the boundaries of the substructures. A very similar idea can be implemented with the element-by-element algorithms.

3.4.1 Superelement-by-Superelement Algorithms

Assume that the structure is divided into N_{super} superelements. Define:

$$\hat{\underset{\sim}{C}}^{S} = \mathop{\mathbf{A}}_{e=1}^{N_{els}} \hat{\underset{\sim}{C}}^{e}, \quad S = 1, 2, \ldots, N_{super} \tag{3.4.1}$$

where N_{els} is the number of elements in the S^{th} superelement and $\mathbf{A}$ is the regular finite element assembly operator.

Let

$$\hat{\underset{\sim}{A}}^{S}(1, \Delta\tau) = (\underset{\sim}{1} + \Delta\tau\, \hat{\underset{\sim}{C}}^{S})^{-1} \tag{3.4.2}$$

then the one-pass form of the algorithm becomes:

$$\tilde{\underset{\sim}{A}}(1, \Delta\tau) = \mathop{\pi}_{S=1}^{N_{super}} \hat{\underset{\sim}{A}}^{S}(1, \Delta\tau) \tag{3.4.3}$$

and the two-pass form:

$$\tilde{\underset{\sim}{A}}(1, \Delta\tau) = \prod_{S=N_{super}}^{1} \hat{\underset{\sim}{A}}^{S}(1, \Delta\tau/2) \prod_{S=1}^{N_{super}} \hat{\underset{\sim}{A}}^{S}(1, \Delta\tau/2) \qquad (3.4.4)$$

Since the symmetry and positive semi-definiteness of $\hat{\underset{\sim}{C}}^{e}$ lead to the same properties in $\hat{\underset{\sim}{C}}^{S}$, $\hat{\underset{\sim}{A}}^{S}$ is symmetric positive definite. The stability and accuracy results for the element-by-element algorithms hold for the super-element-by-superelement algorithms as well.

Note that for $N_{super} = N_{e\ell}$ this algorithm is reduced to the element-by-element algorithm, and for $N_{super} = 1$ a global solution scheme is obtained.

CHAPTER IV

IMPLEMENTATION AND NUMERICAL EXAMPLES

In this chapter, the structure of a finite element code employing element-by-element algorithms is described and numerical examples are used for comparing the techniques described in chapter III to other iterative methods. The results are evaluated and guide lines are drawn for choosing the most suitable technique for solving a given problem.

4.1 Computer Program

As was mentioned in the introduction to this work, the proposed element-by-element algorithms offer a very attractive data structure for implementation with the new generation of small, inexpensive, mini/micro computers.

The first step in writing a computer code is to select a computer language. A variety of new high level structured languages were introduced in the last years, such as "C", "RATFOR" and "ALGOL 68", however the existence of extensive software written in the traditional "FORTRAN IV" and the lack of compilers, especially for small machines, makes the usage of these languages difficult when one wishes to produce a machine independent code.

The only language supported by the majority of the new generation of computers and suitable for engineering code writing is "FORTRAN 77". Although "FORTRAN 77" still suffers from many of the deficiencies of "FORTRAN IV" it offers the capability of structured programming which simplifies the writing (and reading...) of complex codes.

Our next step was to segregate the code into three separate codes which interact and communicate with each other through disk files. The motivations for doing this are several. Firstly, one should realize that a complex finite element object code requires a significant in-core storage space which reduces memory space available for the data itself. Secondarily, there are different and distinct tasks in a finite element analysis program which are performed sequentially and allow "natural" segregation of the code simplifying the writing and debugging stages.

Our finite element system consists of the following codes:

1. CINPUT - data generator
2. CONTINUUM - finite element analyzer
3. CPLOT - graphic post-processor

A communication chart for the three codes is included in appendix I.

Next we shall describe each module and explain its tasks in the finite element system.

4.1.1 CINPUT - Data Generator

Preparing input data files for the finite element analysis of a large scale problem is a very tedious and time consuming task. For this reason a data generation code usually accompanies a finite element analyzer and is considered an integral part of the system. The idea is to minimize the amount of data supplied by the user and generate the majority of data required by the analyzer within the computer itself.

CINPUT includes 1-D, 2-D and 3-D linear and quadratic mesh generators which are powerful enough to handle the complex geometries arising in common engineering problems.

The communication with the user is through labeled unformatted input files which utilize the flexibility of unformatted data together with the new "NAMELIST"* option added to the "FORTRAN 77" compilers to produce a readable unformatted input file.

A new idea implemented in CINPUT is a global function library. The global function library is a collection of user defined functions which allows the generation of data profiles in temporal and spatial domains. As an example, assume that one wishes to employ a prescribed displacement field, $\underset{\sim}{g}(\underset{\sim}{x}, t)$, which admits the following decomposition:

$$\underset{\sim}{g}_i(\underset{\sim}{x}, t) = X_i(x)\, Y_i(y)\, Z_i(z)\, T_i(t)$$

where $X_i, Y_i, Z_i, T_i: \mathbb{R} \longrightarrow \mathbb{R}$ are given, user defined, library functions. The data required by CINPUT are the definition of the library functions and a set of integer identification numbers associating the prescribed displacement field, $\underset{\sim}{g}(\underset{\sim}{x}, t)$, with its spatial and temporal interpolation functions.

CINPUT allows the user to define any of the problem fields (initial states, forces, body-force, etc.) and material properties (nonhomogeneous material) as an extrapolated data similar to the above example so that even very complex problems require only modest input data files.

After the problem data are generated CINPUT allocates the memory for the run (dynamic memory allocation routines), creates a restart file for CONTINUUM (a core image dump) and provides the user with a list of generated data and a MESH file for graphic mesh verification.

---

*The "NAMELIST" option was implemented differently on different compilers. Thus we added a "NAMELIST" simulator to CINPUT which makes it a machine independent code and offers some extensions to the original "NAMELIST" option.

4.1.2 CONTINUUM - Finite Element Analyzer

CONTINUUM implements the algorithms presented in chapter III together with linear and nonlinear 2-D (axisymmetric, plane strain and plane stress) and 3-D elasticity. CONTINUUM employs the four-node isoparametric quadrilateral elements for the 2-D cases and the eight-node trilinear brick elements for 3-D problems. The code is capable of solving static and dynamic problems using either a global implicit technique or an element-by-element approach (one or two pass formulation) and has a substructuring option (superelement-by-superelement) built into it.

CONTINUUM is an implicit/explicit [H5, H6] predictor multi-corrector code and implements the following nonlinear theories:

i) finite deformations

ii) finite rotations [H7]

iii) Key-Krieg [K1] plasticity model

An algorithmic flow chart of CONTINUUM is included in appendix I.

For detailed theory derivations consult chapters II and III.

4.1.3 CPLOT - Graphic Post-Processor

Evaluating and understanding the results of a finite element program, for large scale problems, is a task in itself. Associating the enormous data produced by the code to the physical problem in hand is sometimes very difficult, time consuming and one may lose the overall picture and misinterpret the results. For these reasons we included a graphic post-processing code as part of the finite element system. A graphic output allows us to take the output data that need to be evaluated by the user and present them in a way which is easy to understand so that the overall picture is never lost.

CPLOT was written to serve two purposes:

i) produce mesh plots

ii) produce graphic results

The mesh plots are used as a clear visual check for the data generated by CINPUT and allows the user to verify his input data file before submitting an expensive run. The graphic post processing capabilities of CPLOT are: a) production of time history plots of any point in the structure (displacement, velocity or acceleration), b) production of stress contour line plots on the deformed configuration for any stress component.

Samples of CPLOT graphic results are presented in the numerical examples section of this work.

4.2 Numerical Examples

We selected to test the techniques upon beam problems as these problems exhibit a relatively rich response spectrum (from fast wave propogation to slow bending modes) and are difficult to handle with regular iterative techniques. Observe that in all cases the same solution as that of the globally implicit algorithm is attained if convergence is achieved. Throughout a value of $\Delta\tau = 1$ was employed.

4.2.1 Example 1: Cantilever Beam

The geometrical definition of the problem is shown in figure 7a. The upper half of the beam was discretized into a 32 element mesh as shown in figure 7b. The beam was subjected to two loading conditions:

i) prescribed traction load:

$$t_x(16,\ y) = 0\ ; \qquad t_x(0,\ y) = C_1(y - 2)$$

$$t_y(16,\ y) = C_2\ y(2 - y)$$

$$t_y(0,\ y) = -C_2\ y(2 - y)$$

where t_x, t_y are the traction components in x and y directions respectively.

ii) initial displacements in first bending mode [M2]:

$$v(x) = (\sin \beta_1 L - \sinh \beta_1 L)(\sin \beta_1 \bar{x} - \sinh \beta_1 \bar{x}) + (\cos \beta_1 L + \cosh \beta_1 L)(\cos \beta_1 \bar{x} - \cosh \beta_1 \bar{x})$$

$$\beta_1 = 1.875/L \quad ; \quad L = 16$$

$$\bar{x} = x - 16$$

where $v(x)$ is the displacement of the neutral axis of the beam in the y-direction.

The remainder of the initial displacement field was calculated using the classical assumption that straight lines perpendicular to the neutral axis remain so in the deformed configuration.

The problem was solved with two different time steps:

a) $\Delta t = 18.7 \ \Delta t_{crit}$

b) $\Delta t = 187 \ \Delta t_{crit}$

where Δt_{crit} was calculated as the shortest element side divided by the dilatational wave velocity ($\Delta t_{crit} = 1.336 \times 10.^{-5}$).

Tables 2 and 3 compare the number of iterations required per step by the element-by-element algorithm and the Jacobi iterative scheme for loading cases (i) and (ii) respectively. The superiority of the element-by-element algorithm over the Jacobi technique, in this problem, is clearly shown and the important effect of the line searches and updating techniques on the number of iterations required for convergence is demonstrated. Figure 8 compares the displacement histories of the beam end (node 45) for loading case (ii) obtained using an implicit solution to those obtained by the element-by-element algorithm with the BFGS update and then with the complementary update. Figure 9 shows the normalized search parameters (S1 and S2) in the $\Delta \underset{\sim}{x}$ and $\underset{\sim}{R}$ directions for

loading case (i). The second search direction contributes less than 4 percent to the solution in this case since the response is a low mode which is captured by the element-by-element algorithm and not by the Jacobi method (see 3.3.2).

4.2.2 Example 2: Cantilever Beam With Built-in Root

The geometrical definition of this problem is identical to the previous example except for the left hand side boundary conditions of the beam model (see figure 7b) which were replaced by built-in boundary conditions. The loading case employed was a prescribed traction (shear) on the right hand side having the same distribution as in example 1 loading case (i). Table 4 is a comparison between the number of iterations required for convergence by the Jacobi method and the element-by-element algorithms.

The built-in boundary condition introduces smaller time scales to the response (rigidity due to boundary conditions) and a singular point at the corner point on the boundary. However, in this problem, as in the previous one, the element-by-element algorithm performed better than the Jacobi method. Figure 10 compares the stress σ_{xx} contour lines and deformed configurations obtained using implicit solution with those obtained using the element-by-element algorithm. The results were accurate to four significant digits in all the methods tested.

4.2.3 Example 3: Elastic-Perfectly Plastic Uniform Beam

The geometrical definition of this problem is identical to those of examples 1 and 2 except that the entire beam is discretized by a 64 element mesh (the lower part of the beam was added to the mesh of examples 1 and 2). We employed the following boundary conditions:

$$\begin{cases} u_x(0, y, t) = u_y(0, 0, t) = 0 & t \in [0, T],\ y \in [-C, C] \\ u_x(0, C, t) = u_x(0, -C, t) = 0 & t \in [0, T] \\ t_y(L, y, t) = Q\left(1 - \dfrac{y^2}{C^2}\right)\left(2t/T_1\right) & t \in [0, T] \\ & y \in [-C, C] \end{cases}$$

where $Q = 1000$, $T_1 = T = .04$, $C = 2$.

The Key - Krieg plasticity model was used for an elastic-perfectly-plastic material having a uniaxial yield stress of $\sigma_y = 3{,}000$. Small deformations were assumed. The elastic stiffness was used on the left-hand side throughout.

Figures 11 and 12 compare the elastic and plastic stress distributions at $t = .036$. A fully developed plastic hinge is present at the root of the beam in the plastic case. The elastic critical time step of this problem is $\Delta t_{crit} = 1.336 \times 10^{-5}$ and the time step used was:

$$\Delta t = 2.5 \times 10^{-3} = 187\ \Delta t_{crit}$$

We employed the BFGS update method and the complementary update method for both the elastic and plastic solutions. The average number of updates was 4 for both updating methods for the elastic and and the plastic solutions.

4.2.4 Example 4: Elastic-Perfectly-Plastic Beam

The geometric definition of the problem is shown in figure 13a.

The beam was discretized using a 500 element mesh (figure 13b). We employed the following kinematic and stress boundary conditions:

$$\begin{cases} u_x(0, y, t) = 0 & y \in [-4, 4],\ t \in [0, T] \\ u_y(0, 0, t) = 0 & t \in [0, T] \end{cases}$$

$$\begin{cases} t_y(28, y, t) = Q\left(1 - \frac{y^2}{16}\right)\left(\frac{t}{.09}\right) & y \in [-4, 4] \\ & t \in [0, T] \\ Q = 250 \end{cases}$$

The Key-Krieg [K1] plasticity model was employed for an elastic perfectly plastic material behavior having a yield stress $\sigma_y = 1000$.

Figure 14 compares the elastic displacement time history at the tip of the beam to the plastic solution. Figures 15 and 16 show the stress distributions at time t = 0.09 for the elastic and plastic solutions. A fully developed plastic hinge is present at the end of the beam and a secondary plastic hinge has partially developed in the stress concentration zone (plastic regions are shown dashed in figure 16c).

We repeated the solution using element-by-element algorithms with both BFGS and complementary updates. The critical time step for this problem was $\Delta t_{crit} = 5.41 \times 10^{-6}$ (calculated as the smallest element size in the mesh devided by the dilatational wave velocity*).

We employed two time steps for these runs: $\Delta t = 10 \times \Delta t_{crit}$ and $\Delta t = 50 \times \Delta t_{crit}$ and for the first time step both updating methods converged after a single update and for the second time step ($\Delta t = 50\ \Delta t_{crit}$) the BFGS update required 7 updates and the complementary update 6 updates.

A word about generating the contour line for the graphic post-processing. The finite element analyzer, CONTINUUM, calculates the stresses at the integration points. The data are then extrapolated to the nodal points by means of a weighted average of all the integration points in the interior domains of all elements connected to the nodal point. The weights are taken as the inverse

---

*Note that for the majority of elements in this problem (all elements outside the hole region) the critical time step is $3.38 \times 10^{-5} = 6.25\ \Delta t_{crit}$.

of the distance between the integration point and the nodal point.

This type of data smoothing ensures that the values obtained at the nodal points will be bounded by the data calculated at the integration points which is an essential property when plastic stresses are present and ensures continuity of stress contours between element domains. However, this method has some drawbacks. For example, data which are antisymmetricabout the neutral axis of the beam, such as σ_{11}, will result in linear distribution of contour lines in all elements having the center line as part of their boundary even in the case where these elements have a uniform stress distribution (part of a plastic hinge). Data with symmetry with respect to the neutral axis, such as the Von-Mises stress, will not suffer from this type of smoothing.

4.3 Operator Storage and Operations Count

In this section we shall compare the storage requirement and the number of operations required per step by an implicit solver and the element-by-element algorithms.

4.3.1 Operator Storage

The element-by-element algorithms require only a single element operator in core at all times.

The implicit solver requires global operator storage and using optimal node numbering (minimum band-width) and compact column storage for the global operator, the following results are obtained:

Examples 1 & 2: 1,260 words of storage

Example 3: 3,564 words of storage

Example 4: 50,232 words of storage

Thus the storage saved by the element-by-element algorithm, even in these small

size problems is significant.

Note: The element-by-element operator storage remains the same regardless of problem size.

4.3.2 Operations Count

The element-by-element algorithm differs from the implicit solver in two basic operations: The LDU factorization of the operator and the forward reduction and back substitution sweeps required in every step.

The number of operations required for the operator factorization are given by:

$$\text{implicit operator: } {}^{1}/_{3}\, n_{eq} \times b^{2}$$

$$\text{element-by-element operator: } n_{e\ell} \times {}^{1}/_{3}\, n_{ee}^{3}$$

The number of operations required for the forward reduction and back substitution sweeps are given by:

$$\text{implicit : } 2n_{eq} \times b$$

$$\text{element-by-element: } n_{e\ell} \times n_{pass} \times n_{iter} \times (n_{ee}^{2} + n_{ee})$$

where:

n_{eq} - number of equations

b - mean half band-width of the global operator

$n_{e\ell}$ - number of elements

n_{ee} - number of element equations

n_{pass} - number of passes in the algorithm

n_{iter} - number of iterations required for convergence

CHAPTER V

CONCLUSIONS

In this work we presented new element-by-element algorithms for finite element equation systems which arise in the linear and nonlinear statics and dynamics of solids and structures. These new algorithms present a new point of view for the finite element approach in which the element operator is treated as an independent operator and not as a part of some global operator.

We formulated the regularized problem associated with a linearized equation system emanating from the finite element discretization of problems in continuum mechanics and structural analysis. The element-by-element approximate factorization concept was then applied to the regularized problem resulting in algorithms possessing stability and accuracy properties of global implicit schemes and having data structures similar to those of explicit algorithms.

The new element-by-element algorithms take advantage of the fact that the connectivity among elements is a local property (elements are connected and effected directly only by neighboring elements). By solving the equations on the element level and localizing and globalizing only the solution vectors, the need for a costly heavily populated global operator* is circumvented.

This new way of thinking about the finite element method opens the door for a new type of algorithm, based on the product of element operators, and avoids the need for a costly (storage wise) global operator. As was shown in

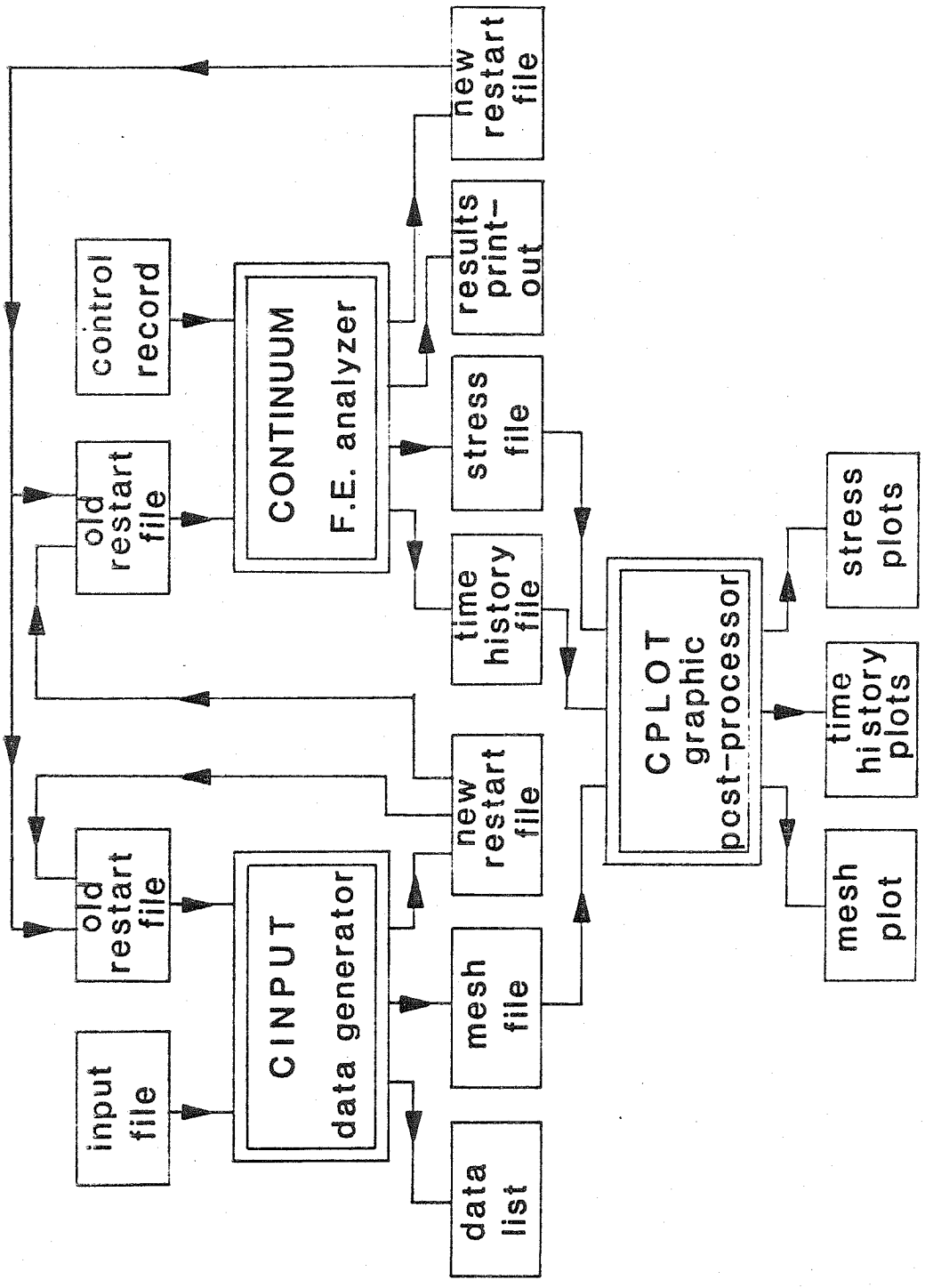
*The global implicit operator is heavily populated since the local connectivity among elements is introduced through the finite element assembly operator employing a mapping of local equation numbers to global ones.

the examples of this work, the element-by-element algorithms maintain the desired properties of their governing implicit algorithms (stability and accuracy) while having a data base similar to that of an explicit algorithm.

Future research on this new generation of element-by-element algorithms will be concentrated on generalizing these ideas to other areas of engineering such as heat conduction, fluid mechanics and eigenvalue analysis.

APPENDIX I

FLOW CHARTS



FLOW CHART I. CONTINUUM System: Comunication Chart

1. INPUT PHASE

read restart file containing:

(i) generated data

(ii) integer arrays ($\underset{\sim}{LM}$, $\underset{\sim}{IEN}$ and $\underset{\sim}{ID}$)

(iii) blank common pointers

clear arrays.

2. CALCULATE AND STORE ELEMENT MATRICES

loop on element groups: $I_{group} = 1, N_{groups}$

calculate shape functions in $\underset{\sim}{\xi}$ domain:

$$\{N_a(\underset{\sim}{\tilde{\xi}}_\ell),\ N_{a,\underset{\sim}{\xi}}(\underset{\sim}{\tilde{\xi}}_\ell)\}_{\ell=1,\ 2,\ldots,\ N_{int}}$$

$$a=1,\ 2,\ldots,\ N_{en}$$

loop on elements within the group: $e=1,\ N_{e\ell}$

call shape functions to obtain

$$\{\underset{\sim}{\tilde{x}}_\ell,\ N_a(\underset{\sim}{\tilde{x}}_\ell),\ N_{a,\underset{\sim}{x}}(\underset{\sim}{\tilde{x}}_\ell)\}_{\ell=1,\ 2,\ldots,\ N_{int}}$$

$$a=1,\ 2,\ldots,\ N_{en}$$

integrate and store elements matrices:

static problem: $\underset{\sim}{k}^e$

dynamic problem: $\underset{\sim}{m}^e,\ \underset{\sim}{c}^e,\ \underset{\sim}{k}^e$

assemble lumped mass vector: $\underset{\sim}{M}_E = \mathop{\mathbf{A}}_{e=1}^{n_{e\ell}} \underset{\sim}{m}_E^e$

3. INITIAL DATA CALCULATION

static problem: $\underset{\sim}{d} \leftarrow \underset{\sim}{0} + \underset{\sim}{g}$

dynamic problem: $\underset{\sim}{d} \leftarrow \underset{\sim}{d}_0 + \underset{\sim}{g}(0)$

$$\underset{\sim}{v} \leftarrow \underset{\sim}{v}_0 + \underset{\sim}{\dot{g}}(0)$$

initial acceleration:

$$\underset{\sim}{a} \leftarrow \underset{\sim}{F}(0)$$

FLOW CHART II. Finite Element Analyzer: Global Driver

loop on elements: e=1, N_{el}

localize: $\underset{\sim}{d} \rightarrow \underset{\sim}{d}^e$, $\underset{\sim}{v} \rightarrow \underset{\sim}{v}^e$

read: $\underset{\sim}{c}^e$, $\underset{\sim}{k}^e$

calculate: $\underset{\sim}{a}^e \leftarrow -\underset{\sim}{c}^e \underset{\sim}{v}^e - \underset{\sim}{k}^e \underset{\sim}{d}^e$

globalize: $\underset{\sim}{a} \leftarrow \underset{\sim}{a} + \underset{\sim}{a}^e$

$$\underset{\sim}{a} \leftarrow \underset{\sim}{M}_E^{-1} \underset{\sim}{a} + \ddot{\underset{\sim}{g}}(0)$$

4. TIME INTEGRATION LOOP: n=1, N_{steps}

(N_{steps} = 1 for static problems)

5. PREDICTOR PHASE

static problem: $\tilde{\underset{\sim}{d}} \leftarrow \underset{\sim}{d}$

dynamic problem: $\tilde{\underset{\sim}{d}} \leftarrow \underset{\sim}{d} + \Delta t \, \underset{\sim}{v} + \frac{\Delta t^2}{2} (1 - 2\beta)\underset{\sim}{a}$

$$\tilde{\underset{\sim}{v}} \leftarrow \underset{\sim}{v} + \Delta t(1 - \gamma)\underset{\sim}{a}$$

6. ITERATION LOOP: i=1, N_{iter}

(N_{iter} = 1 for linear problem)

7. CALCULATE AND STORE ELEMENT ARRAYS

same as step 2. except omit mass calculation

(skip this step if i=1 and n=1 or linear problem)

8. REGULARIZATION MASS CALCULATION

$\underset{\sim}{M}_{reg} \leftarrow \underset{\sim}{0}$

loop on element: e=1, N_{el}

read elements matrices

define $\underset{\sim}{M}^e_{reg}$ through:

A B C

FLOW CHART II (Cont'd.)

static: $M^e_{reg} \leftarrow \text{diag}(k^e)$

dynamic: $M^e_{reg} \leftarrow \text{diag}(m^e + \Delta t\, \gamma\, c^e_I + \Delta t^2 \beta\, k^e_I)$

assemble: $M_{reg} \leftarrow M_{reg} + M^e_{reg}$

9. FORM R.H.S. VECTOR b .

$b \leftarrow F^{(i)}_n$

loop on elements: e=1, N_{el}

localize:

static: $d^{(i)}_n \rightarrow d^e$

dynamic: $d^{(i)}_n \rightarrow d^e$, $v^{(i)}_n \rightarrow v^e$, $a^{(i)}_n \rightarrow a^e$

read element matrices

calculate b^e:

static: $b^e \leftarrow \begin{cases} -k^e d^e & \text{linear} \\ -N^e(d^e) & \text{nonlinear} \end{cases}$

dynamic: $b^e \leftarrow \begin{cases} -m^e a^e - c^e v^e - k^e d^e & \text{linear} \\ -m^e a^e - N^e(d^e, v^e) & \text{nonlinear} \end{cases}$

assemble: $b \leftarrow b + b^e$

10. SOLVE REGULARIZED PROBLEM $A\, x = b$

call ELBYEL

(see flow chart III)

11. CORRECTOR PHASE

static: $d^{(i+1)}_n \leftarrow d^{(i)}_n + x$

FLOW CHART II (Cont'd.)

$$\text{dynamic:} \quad \begin{cases} \underset{\sim}{a}_n^{(i+1)} \leftarrow \underset{\sim}{a}_n^{(i)} + \underset{\sim}{x} \\ \underset{\sim}{v}_n^{(i+1)} \leftarrow \underset{\sim}{v}_n^{(i)} + \gamma\Delta t\, \underset{\sim}{x} \\ \underset{\sim}{d}_n^{(i+1)} \leftarrow \underset{\sim}{d}_n^{(i)} + \beta\Delta t^2\, \underset{\sim}{x} \end{cases}$$

(note $\underset{\sim}{x} \equiv \Delta\underset{\sim}{d}$ in static problem

and $\underset{\sim}{x} \equiv \Delta\underset{\sim}{a}$ in dynamic problem)

12. UPDATE GEOMETRY COORDINATES $\underset{\sim}{X}$

(only for large deformation problem)

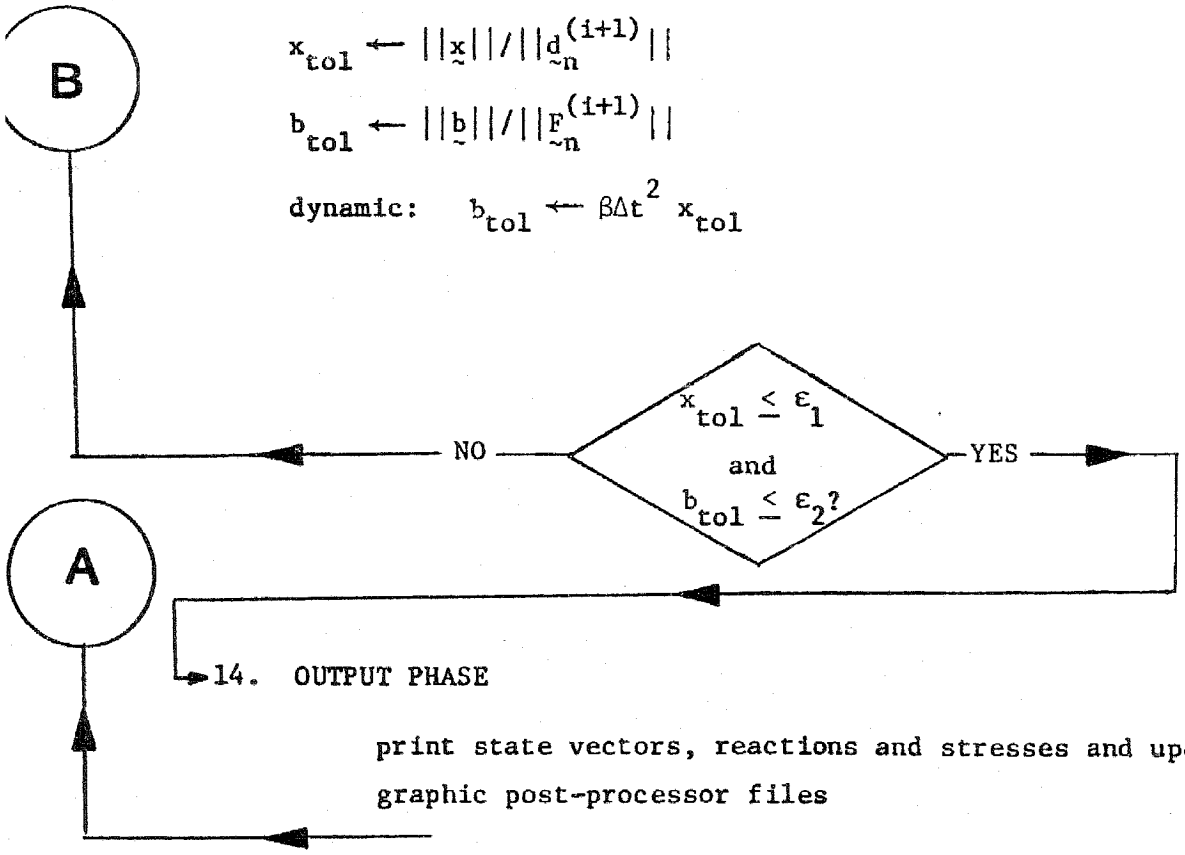
$$\underset{\sim}{X} \leftarrow \underset{\sim}{X} + \underset{\sim}{d}_n^{(i+1)} - \underset{\sim}{d}_n^{(i)}$$

13. CONVERGENCE TEST (nonlinear problem)

$$x_{tol} \leftarrow ||\underset{\sim}{x}||/||\underset{\sim}{d}_n^{(i+1)}||$$

$$b_{tol} \leftarrow ||\underset{\sim}{b}||/||\underset{\sim}{F}_n^{(i+1)}||$$

dynamic: $b_{tol} \leftarrow \beta\Delta t^2\, x_{tol}$



14. OUTPUT PHASE

print state vectors, reactions and stresses and update graphic post-processor files

15. STOP

FLOW CHART II (Cont'd.)

1. INITIALIZATION PHASE

$\underset{\sim}{x} \leftarrow \underset{\sim}{0}$

2. ASSEMBLE AND STORE SUPER-ELEMENT OPERATORS

(this step is executed only for i=1 and n=1 in linear case)

loop on super-elements: $i_{super} = 1, N_{supers}$

$$N_{supers} = \begin{cases} 1 & - \text{ implicit} \\ N_{el} & - \text{ element-by-element} \\ 1 \leq \leq N_{el} & - \text{ substructuring} \end{cases}$$

clear super-element operator

$\underset{\sim}{A}_{i_{super}} \leftarrow \underset{\sim}{0}$

loop on elements: $e=1, N_{el}$

read element matrices

calculate $\underset{\sim}{A}^e$:

static: $\underset{\sim}{A}^e \leftarrow \underset{\sim}{k}^e$

dynamic: $\underset{\sim}{A}^e \leftarrow \underset{\sim}{m}^e + \Delta t \gamma \, \underset{\sim}{c}_I^e + \Delta t^2 \beta \, \underset{\sim}{k}_I^e$

assemble: $\underset{\sim}{A}_{i_{super}} \leftarrow \underset{\sim}{A}_{i_{super}} + \underset{\sim}{A}^e$

decompose: $\underset{\sim}{A}_{i_{super}}^{-1} \leftarrow (1 + \frac{1}{N_{pass}} \underset{\sim}{M}_{reg}^{-\frac{1}{2}} \underset{\sim}{A}_{i_{super}} \underset{\sim}{M}_{reg}^{-\frac{1}{2}})^{-1}$

store $\underset{\sim}{A}_{i_{super}}$; $\underset{\sim}{A}_{i_{super}}^{-1}$ on disk file

3. ITERATION/UPDATE LOOP: $I_{iter} = 1, N_{iters}$

4. FORM R.H.S. VECTOR, $\underset{\sim}{R}$.

$\underset{\sim}{R} \leftarrow \underset{\sim}{b}$

A

FLOW CHART III. Element By Element Solver

loop on super-elements: $i_{super} = 1, N_{supers}$

read: $\underset{\sim}{A}_{i_{super}}$

localize: $\underset{\sim}{x} \longrightarrow \underset{\sim}{x}_{i_{super}}$

globalize: $\underset{\sim}{R} \leftarrow \underset{\sim}{R} - \underset{\sim}{A}_{i_{super}} \underset{\sim}{x}_{i_{super}}$

5. SOLUTION PHASE

$\Delta\underset{\sim}{x} \leftarrow \underset{\sim}{M}_{reg}^{-\frac{1}{2}} \underset{\sim}{R}$

loop on number of passes: $i_{pass} = 1, N_{pass}$

loop on super elements: $i_{super} = 1, N_{supers}$

localize: $\Delta\underset{\sim}{x} \longrightarrow \underset{\sim}{x}_{i_{super}}$

read: $\underset{\sim}{A}_{i_{super}}^{-1}$

solve: $\Delta\underset{\sim}{x}_{i_{super}} \leftarrow \underset{\sim}{A}_{i_{super}}^{-1} \Delta\underset{\sim}{x}_{i_{super}}$

globalize: $\Delta\underset{\sim}{x} \leftarrow \Delta\underset{\sim}{x}_{i_{super}}$

flip order of super elements

$\Delta\underset{\sim}{x} \leftarrow \underset{\sim}{M}_{reg}^{-\frac{1}{2}} \Delta\underset{\sim}{x}$

6. QUASI NEWTON/COMPLEMENTARY UPDATE

(see chapter III for details)

7. CONVERGENCE TEST

$x_{tol} \leftarrow ||\Delta\underset{\sim}{x}||/||\underset{\sim}{x}||$

$r_{tol} \leftarrow ||\underset{\sim}{R}||/||\underset{\sim}{b}||$

A

$x_{tol} \leq \varepsilon_1$ and $r_{tol} \leq \varepsilon_2$?

NO

YES

Return

FLOW CHART III (Cont'd.)

TABLES AND FIGURES

	$\underset{\sim}{A}$	$\underset{\sim}{x}$	$\underset{\sim}{b}$
Linear Elastostatics	$\underset{\sim}{K}$	$\underset{\sim}{d}$	$\underset{\sim}{F}$
Linear Elastodynamics	$\underset{\sim}{M}^{*}=\underset{\sim}{M}+\gamma\Delta t\underset{\sim}{C}^{I}+\beta\Delta t^{2}\underset{\sim}{K}^{I}$	$\underset{\sim}{a}_{n+1}$	$\underset{\sim}{F}^{*}_{n+1}=\underset{\sim}{F}_{n+1}-\underset{\sim}{C}\underset{\sim}{v}_{n+1}-\underset{\sim}{K}\underset{\sim}{d}_{n+1}$
Nonlinear Elastostatics	$DN(\underset{\sim}{d}_{(i)})$	$\Delta\underset{\sim}{d}_{(i)}$	$\underset{\sim}{R}_{(i)}=\underset{\sim}{F}-\underset{\sim}{N}(\underset{\sim}{d}_{(i)})$
Nonlinear Elastodynamics	$\underset{\sim}{M}^{*}=\underset{\sim}{M}+\gamma\Delta t\underset{\sim}{C}^{I}_{T}+\beta\Delta t^{2}\underset{\sim}{K}^{I}_{T}$	$\Delta\underset{\sim}{a}$	$\underset{\sim}{R}^{(i)}_{n+1}=\underset{\sim}{F}_{n+1}-\underset{\sim}{M}\underset{\sim}{a}^{(i)}_{n+1}-\underset{\sim}{N}(\underset{\sim}{d}^{(i)}_{n+1},\underset{\sim}{v}^{(i)}_{n+1})$

TABLE 1. Linear Algebraic Equations in Elasticity ($\underset{\sim}{A}\underset{\sim}{x}=\underset{\sim}{b}$)

Number of Iterations	$\Delta t = 18.7 \times \Delta t_{crit.}$		$\Delta t = 187. \times \Delta t_{crit.}$	
	Jacobi	EBE	Jacobi	EBE
Basic	99	14	∞	16
1-D L.S.	38	9	75	6
2-D L.S.	——	8	——	6
BFGS and 1-D L.S.	15	5	21	4
Complementary and 1-D L.S.	55	4	∞	4
Complementary and 2-D L.S.	35	4	63	4

1-D L.S. -- one directional line search

2-D L.S. -- two directional line search

(∞ -- over 100 iterations)

TABLE 2. Example 1 : Comparison Between the Jacobi Method and the EBE Algorithm (load case i)

Number of Iterations	$\Delta t = 18.7 \times \Delta t_{crit.}$		$\Delta t = 187. \times \Delta t_{crit.}$	
	Jacobi	EBE	Jacobi	EBE
Basic	46	15	∞	17
1-D L.S.	20	4	35	6
2-D L.S.	——	3	——	6
BFGS and 1-D L.S.	9	3	12	6
Complementary and 1-D L.S.	19	3	33	5
Complementary and 2-D L.S.	18	3	32	4

1-D L.S. -- one directional line search

2-D L.S. -- two directional line search

(∞ -- over 100 iterations)

TABLE 3. Example 1 : Comparison Between the Jacobi Method and the EBE Algorithm (load case ii)

Number of Iterations	$\Delta t = 18.7 \times \Delta t_{crit.}$		$\Delta t = 187. \times \Delta t_{crit.}$	
	Jacobi	EBE	Jacobi	EBE
Basic	444	50	∞	122
1-D L.S.	151	16	∞	48
2-D L.S.	——	14	——	43
BFGS and 1-D L.S.	∞	8	∞	∞ ↑
Complementary and 1-D L.S.	∞	8	∞	25
Complementary and 2-D L.S.	69	8	∞	24

1-D L.S. -- one directional line search

2-D L.S. -- two directional line search

(∞ -- over 500 iterations)

(∞ ↑ -- machine overflow)

TABLE 4. Example 2: Comparison Between the Jacobi Method and the EBE Algorithm

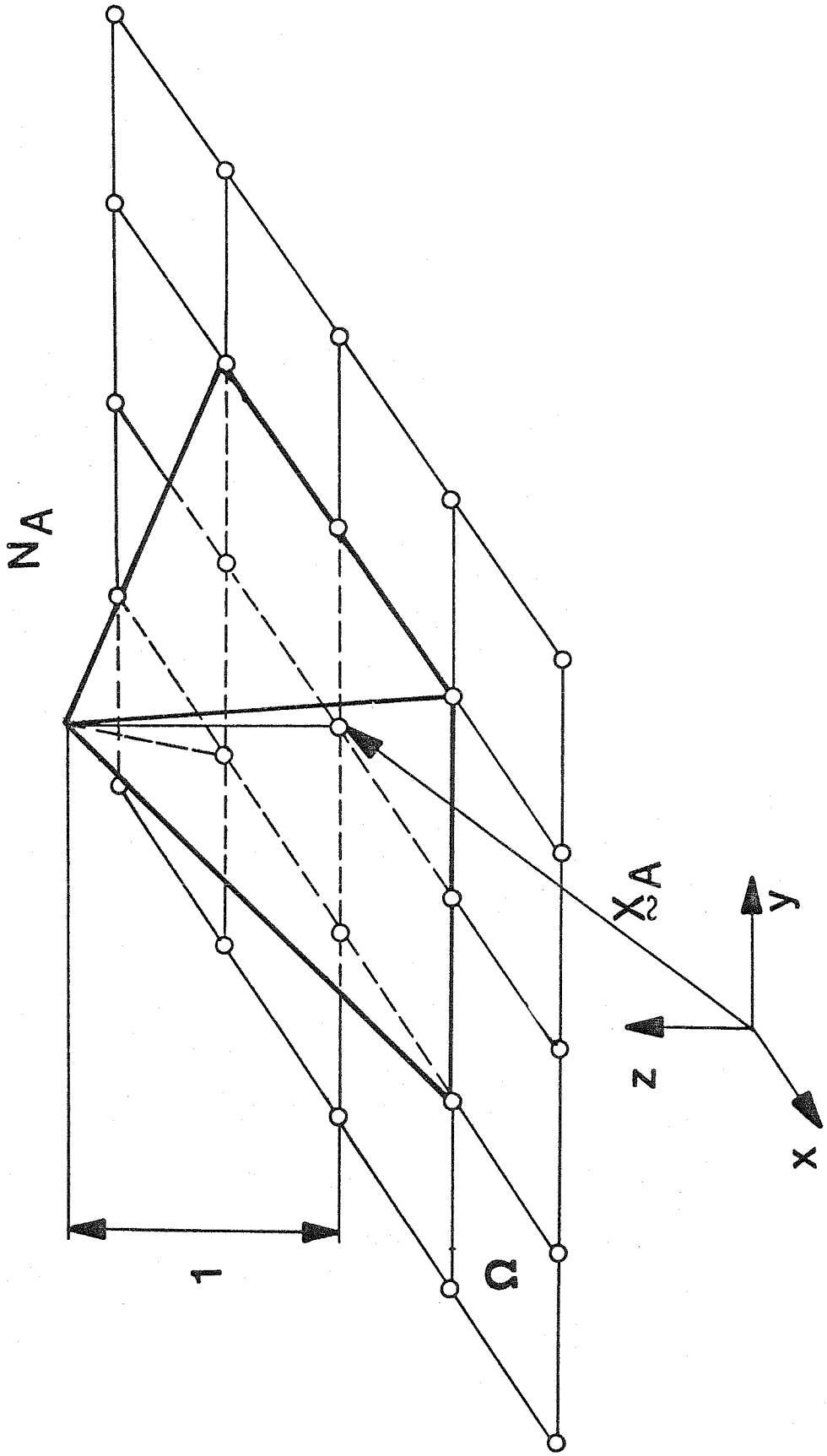


FIGURE 1. 2-D Shape Function N_A

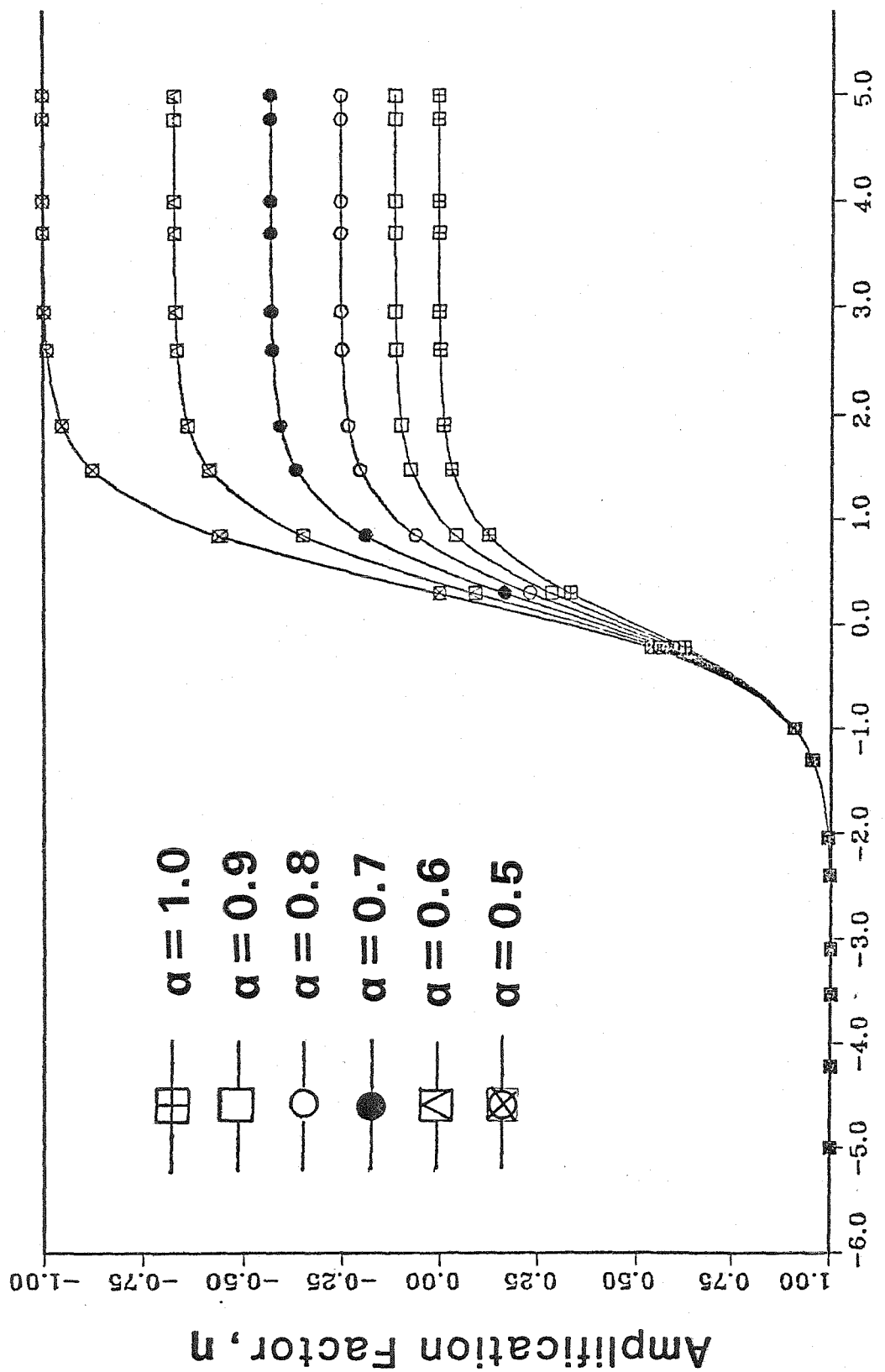
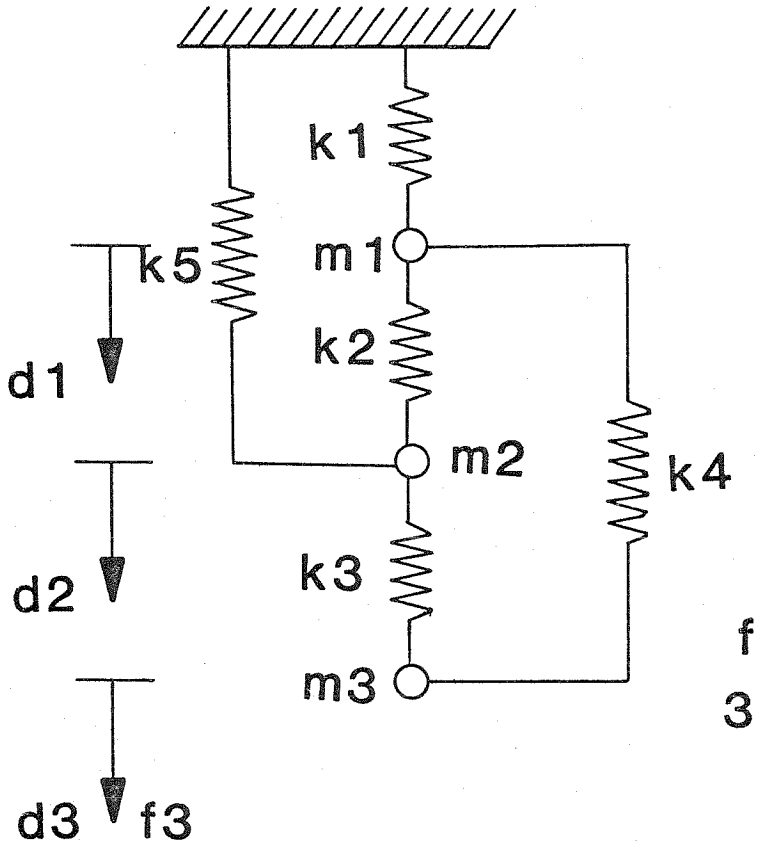
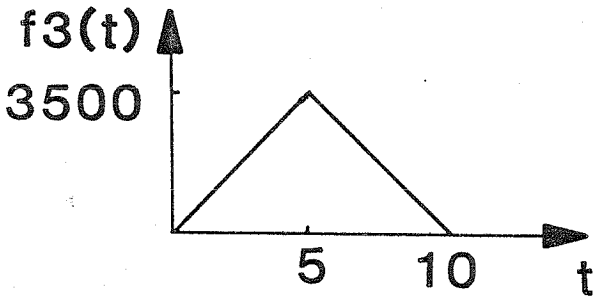


FIGURE 2. Generalized α Method: Amplification Factor



	case1	case2	case3
f3	3000	f3(t)	f3(t)



	case1	case2	case3
k1	6.2	6.2	1.
k2	62.	62.	100.
k3	620.	620.	10000.
k4	124.	124.	200.
k5	12.4	12.4	2.

	case1	case2	case3
m1	0	50	1
m2	0	50	1
m3	0	50	1

FIGURE 3. Test Problem

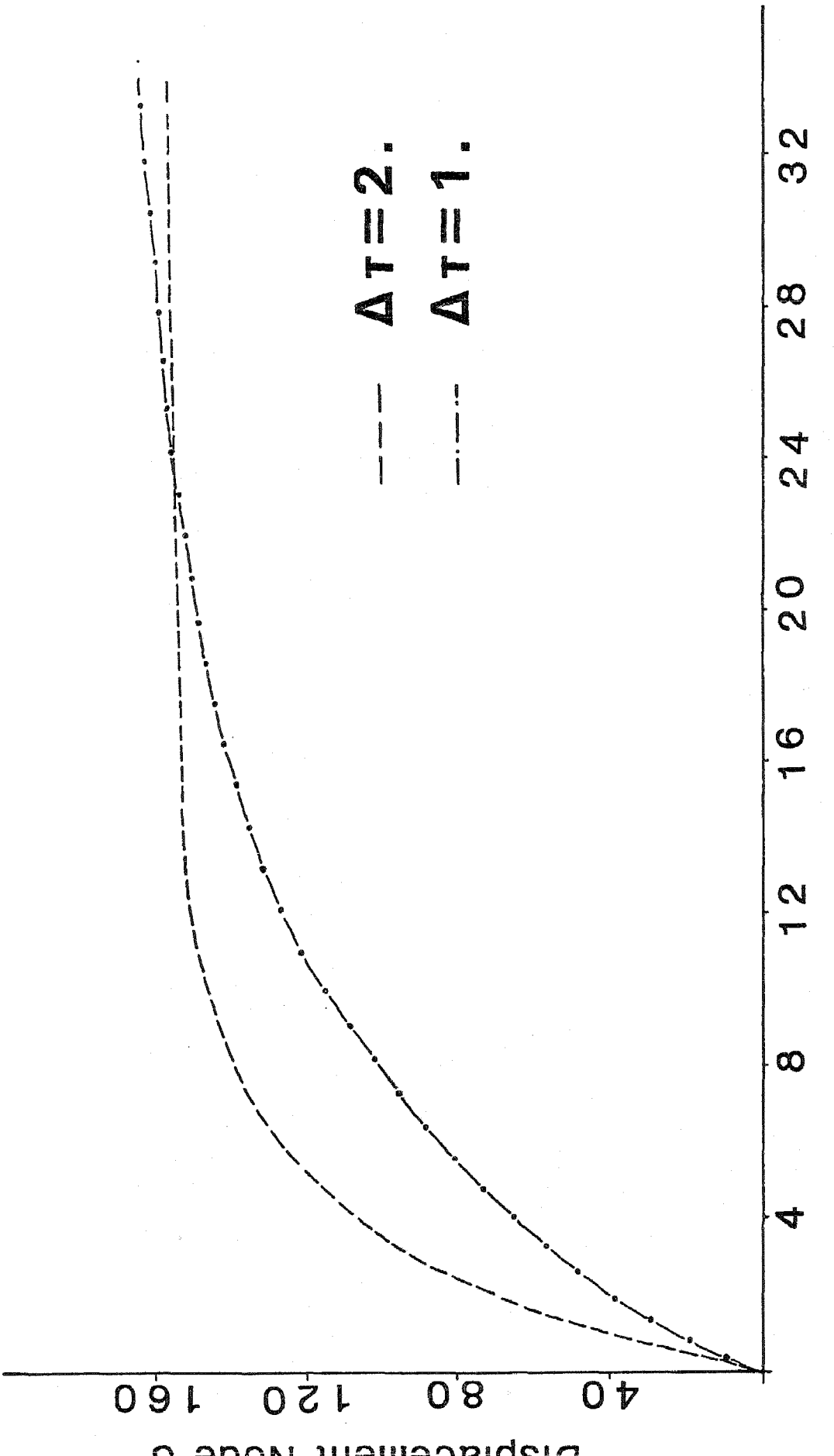


FIGURE 4. Static Test Problem

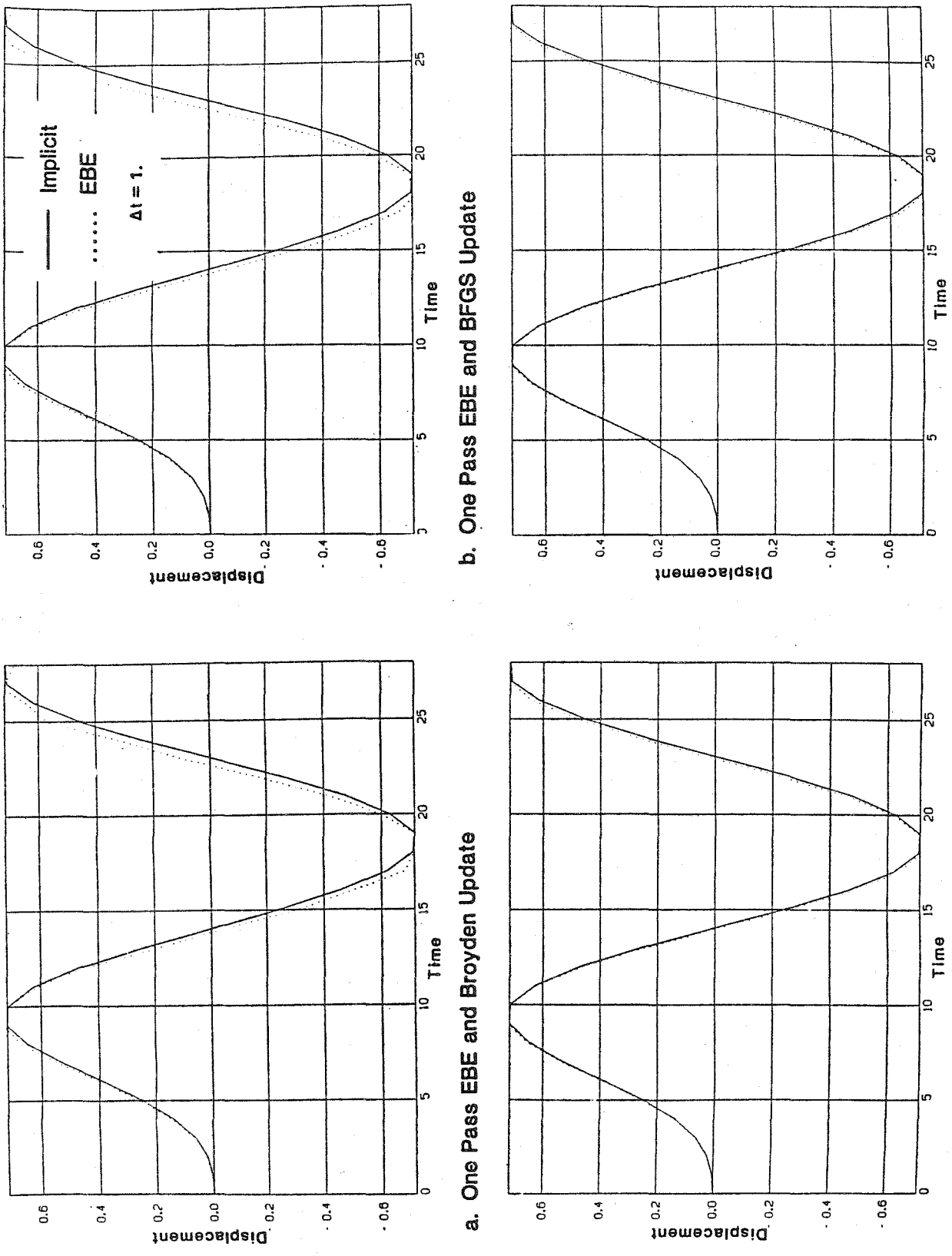
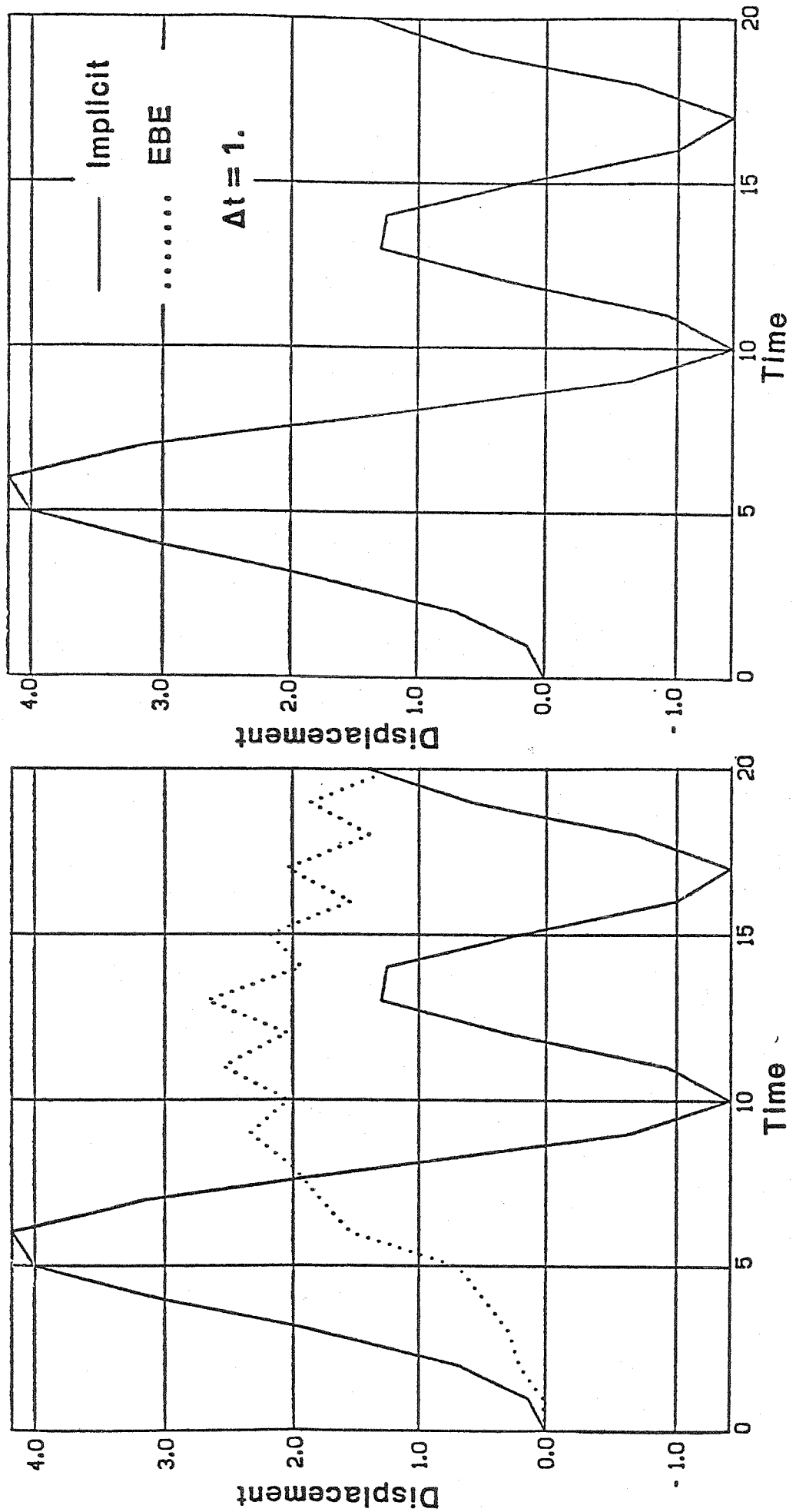


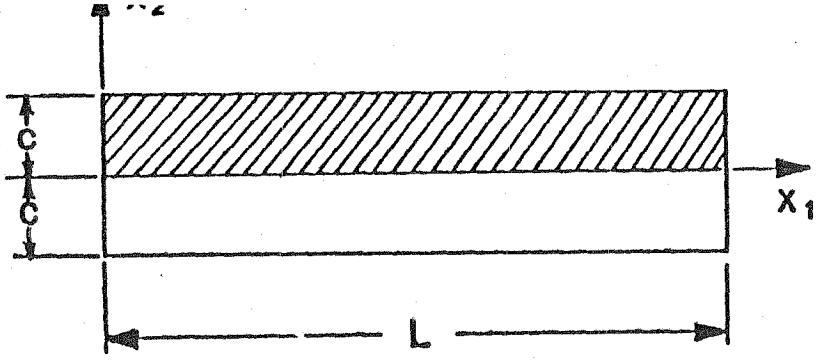
FIGURE 5. Test Problem (case 2): Comparison Between EBE Algorithms (single update results)



a. Two Pass EBE and Broyden Update b. Two Pass EBE and BFGS Update

FIGURE 6. Test Problem (case 3): Comparison Between EBE Algorithms

(three update results)



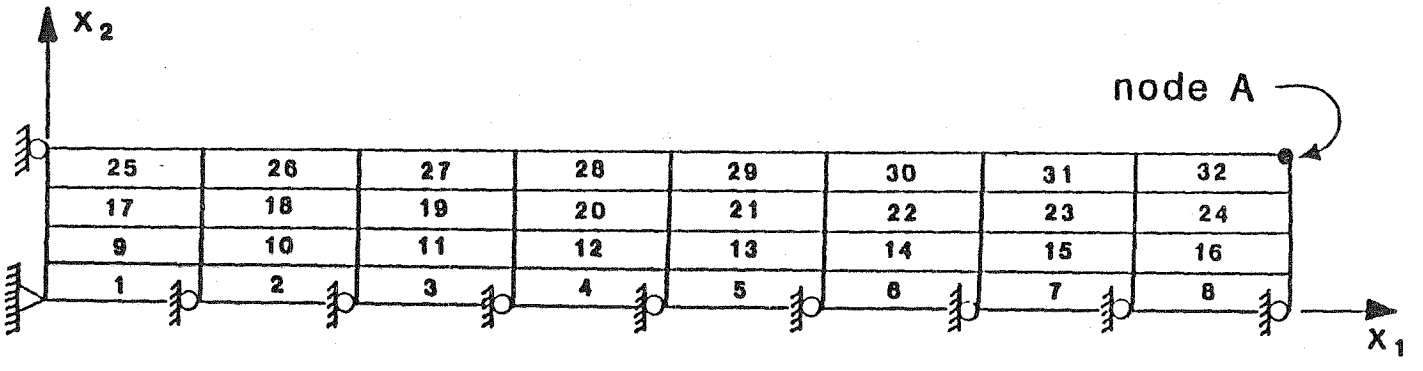
$\mu = .8 \times 10^6$

$\rho = .002$

$L = 16$

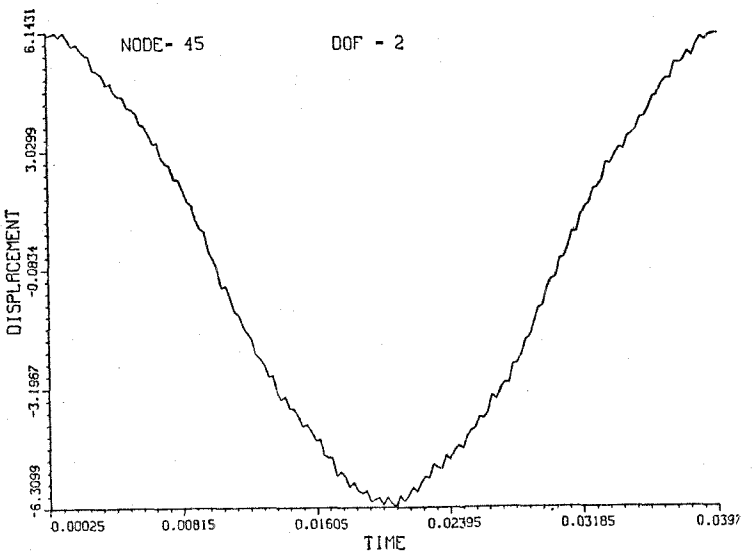
$c = 2$

a. Geometry Definition

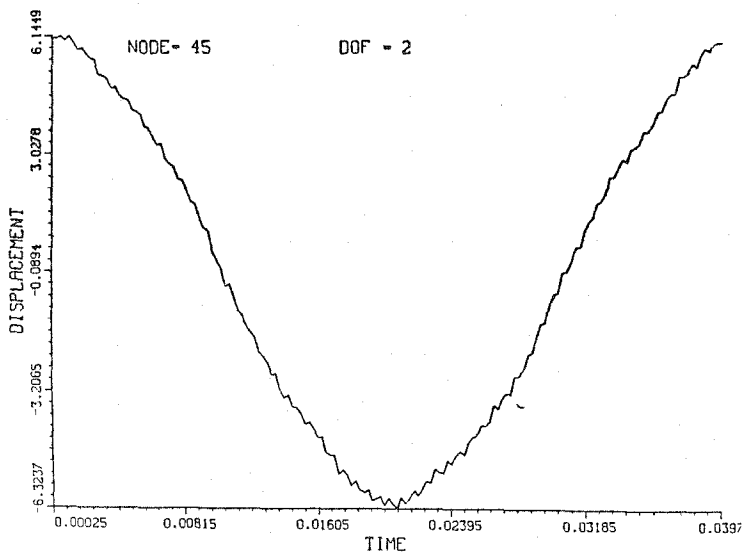


b. Finite Element Mesh

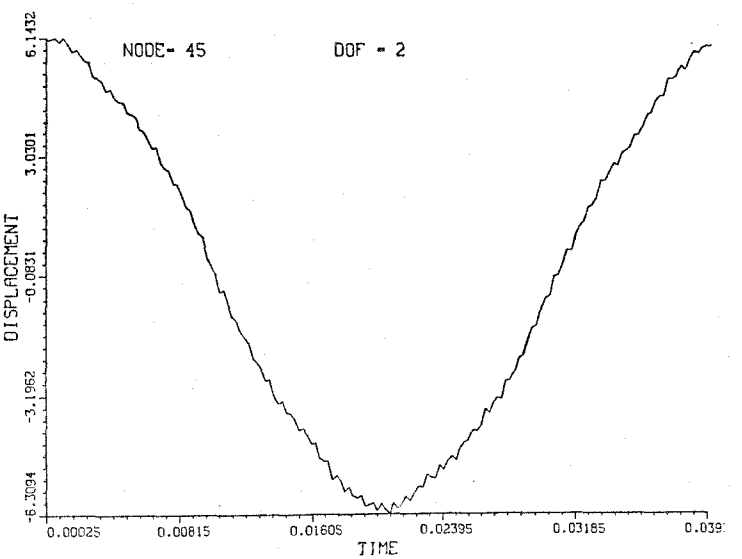
FIGURE 7. Example 1: Uniform Cantilever Beam



a. Implicit



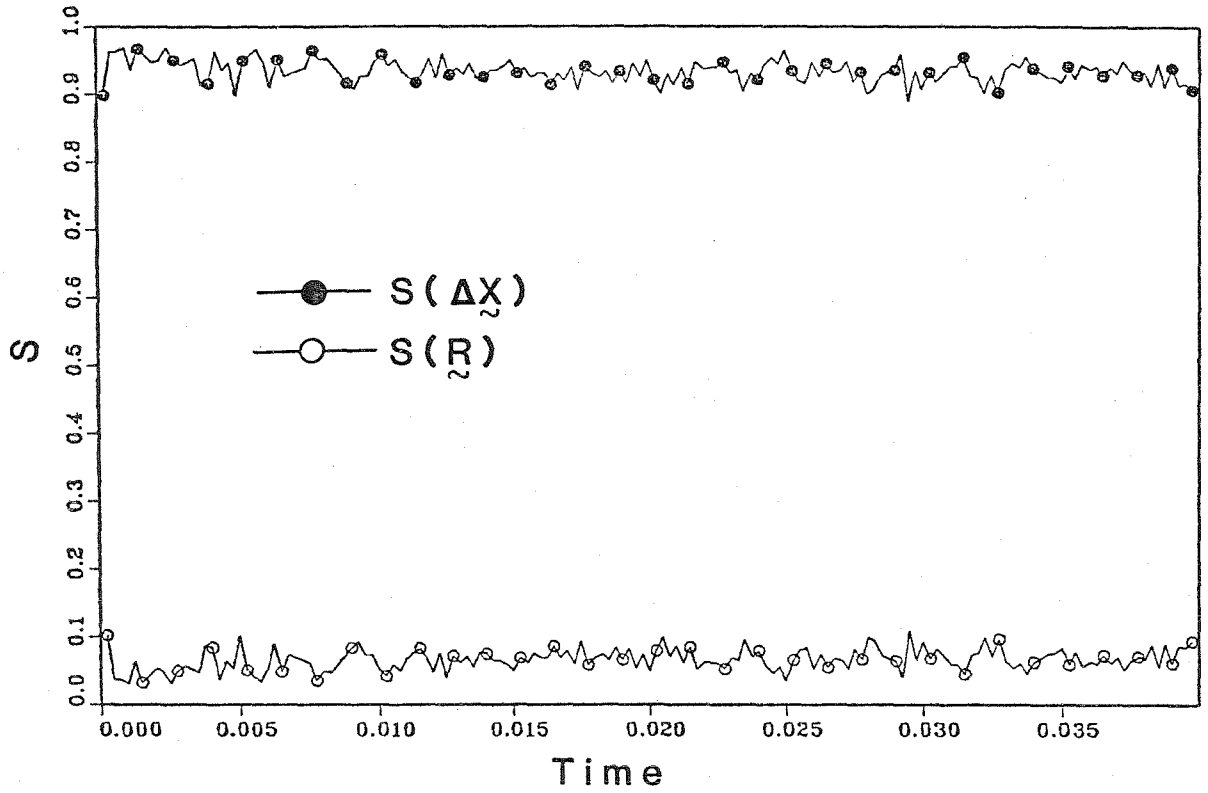
b. BFGS Update



c. Complementary Update

FIGURE 8. Example 1: Comparison of Displacements

a. Two Directional Line Search



b. Complementary Update

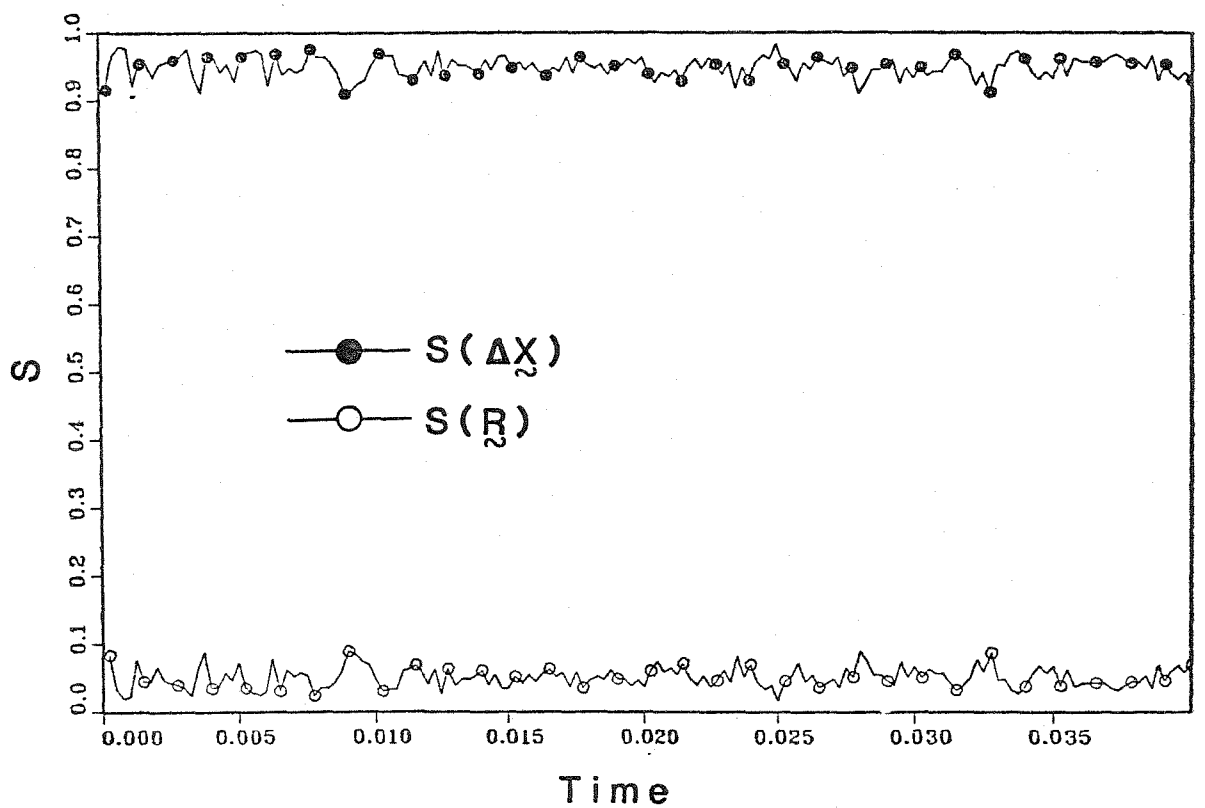
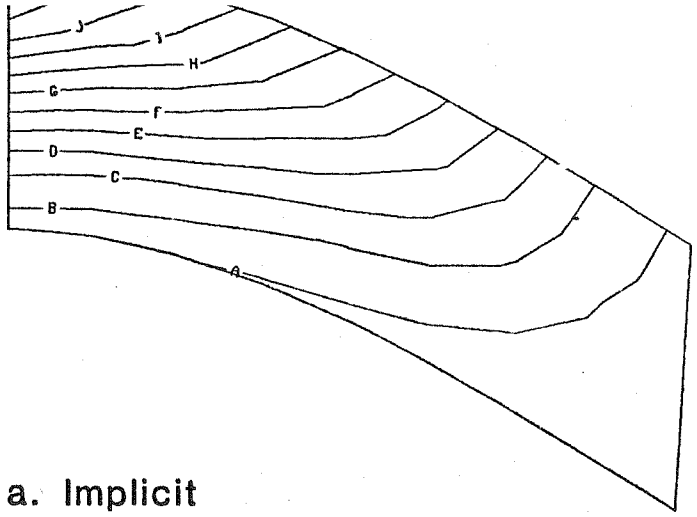
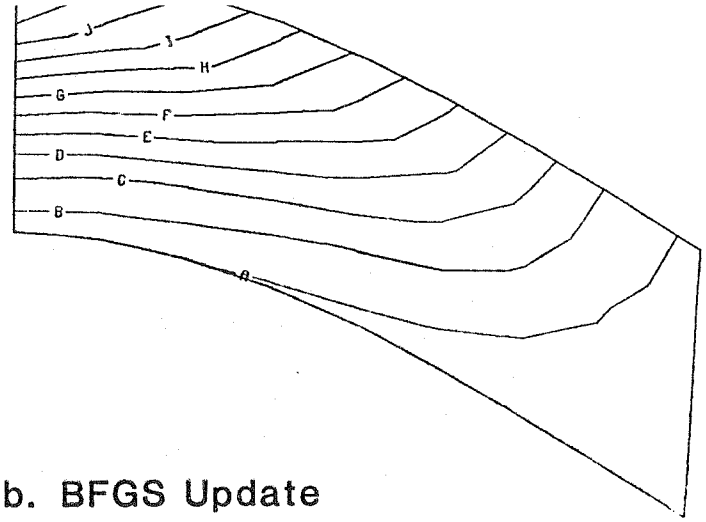


FIGURE 9. Comparison Between Search Directions

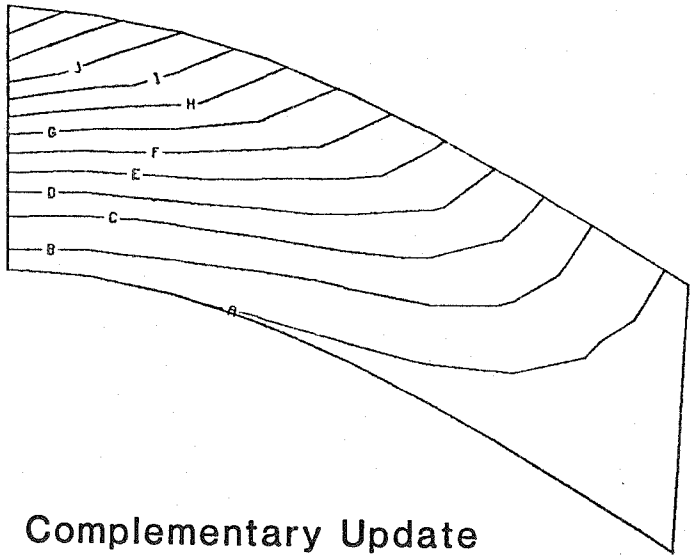


a. Implicit



b. BFGS Update

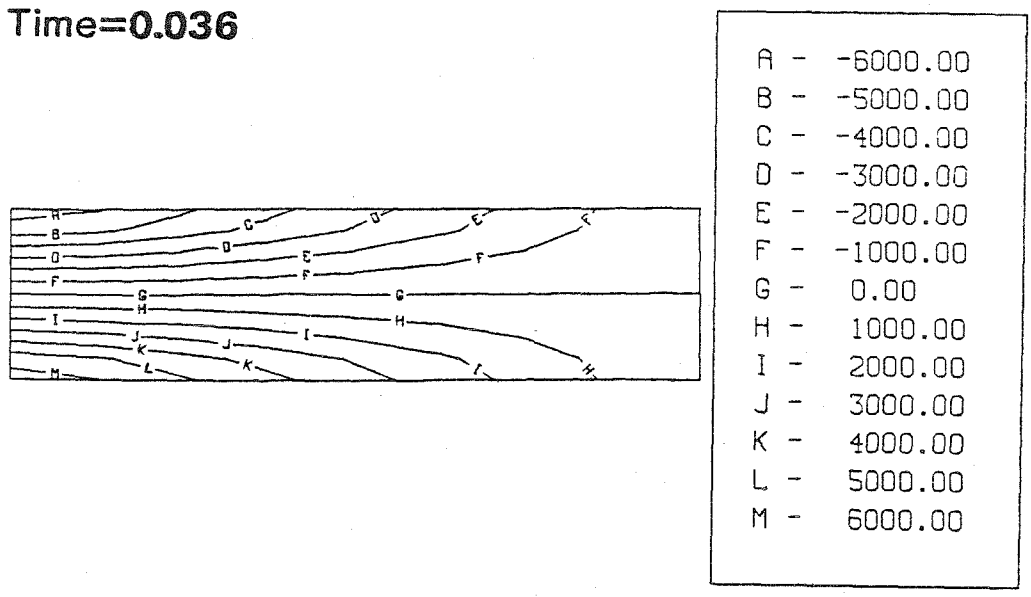
TIME= 0.0187 STEP= 75



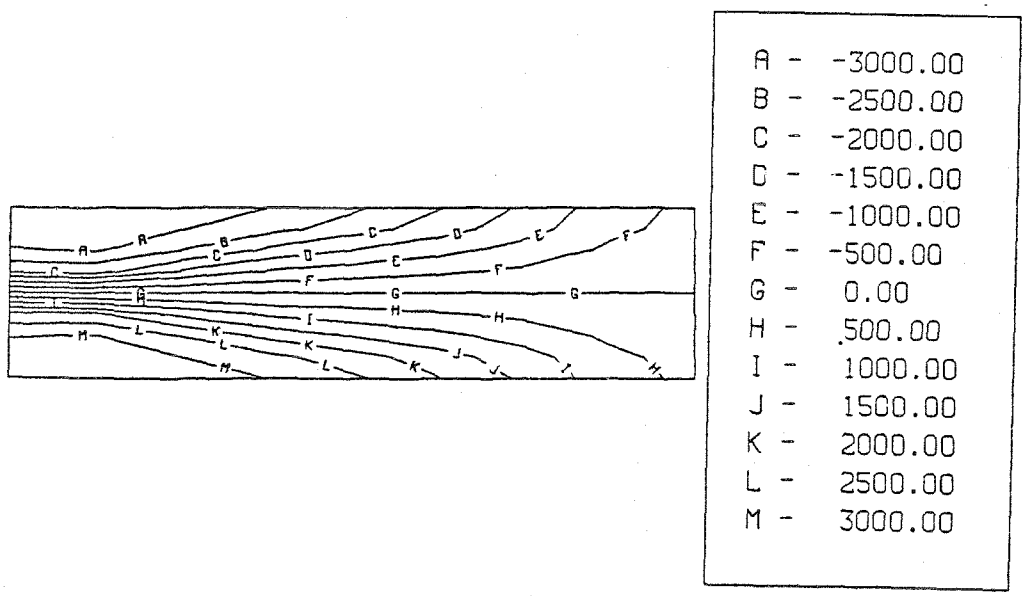
c. Complementary Update

A -	7500.00
B -	15000.00
C -	22500.00
D -	30000.00
E -	37500.00
F -	45000.00
G -	52500.00
H -	60000.00
I -	67500.00
J -	75000.00
K -	82500.00
L -	90000.00
M -	97500.00

FIGURE 10. Example 2: Stress σ_{11}

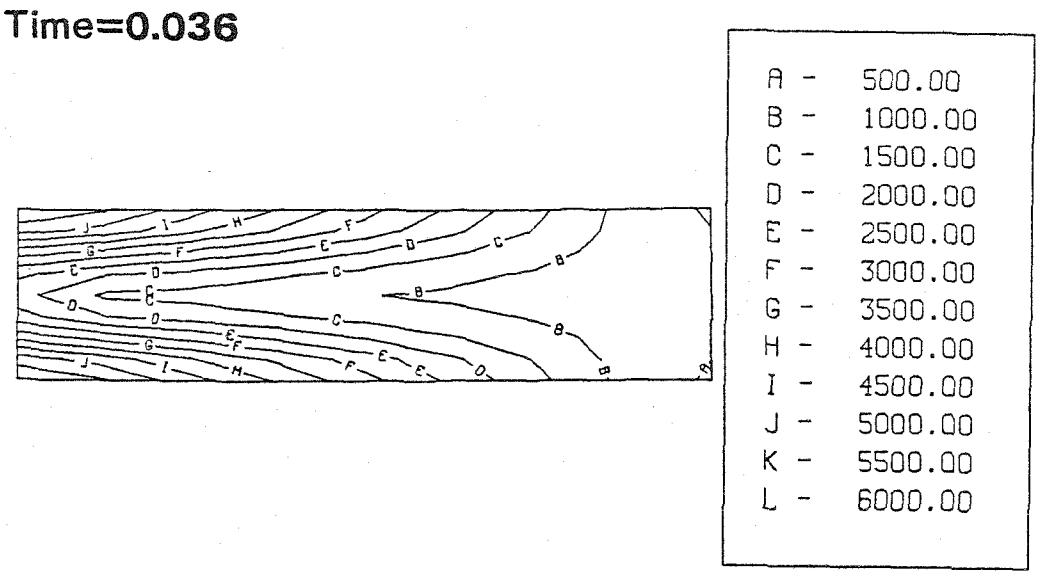


a. Elastic Solution

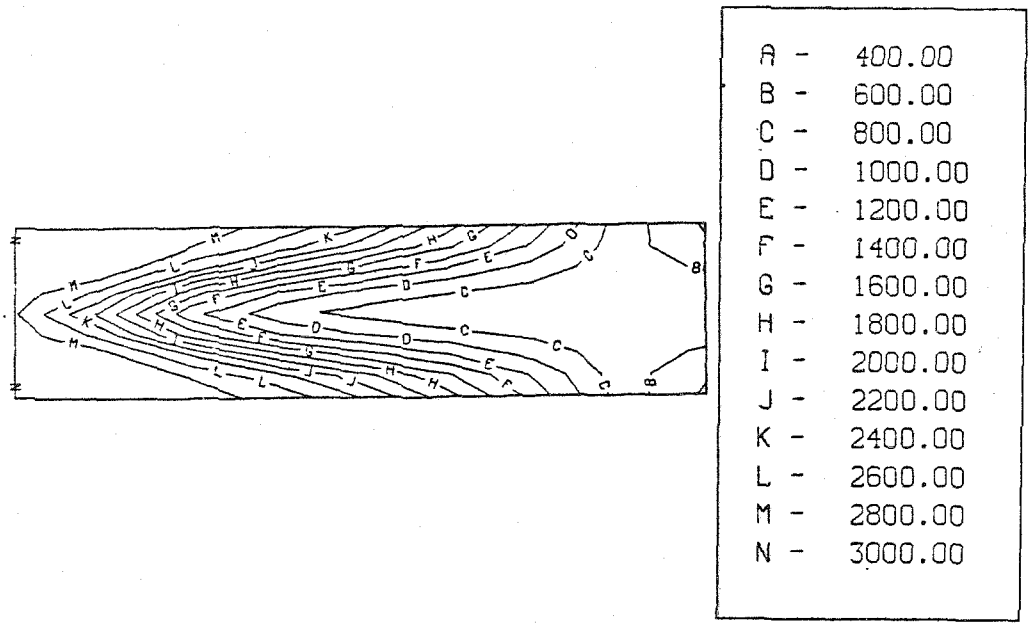


b. Plastic Solution

FIGURE 11. Example 3: Stress σ_{11}

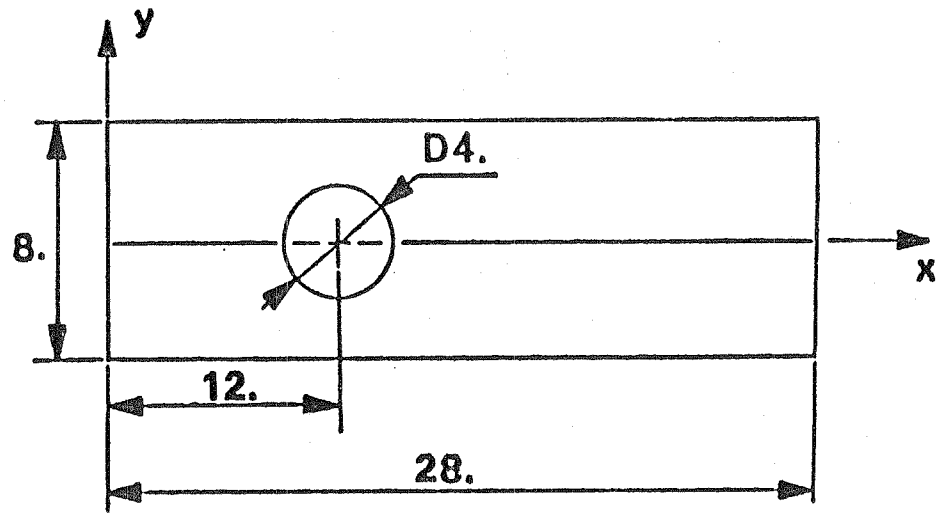


a. Elastic Solution

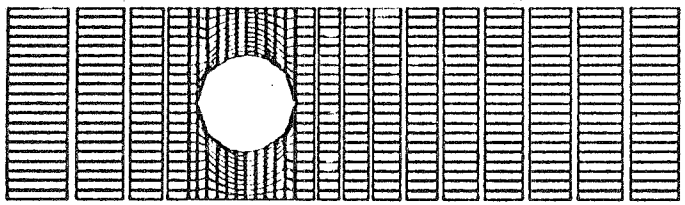


b. Plastic Solution

FIGURE 12. Example 3: Von Mises Stress



a. Geometry Definition



b. Finite Element Mesh

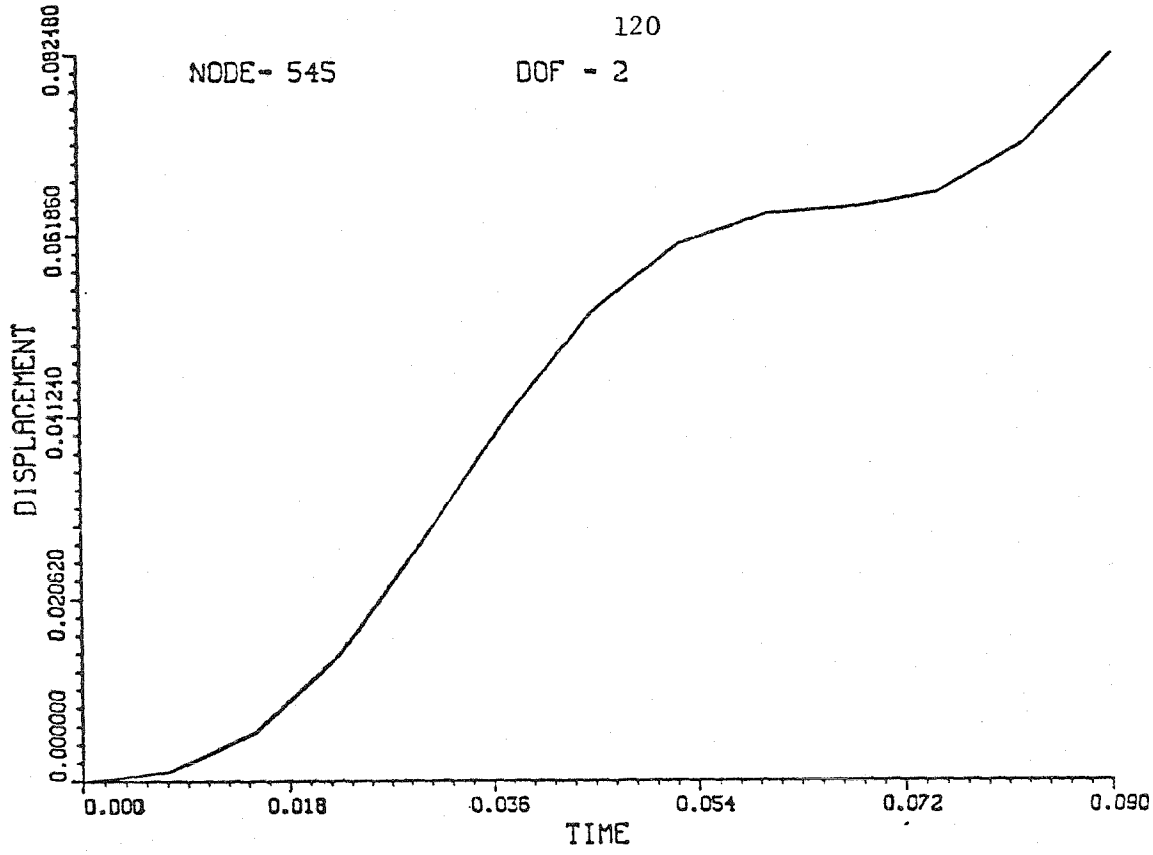
$\rho = 0.02$

$\lambda = 1200000.$

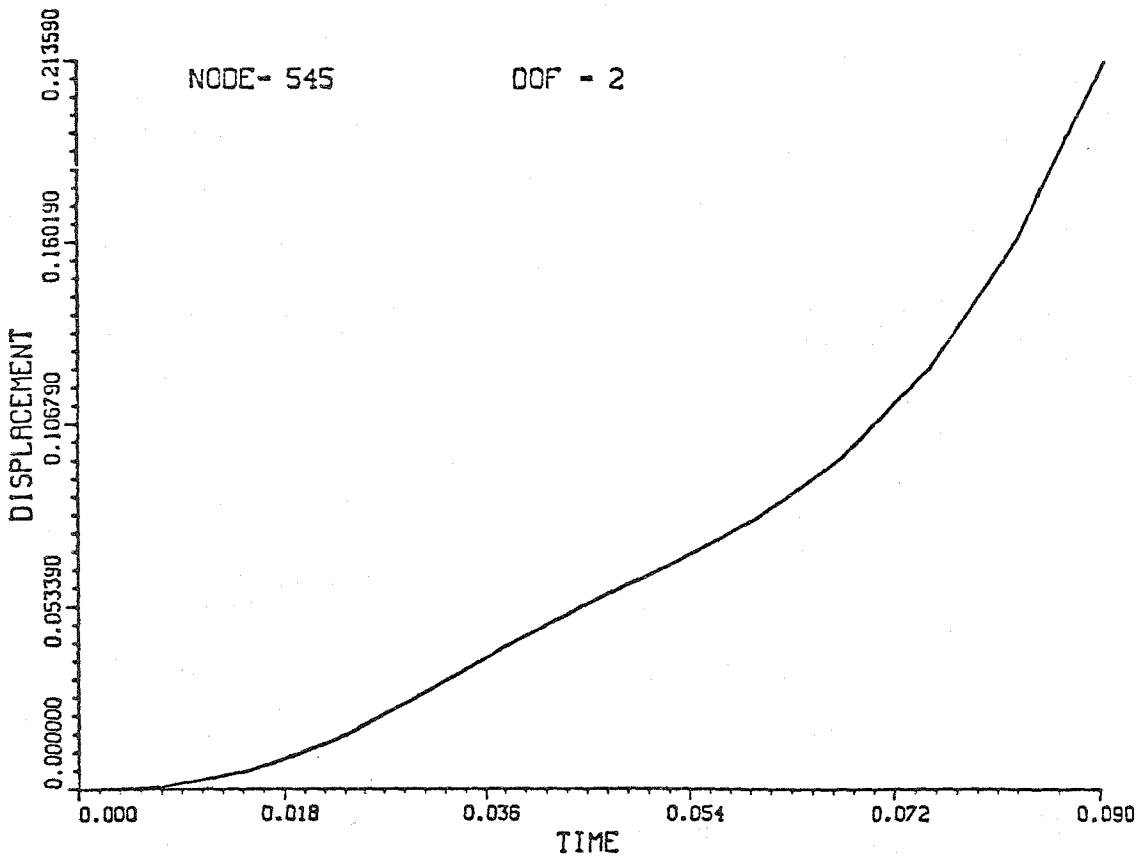
$\mu = 800000.$

$\sigma y = 1000.$

FIGURE 13. Example 4: Plastic Beam

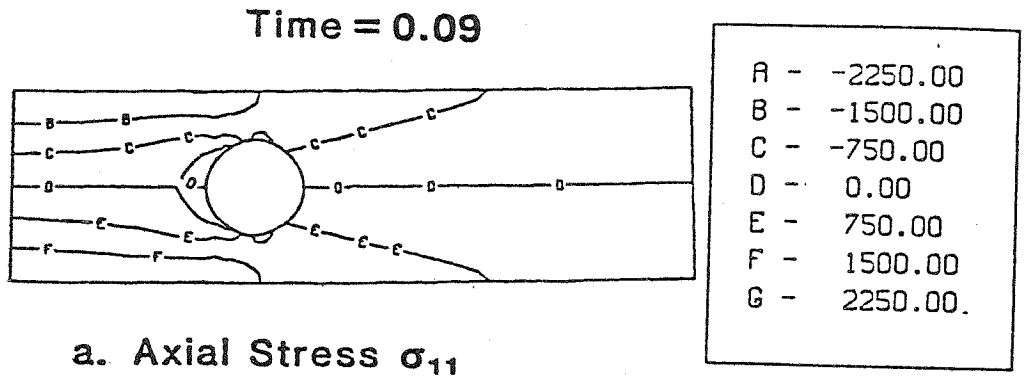


a. Elastic Solution

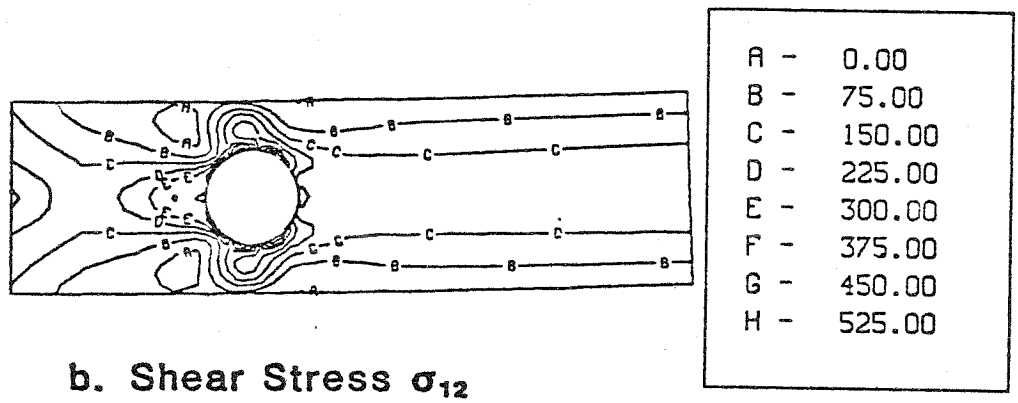


b. Plastic Solution

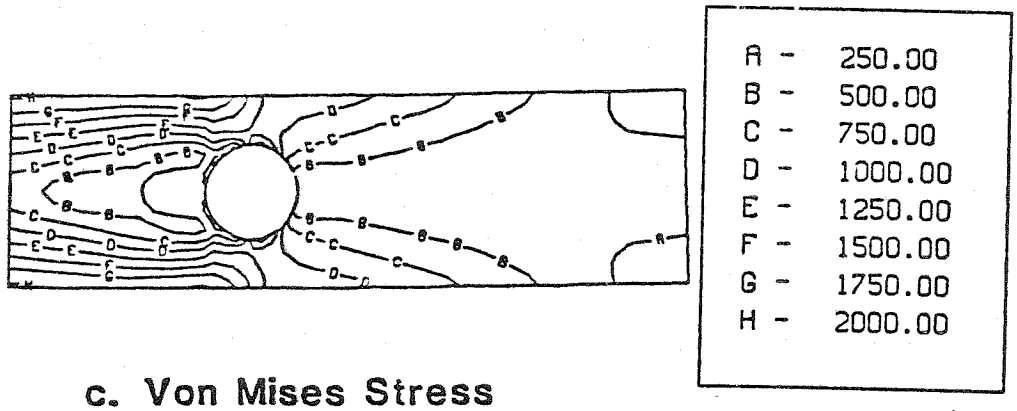
FIGURE 14. Example 4: Displacement Time History



a. Axial Stress σ_{11}

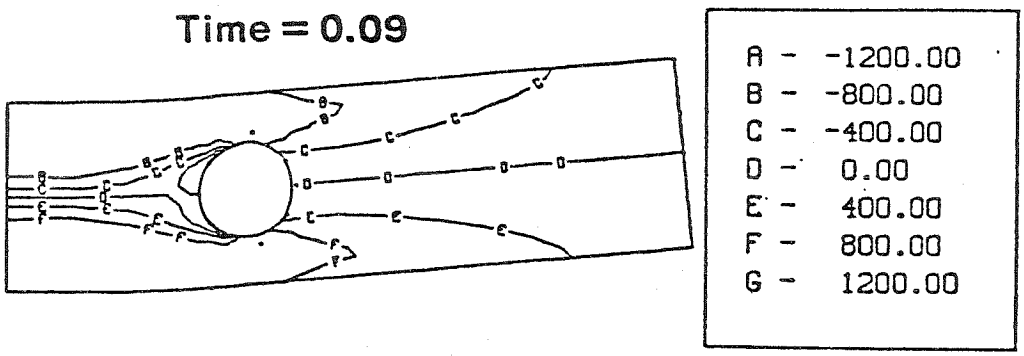


b. Shear Stress σ_{12}

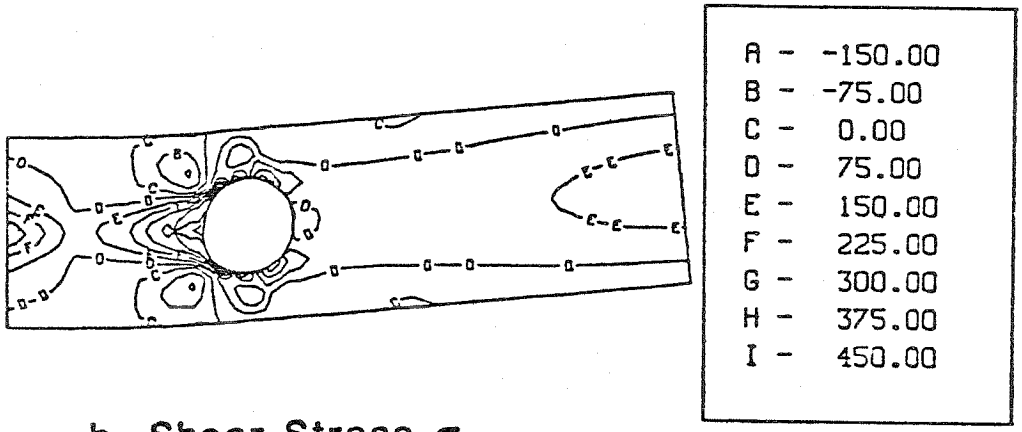


c. Von Mises Stress

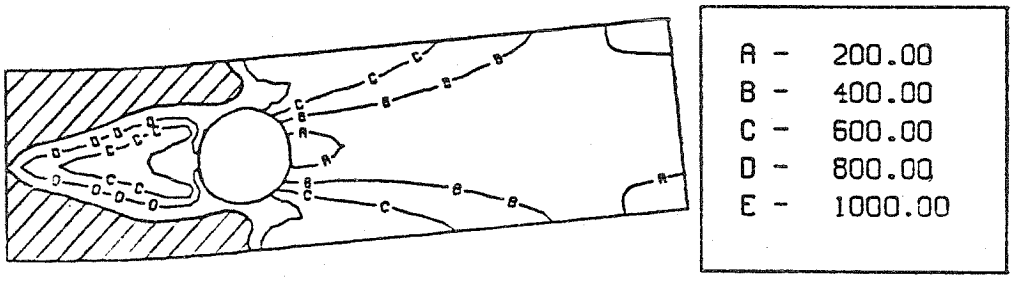
FIGURE 15. Example 4: Stress Distribution (elastic solution)



a. Axial Stress σ_{11}



b. Shear Stress σ_{12}



c. Von Mises Stress

FIGURE 16. Example 4: Stress Distribution (plastic solution)

REFERENCES

B1. M.A. Biot, Mechanics of Incremental Deformation, John Wiley and Sons, Inc., (1965).

B2. J.S. Brew and D.M. Brotten, "Nonlinear Structural Analysis by Dynamic Relaxation," Int. J. for Num. Meth. in Engrg., 3, 145-147, (1971).

C1. R. Courant, K. Friedrichs and H. Lewy, "On the Partial Difference Equations of Mathematical Physics," IBM J. 215-234 (Mar. 1967).

C2. A.C. Cassell, "Shells of Revolution Under Arbitrary Loading and the Use of Fictitious Densities in Dynamic Relaxation," Proc. Inst. Civ. Engrs. 45, 65-78 (Jan 1970).

C3. A.C. Cassell and R.E. Hobbs, "Numerical Stability of Dynamic Relaxation Analysis of Non-Linear Structures," Int. J. for Num. Meth. in Engrg. 10, 1407-1410 (1976).

D1. A.S. Day, "An Introduction to Dynamic Relaxation," The Engineer, 219, 218-221 (1965).

D2. G. Dahlquist and Å. Björk, Numerical Methods, Prentice-Hall, Englewood Cliffs, N.J., (1974).

D3. J.E. Dennis and Jorge More, "Quasi-Newton Methods, Motivation and Theory," SIAM Rev., 19, 46-89 (1977).

D4. J. Douglas and H.H. Rachford, "On the Numerical Solution of Heat Conduction Problems in Two and Three Space Variables," Trans. Amer. Math. Soc., 82, 421-439 (1956).

D5. J. Douglas, "Alternating Direction Methods for Three Space Variables," Numer. Math., 4, 41-63 (1962).

H1. H.M. Hilber and T.J.R. Hughes, "Collocation, Dissipation and 'Overshoot' for Time Integration Schemes in Structural Dynamics," Earthquake Engineering and Structural Dynamics, 6, 94-118 (1978).

H2. H.M. Hilber, T.J.R. Hughes and R.L. Taylor, "Improved Numerical Dissipation for Time Integration Algorithms in Structural Dynamics," Earthquake Engineering and Structural Dynamics, 5, 283-299 (1977).

H3. J.C. Houbolt, "A Recurrent Matrix Solution for the Dynamic Response of Elastic Aircraft," Journal of the Aeronautical Sciences 17, 540-550 (1950).

H4. T.J.R. Hughes, "Analysis of Transient Algorithms with Particular Reference to Stability Behavior" to be published as a chapter in Computational Methods in Transient Analysis (eds. T. Belytschko and T.J.R. Hughes), North-Holland Publishing Company, Amsterdam (in press).

H5. T.J.R. Hughes and W.K. Liu, "Implicit-Explicit Finite Elements in Transient Analysis: Implementation and Numerical Examples", Journal of Applied Mechanics, 45, 375-378 (1978).

H6. T.J.R. Hughes and W.K. Liu, "Implicit-Explicit Finite Elements in Transient Analysis: Stability Theory", Journal of Applied Mechanics, Vol. 45, 371-374 (1978).

H7. T.J.R. Hughes and J.M. Winget, "Finite Rotation Effects in Numerical Integration of Rate Constitutive Equations Arising in Large Deformation Analysis," International Journal of Numerical Mathematic in Engineering, 15, 1862-1867 (1982).

H8. T.J.R. Hughes, "On Consistency Derived Tangent Stiffness Matrices," unpublished manuscript.

H9. T.J.R. Hughes and J.E. Marsden, "The Mathematical Foundations of Elasticity," Prentice Hall, Englewood Cliffs, New Jersey, 1983.

H10. T.J.R. Hughes and K.S. Pister, "Consistent Linearization in Mechanics of Solids," Computers and Sructures 8, 391 - 397 (1978).

H11. T.J.R. Hughes, I. Levit and J. Winget, "Implicit, Unconditionally Stable, Element-By-Element Algorithm for Heat Conduction Analysis," Journal of the Engineering Mechanics Division, ASCE (in press).

H12. T.J.R. Hughes, I. Levit and J. Winget, "An Element-By-Element Algorithm for Problems of Structural and Solid Mechanics," Computer Methods in Applied Mechanics and Engineering (in press).

H13. H.D. Hibbitt and B.I. Karlsson, "Analysis of Pipe Whip," ASME Publication, 79-PVP-122 (1980).

K1. S.W. Key, C.M. Stone and R.D. Krieg, "A Solution Strategy for the Quasi-Static, Large Deformation, Inelastic Response of Axisymmetric Solids," Proceedings of the US-Europe Workshop, Bochum, West-Germany, July 28-30, 1980.

M1. H. Matthies and G. Strang, "The Solution of Nonlinear Finite Element Equations," Intern. J. for Num. Meth. in Engrg., 14, 1613-1626 (1979).

M2. L. Meirovitch, Analytical Methods in Vibrations, The Macmillan Company, 1967.

M3. G.I. Marchuk, "Numerical Methods in Weather Prediction," Academic Press, New York and London, 1974.

M4. G.I. Marchuk, "Methods of Numerical Mathematics," Springer-Verlag, New-York -Heidelberg-Berlin, 1975.

N1. N.M. Newmark, "A Method of Computation for Structural Dynamics," Journal of the Engineering Mechanics Division, ASCE, 67-94, (1959).

O1. J.R.H. Otter, "Dynamic Relaxation," Proc. Inst. Civ. Engrs., 35, 633-656 (1966).

P1. K.C. Park, "Evaluating Time Integration Methods for Nonlinear Dynamic Analysis," in Finite Element Analysis of Transient Non-linear Behavior. Applied Mechanics Symposia Series, ASME, New York (1975).

P2. M. Papadakakis, "A Method for the Automatic Evaluation of the Dynamic Relaxation rarameters," Computer Methods in Applied Mechanics and Engineering, 25, 35-48 (1981).

P3. K.C. Park, "Semi-Implicit Transient Analysis Procedure for Structural Dynamics Analysis," Int. J. for Num. Meth. in Engng., 18, 604-622 (1982).

S1. G. Strang, Linear Algebra and Its Applications, Academic Press, New York (1976).

T1. C. Truesdell and W. Noll, "The Nonlinear Field Theories of Mechanics," Vol. III/3 in Encyclopedia of Physics, Springer-Verlag, Berlin-Heidelberg-New-York, 1965.

T2. D.M. Trujillo, "An Unconditionally Stable, Explicit Algorithm for Finite Element Heat Conduction Analysis", Journal of Nuclear Engineering and Design, 41, 175-180, 1977.

T3. D.M. Trujillo, "An Unconditionally Stable, Explicit Algorithm for Structural Dynamics," Int. J. Num. Meth. in Engng., 11, 1574-1542 (1977).

U1. P. Underwood, "Dynamic Relaxation-A Review," Lockheed Palo-Alto Research Lab. Report. (April 1981).

W1. E.L. Wilson, "A Computer Program for the Dynamic Stress Analysis of Underground Structures, "SESM Report No. 68-1, Division of Structural Engineering and Structural Mechanics, University of California, Berkeley (1968).

W2. A.K. Welsh, "Discussion on Dynamic Relaxation," Proc. Inst. Civ. Engrs., 37, 723-750 (1967).

W3. W.L. Wood, "Comparison of Dynamic Relaxation with Three Other Iterative Methods," Engineer, 224, 683-687 (1967).

W4. R.F. Warming and R. M. Beam, "On the Construction and Application of Implicit Factored Schemes for Conservation Laws", SIAM-AMS Proceedings, Vol. II, 1978.

Y1. N.N. Yanenko," The Method of Fractional Steps," Springer-Verlag, New York-Heidelberg-Berlin, 1971.

Z1. O.C. Zienkiewicz, The Finite Element Method, Third Edition, McGraw-Hill Book Company, UK, (1977).