REPORT NO.
UCB/SESM-83/02

STRUCTURAL ENGINEERING AND
STRUCTURAL MECHANICS

SOLUTION TECHNIQUES IN FINITE ELEMENT ANALYSIS

by
BAHRAM NOUR-OMID
CARLOS RODRIGUES
ROBERT L. TAYLOR

Report to:
Naval Civil Engineering Laboratory

JANUARY 1983

DEPARTMENT OF CIVIL ENGINEERING
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA

PART I

A PRECONDITIONED CONJUGATE GRADIENT METHOD FOR SOLUTION OF FINITE ELEMENT EQUATIONS

ABSTRACT

A desirable advantage of iterative methods is that they provide means of controlling the accuracy of the solution. In particular, when low levels of accuracy are required this can result in faster algorithms than the direct methods. The use of the conjugate gradient algorithm to solve the linearized system of equations is considered. A preconditioning matrix based on a splitting method is constructed. The outcome is an algorithm which results in substantial reduction in storage over direct methods. The above method is compared with its rivals on several quite different problems in structural mechanics and favourable results were obtained.

TABLE OF CONTENTS

1. Introduction

The finite element method of discretization is used to reduce many complex continuum problems to discrete systems. Although this reduction is the most important step in the overall analysis of a structure, solving the discrete problem is far from trivial. In general, the reduced system is nonlinear and an iterative method must be employed to arrive at the solution. Most solution methods are based on some form of Newton's method in which the nonlinear problem is linearized using an initial approximation to arrive at a linear set of simultaneous equations. The solution of the set of linear equations leads to a correction of the initial approximation. When solving the linear equations, one should not loose sight of the primary objective: solving the nonlinear problem.

Iterative methods, such as the conjugate gradient or Lanczos method, are among the many methods that may be used to solve systems of linear equations. The advantage of these methods, when used as the inner loop of the Newton iteration, is twofold.

(i) The linear equation may be solved to any desired level of accuracy as governed by the Newton iteration.

(ii) A considerable reduction in storage can be achieved when no triangular factorization need be performed.

In [4] a method was developed, based on the preconditioned Lanczos method, to realize some of the advantages of iterative methods. In this previous study, the triangular factors of the initial tangent matrix were used to form a preconditioning matrix for the subsequent solution steps. In the present we have eliminated factorizations by employing other preconditioners and further, have reduced the storage needs of the method.

2. A Preconditioned Conjugate Gradient Method

An essential step in nonlinear analysis of structures is solving a linear system of algebraic equations. The preconditioned conjugate gradient method (hereafter called PCG) is one of the many procedures for solving

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x} = 0 \tag{2.1}$$

where $\mathbf{A}$ is an $n \times n$ symmetric positive definite matrix (which is sparsely populated) and $\mathbf{b}$ is the right-hand side vector. In the case of static analysis, $\mathbf{A}$ is the current tangent matrix and in the case of dynamic analysis, $\mathbf{A}$ depends on the mass, damping and stiffness matrices, as well as the time increment.

The initial popularity of the conjugate gradient method was due to a number of factors. In exact arithmetic the method required a maximum of n iterations to solve (2.1) which made the method superior to other iterative methods. In fact conjugate gradient is in the class of semi-iterative methods which also includes the Lanczos algorithm [6]. The disadvantage of direct methods is their large storage demands for keeping the factors of $\mathbf{A}$. The only interface between the conjugate gradient method and $\mathbf{A}$ is through the product $\mathbf{A}\mathbf{v}$ for a given vector $\mathbf{v}$. This is an elegant way of taking advantage of sparsity of $\mathbf{A}$ which has the added advantage that $\mathbf{A}$ need not be known explicitly but only a means of computing the matrix-vector product is required.

The popularity of the conjugate gradient method vanished once it was found that under certain conditions the method required as many as $5n$ or $6n$ steps to reduce the residual to the desired level. This blemish is accounted for by the strong influence of round-off error.

The addition of preconditioning eliminated this difficulty. Instead of solving (2.1) we solve

$$\mathbf{P}^{-1}\mathbf{A}\mathbf{x} = \mathbf{P}^{-1}\mathbf{b} \tag{2.2}$$

for some appropriate choice of $\mathbf{P}$. The object then is to choose $\mathbf{P}$ such that the coefficient matrix of (2.2) is well conditioned.

Theoretical considerations suggest that at the end of each iteration of CG the residual norm is reduced by a factor $\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$ when solving (2.1) where κ is the condition number of $\mathbf{A}$, defined by $\kappa = \|\mathbf{A}\| \, \|\mathbf{A}^{-1}\|$. See [1] for more details. Note that when $\kappa = 1$, one iteration is sufficient to solve the equation. This provides us with a guideline for choosing $\mathbf{P}$. For a well chosen $\mathbf{P}$ only a few iterations reduce the residual norm to the desired level. Here we give an outline for the preconditioned conjugate algorithm:

Given an initial guess $\mathbf{x}_0$, a positive definite preconditioning matrix $\mathbf{P}$, the matrix $\mathbf{A}$ and the right hand side $\mathbf{b}$:

```
(1)  Set $\mathbf{p}_0 = \mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$
(2)  Solve $\mathbf{P}\mathbf{d}_0 = \mathbf{r}_0$, for $\mathbf{d}_0$
(3)  for $k = 0, 1, 2, \cdots$ until convergence do
       (a) $\alpha_k = (\mathbf{r}_k, \mathbf{d}_k)/(\mathbf{p}_k, \mathbf{A}\mathbf{p}_k)$
       (b) $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$
       (c) $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k$
       (d) Solve $\mathbf{P}\mathbf{d}_{k+1} = \mathbf{r}_{k+1}$
       (e) $\beta_k = (\mathbf{r}_{k+1}, \mathbf{d}_{k+1})/(\mathbf{r}_k, \mathbf{d}_k)$
       (f) $\mathbf{p}_{k+1} = \mathbf{d}_{k+1} + \beta_k \mathbf{p}_k$
```

The operation $(\mathbf{v}, \mathbf{u})$ denotes the inner product $\mathbf{v}^T\mathbf{u}$. The algorithm generates a sequence of approximations to the solution $\mathbf{x}$ with a corresponding residual vector $\mathbf{r}_k$. The termination criterion can be chosen based on these quantities. In addition to storage demands for $\mathbf{A}$ and $\mathbf{P}$ the algorithm requires storage for 4 vectors. The total number of operation per iteration is $NZA + NZP + 5N$, where NZA and NZM are the number of operations for forming $\mathbf{A}\mathbf{u}$ and $\mathbf{P}^{-1}\mathbf{v}$ for a given $\mathbf{u}$ and $\mathbf{v}$.

3. Splitting Methods

Here we turn to a topic which at first sight may seem unrelated to the solution of nonlinear algebraic equations. Consider the system of first order differential equations

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t) \tag{3.1}$$

where $\mathbf{x}$ is an n-dimensional vector, the superposed dot, ($\dot{}$), denotes differentiation with respect to time and $\mathbf{f}$ is a function of the unknown vector $\mathbf{x}$ and t.

We consider a special form of $\mathbf{f}$ which can be written as a sum of its subcomponents $\mathbf{f}_i$.

$$\mathbf{f} = \sum_{i=1}^{s} \mathbf{f}_i \tag{3.2}$$

Under these conditions the original problem can be thought as a sum of s sub-problems

$$\dot{\mathbf{x}} = \mathbf{f}_i(\mathbf{x}, t) \qquad i = 1, \ldots, s \tag{3.3}$$

In the case of finite element discretization of the spatial domain the sum in (3.2) ranges over the elements. In other cases the splitting may be formed by other means, one of which is demonstrated in the following section.

A consistent algorithm for the solution of (3.1), based on the notion of a splitting technique [3], can now be constructed as a product of algorithms for the sub-problems. In other words, write the algorithm for (3.3) as

$$\mathbf{x}_{m+1} = S_i^{(h)}[\mathbf{x}_m] \tag{3.4}$$

where $S_i^{(h)}$ is an operator denoting the algorithm and the index m ranges over the increment in time, h. Then the algorithm for (3.1) can be written as

$$\mathbf{x}_{m+1} = S^{(h)}[\mathbf{x}_m] \tag{3.5}$$

where

$$S^{(h)} = \prod_{i=1}^{s} S_i^{(h)} \tag{3.6}$$

One of the disadvantages of the splitting method is its low accuracy. The best that these methods can achieve is second order accuracy. That is the truncation error is of the order of h^3 at best. In the sequel we will use the above procedure to construct a preconditioning matrix for the conjugate gradient algorithm described in section 2. The inherent inaccuracy of the splitting method poses no problem since the algorithm is used only as a preconditioner and therefore one can obtain very high accuracies through the conjugate gradient iteration.

4. Solution of Static Problems

Consider the system of linear first order differential equations

$$\tau \dot{\mathbf{x}} + \mathbf{A}\mathbf{x} = \mathbf{b} \tag{4.1}$$

where τ is a given parameter. Formally the solution to equation (4.1) is

$$\mathbf{x}(t) = e^{-\mathbf{A}t/\tau}(\mathbf{x}_0 - \mathbf{A}^{-1}\mathbf{b}) + \mathbf{A}^{-1}\mathbf{b} \tag{4.2}$$

where $\mathbf{x}_0 = \mathbf{x}(0)$, is an initial condition. We observe from (4.2) thet as t ends to infinity $\mathbf{x}(t)$ converges to the solution of (2.1) for $\tau > 0$. Consiquently (4.1) may be utilised to solve the linear equations (2.1). Indeed this approach has been suggested previously (e.g., see [5]). In general the exponential of a large matrix cannot be easily computed and a numerical solution of (4.1) must be used. In order to achieve a soluion of (2.1) a numerical solution to (4.1) must be assymptotically correct for infinite Δt, or a very large number of time steps must be used to compute the solution at infinite time. Here we are not concerned with constructing accurate solution to (4.1) rather we consider the method as a means of constructing a suitable preconditioning matrix for the conjugate gradient algorithm described above.

Splitting methods may be applied to any problem of the form

$$\dot{\mathbf{x}} = \mathbf{B}\mathbf{x} \tag{4.3}$$

where $\mathbf{B}$ is an additive operator defined by

$$\mathbf{B} = \sum_{i=1}^{s} \mathbf{B}_i \tag{4.4}$$

such that the equations

$$\dot{\mathbf{x}} = \mathbf{B}_i \mathbf{x} \qquad i = 1,\ldots,s \tag{4.5}$$

are significantly easier to solve than the original equations. The time stepping algorithm for the global problem is then the product of all the time stepping algorithms for the subproblems with a fractional time step h/s [3].

The coefficient matrix $\mathbf{A}$ in (2.1) may be written as the sum of its diagonal matrix, $\mathbf{D}$, a strictly lower triangular matrix, $\mathbf{L}$, and a srictly upper triangular matrix such that

$$\mathbf{A} = (\tfrac{1}{2}\mathbf{D} + \mathbf{L}) + (\tfrac{1}{2}\mathbf{D} + \mathbf{L})^T \tag{4.6}$$

The associated subproblems, $\dot{\mathbf{x}} = -(\tfrac{1}{2}\mathbf{D} + \mathbf{L})\mathbf{x}$ and $\dot{\mathbf{x}} = -(\tfrac{1}{2}\mathbf{D} + \mathbf{L}^T)\mathbf{x}$ can be solved easily. Applying a backward difference method with a time step $h/2$ to each of the subproblems, we arrive at

$$\mathbf{x}_{m+1} = \left[\mathbf{I} + \frac{h}{4\tau}(\mathbf{D} + 2\mathbf{L}\right]^{-1}\left[\mathbf{I} + \frac{h}{4\tau}(\mathbf{D} + 2\mathbf{L}^T)\right]^{-1}\left[\frac{h}{2\tau}\mathbf{b} + \mathbf{x}_m\right] \tag{4.7}$$

where $\mathbf{x}_m$ is an approximation to $\mathbf{x}(mh)$. For initial condition $\mathbf{x}_0 = 0$ we get an approximation to $\mathbf{x}(h)$

$$\mathbf{x}_1 = \left\{\frac{h}{2\tau}\left[\mathbf{I} + \frac{h}{4\tau}(\mathbf{D} + 2\mathbf{L})\right]^{-1}\left[\mathbf{I} + \frac{h}{4\tau}(\mathbf{D} + 2\mathbf{L}^T)\right]^{-1}\right\}\mathbf{b} \tag{4.8}$$

which is compared to the exact solution

$$\mathbf{x}(h) = \left[\mathbf{I} - e^{-\mathbf{A}t/\tau}\right]\mathbf{A}^{-1}\,\mathbf{b} \tag{4.9}$$

Comparing equations (4.8) and (4.9) suggests that the coefficient matrix in (4.8) may be a good approximation to $\mathbf{A}^{-1}$ for large h, and may therefore be an effective preconditioning matrix. The scalar factor $\frac{h}{2\tau}$ may be ignored for preconditioning purposes.

When using this in conjunction with conjugate gradient algorithm of section 2 the preconditioning matrix becomes

$$\mathbf{P} = (\mathbf{I} + \omega/2\mathbf{D} + \omega\mathbf{L})(\mathbf{I} + \omega/2\mathbf{D} + \omega\mathbf{L}^T) \tag{4.10}$$

where $\omega = h/2\tau$ is now a free parameter.

To simplify the choice of ω we scale the stiffness matrix $\mathbf{A}$ such that diagonal of $\mathbf{A}$ is unity. The resulting matrix is $\bar{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$. The system of equations (2.1) now becomes

$$\bar{\mathbf{A}}\,\bar{\mathbf{x}} = \bar{\mathbf{b}} \tag{4.11}$$

where $\bar{\mathbf{x}} = \mathbf{D}^{\frac{1}{2}}\mathbf{x}$ and $\bar{\mathbf{b}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{b}$.

The preconditioned matrix must now be applied to (4.11) resulting in

$$\bar{\mathbf{P}} = (\mathbf{I} + \omega\bar{\mathbf{L}})(\mathbf{I} + \omega\bar{\mathbf{L}}^T) \tag{4.12}$$

where $\bar{\mathbf{A}} = \bar{\mathbf{L}} + \bar{\mathbf{L}}^T$. It is easy to show that preconditioning (4.11) using $\bar{\mathbf{P}}$ is equivalent to preconditioning (2.1) with

$$\mathbf{P} = (\mathbf{D} + \omega\mathbf{L})\mathbf{D}^{-1}(\mathbf{D} + \omega\mathbf{L}^T) \tag{4.13}$$

This can be identified as a member of the class of incomplete Choleski preconditioners [2]. Note that when $\omega = 0$, $\mathbf{P}$ becomes the diagonal matrix $\mathbf{D}$, resulting in the simplest form of preconditioning; diagonal scaling. When $\omega = 1$ then $\mathbf{P} = \mathbf{A} + \mathbf{L}\mathbf{D}^{-1}\mathbf{L}^T$. The error matrix $\mathbf{L}\mathbf{D}^{-1}\mathbf{L}^T$ is rank deficient since $\mathbf{L}$ has zero diagonals. If the norm of $\mathbf{D}$ is larger then the norm of $\mathbf{L}$ then the norm of the error matrix will be small compared to the norm of $\mathbf{A}$. consequently, for most problems it is expected that the optimum ω will be close to unity.

5. Solution of Dynamic Problems

We construct a preconditioning matrix for the linear system of equations arising in a step-by-step algorithm for dynamic analysis of linear and nonlinear structures. In particular, we consider the Newmark algorithm and the preconditioning matrix follows from the splitting method of section 3, in much the same way as for the static problem.

Consider the linear equations of motion

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{f} \tag{5.1}$$

where $\mathbf{M}$ is the diagonal mass matrix, $\mathbf{K}$ is the stiffness matrix, $\mathbf{f}$ is the external load vector and $\mathbf{u}$ is the response of the structure. For simplicity, we ignore the damping effects, but all the following results may be extended easily to the damped case. The linear system of equations arising at every time step of the Newmark method is

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{5.2}$$

where

$$\mathbf{A} = \mathbf{K} + \frac{1}{\beta \Delta t^2}\mathbf{M} \tag{5.3}$$

and

$$\mathbf{b} = \mathbf{f}_{t+\Delta t} + \frac{1}{\beta \Delta t^2}\mathbf{M}[\mathbf{u}_t + \Delta t\, \mathbf{v}_t + (½ - \beta)\Delta t^2 \mathbf{a}_t\ ] \tag{5.4}$$

Here $\mathbf{v}$ and $\mathbf{a}$ are velocity and acceleration vectors, respectively, Δt is the specified time increment, t is the time and $\mathbf{x}$ is the increment of displacement response. The Newmark parameters are chosen such that $\beta \geq (½ + \gamma)^2/4$ with $\gamma \geq ½$. The discretization in time are

$$\mathbf{u}_{t+\Delta t} = \mathbf{u}_t + \Delta t\, \mathbf{v}_t + +½\Delta t^2[(1-2\beta)\mathbf{a}_t + 2\beta \mathbf{a}_{t+\Delta t})] \tag{5.5}$$

$$\mathbf{v}_{t+\Delta t} = \mathbf{v}_t + \Delta t(1-\gamma)\mathbf{a}_t + \gamma \Delta t\, \mathbf{a}_{t+\Delta t} \tag{5.6}$$

The object is to solve (5.2) without forming the factors of $\mathbf{A}$.

A splitting method similar to the one used for equation (4.1) can now be applied to equation (5.1). The matrix resulting from the splitting algorithm can then be used as a preconditioner for (5.2). Consider

$$\mathbf{P} = (\mathbf{L} + \frac{1}{\beta \Delta t^2}\mathbf{M})\mathbf{M}^{-1}(\mathbf{L}^T + \frac{1}{\beta \Delta t^2}\mathbf{M}) \qquad (5.7)$$

where $\mathbf{K} = \mathbf{L} + \mathbf{L}^T$. Multiplying out the terms in (5.7), we obtain

$$\begin{aligned} \mathbf{P} &= \frac{1}{\beta \Delta t^2}[\beta \Delta t^2 \mathbf{L}\mathbf{M}^{-1}\mathbf{L}^T + \mathbf{L} + \mathbf{L}^T + \frac{1}{\beta \Delta t^2}\mathbf{M}] \\ &= \frac{1}{\beta \Delta t^2}[\beta \Delta t^2 \mathbf{L}\mathbf{M}^{-1}\mathbf{L}^T + \mathbf{A}] \\ &= \frac{1}{\beta \Delta t^2}[\mathbf{E}(\Delta t^2) + \mathbf{A}] \qquad (5.8) \end{aligned}$$

where $\mathbf{E}(\Delta t^2) = \beta \Delta t^2 \mathbf{L}\mathbf{M}^{-1}\mathbf{L}^T$.

The preconditioned conjugate gradient algorithm of section 2 is invariant under the scaling of the preconditioning matrix. Therefore, (5.8) shows that $\mathbf{P}$ will tend quadratically to the dynamic stiffness matrix $\mathbf{A}$ as the time step diminishes. In other words, $\mathbf{E}$ tends to the zero matrix quadratically in Δt. We see later that this characteristic results in an effective preconditioning and the solution of equation (5.2) is obtained in as few as 2 or 3 iterations of the preconditioned conjugate gradient algorithm with moderately small Δt.

6. Numerical Examples

In the following, we present a few numerical examples to illustrate some of the characteristics of the proposed preconditioning matrices for the PCG algorithm. This algorithm is implemented in FEAP, a Finite Element Analysis Program (see [8], chapter 24 for more details). All the numerical tests were carried out on a VAX 11/780 at the University of California, Berkeley, using double precision computation.

We first present the results to some static analyses, both linear and nonlinear. In these examples we choose a stopping criterion based on the residual vector and the algorithm is terminated as soon as the norm of this vector is reduced by a factor smaller than a specified tolerance. In our calculations we set the tolerance to 10^{-8}. Next we demonstrate our algorithm on a few dynamic problems. The termination criterion is similar to the static case with a range of different tolerances to demonstrate the effectiveness of the algorithm.

6.1 Static Examples: Linear elastic

a) *132 degree-of-freedom building*

The object of the first problem is to determine the influence of the preconditioning parameter, ω, in eq. (4.13). The total number of PCG iterations required to achieve convergence, varies considerably with ω. To illustrate this dependence, we chose the example model shown in figure 1 which is a 132 degrees of freedom, multistory building, discretized by 176 2-node truss elements each with the same Young's modulus (30×10^6). The cross-sectional area of the girders, columns and diagonals are 20, 40 and 1 respectively. A single load at the top is applied, as shown in figure 7.

In figure 2, we indicate the number of PCG iterations needed to converge as a function of the preconditioning parameter. The shape of this curve is characteristic of the proposed PCG algorithm and consists of three zones:

1) *Small ω:* The preconditioning matrix approaches the diagonal matrix $\mathbf{D}$. In this case, the total number of iterations is less than that for diagonal preconditioning.

2) *Optimum ω:* With this value the algorithm takes the least number of iterations to obtain the solution. Note that the curve is quite flat around ω_{opt} and therefore the total number of PCG steps is insensitive to small changes in the value of ω. Further, as predicted before, the optimum ω is close to unity.

3) *large ω:* In this range the preconditioning matrix approaches $\mathbf{L}\mathbf{M}^{-1}\mathbf{L}^T$ which is a singular matrix (diagonals of $\mathbf{L}$ are zero). In this example with $\omega > 3.0$ the solution may loose accuracy in all significant digits and eventually floating-point overflow occur.

Figure 3 shows the evolution of the residual norm, $\|\mathbf{r}_i\|$, normalized versus $\|\mathbf{r}_0\|$, at the i-th iteration of the PCG algorithm. The residual at each iterate exhibits characteristics typical of conjugate gradient method. Namely, residual norm remains large for a relatively large number of steps before convergence occurs to the specified tolerance. Part of this behavior is due to the loss of orthogonality among the conjugate vectors. Poor preconditioning can also contribute to slow convergence.

b) *"Cantilever beam" type structures*

From the insight we have gained with the preceding example, we now proceed to answer the following question: How to select the ω_{opt}?

No easy analytical solution can be obtained to this question; ω_{opt} depends on the spectrum of $\mathbf{A}$ which is not known *apriori*. However an initial estimate of unity as indicated in section 4 is not an unreasonable choice for ω. The numerical test here is to investigate the dependence of the number of iterations on ω. An accurate upper bound to the total number of PCG steps can be obtained if the condition number of the preconditioned matrix, $\mathbf{P}^{-1}\mathbf{A}$, is known. However the condition number of $\mathbf{P}^{-1}\mathbf{A}$ depends on ω.

The examples we have chosen are summarized in figure 4. Each problem is computed with a range of ω's to obtain ω_{opt}. Figure 5 shows the number of iterations as a function of ω, for these examples. Notice that all the curves are rather flat when close to ω_{opt}, moreover, that ω_{opt} is close to 1.0.

The following table (I) gives the number of iterations for both ω_{opt} and $\omega = 1.0$.

No. of D.O.F. N	ω_{opt}	No. of Iter. for		$\frac{k_2}{k_1}$	$\frac{k_2}{N}$
		ω_{opt}, k_1	$\omega = 1.0$, k_2		
30	1.0	14	14	1.00	0.47
40	1.0	23	23	1.00	0.57
60	1.0-1.3	20	20	1.00	0.33
132	1.3-1.4	37	45	1.22	0.28
160	1.25	33	35	1.06	0.21
240	1.0	199	199	1.00	0.83
300	1.2-1.5	46	50	1.09	0.15

Table I. Comparison of the Number of PCG Iterations for Various ω's.

The last column of Table I shows the ratio of total number of iterations over the number of degree-of-freedom. As expected this ratio remains below unity. The next to last column shows the loss in optimality when using ω equal to unity. Except for the 132 degree-of-freedom system little loss in computational effort results from using ω equal to one.

6.2 Static Examples: Nonlinear Elastic Problem

a) *nonlinear material problem*

In much the same way as the Newton-Lanczos method [4], the PCG algorithm was implemented within a Newton loop. The resulting algorithm possessed all the properties of the Newton-Lanczos algorithm with the exception that it is restricted to positive definite matrices. Simplicity of programing various alterations was the motivating factor in restricting attention to the PCG algorithm. Our primary objective is to compare the PCG algorithm with Newton and modified Newton strategies. For this comparison we use the 132 degree-of-freedom truss building described above, but modified to have the same cross-section for all the members ($A = 20$). Nonlinearity is introduced by a simple yield model in the constitutive equation.

A single load is applied at the top with sufficient magnitude to produce a nonlinear maximum displacement of about twice the maximum elastic one. Figure 6 shows the mesh, the deformed structure and the constitutive equation adopted. In Table II, we indicate the relative computational cost comparisons for different methods. We modified all the algorithms mentioned above to include a line search. This was initially expected to reduce the final cost of the algorithms; in fact, for this problem the three methods were more expensive when a line search was included.

Looking at the results in Table II, it is interesting to note that the PCG algorithm required only one more nonlinear step than the Newton method. Also, due to the fact that only the nonzero terms of the stiffness matrix are stored, the cost for one matrix-vector operation in the PCG algorithm is smaller than for the other methods. For this example the number of terms in the matrix stored in profile form is 1654, however the PCG algorithm requires only 512 nonzero terms. Therefore the cost of one matrix-vector operation is about a third of a profile multiply. More important is the reduction in the over all storage demand. For this example the storage is reduced to 31% of the amount required for a profile stored solution.

Method	No. of iter.	No. of LU factor.	No. of function evaluations	No. of matrix-vec. operations
Newton	6	6	7	6
Mod. Newton	215	1	216	215
PCG	7	0	8	186
Newton + LS	6	6	7	6
Mod. Newton + LS	111	1	112	111
PCG + LS	7	0	8	176

Table II. Cost Comparisons for different Nonlinear Methods (Truss Example).

The average number of PCG iterations was 27, for a preconditioning parameter $\omega = 1.5$. This compares with 37 PCG iterations for a linear problem with much the same structure (see section 6.1a). A lower tolerance for PCG algorithm is used in the earlier stages of the Newton loop which accounts for the lower average number of iterations (see [4] for more details).

In this test the total cost for the PCG algorithm was twice the Newton cost. However, this ratio is expected to drop well below 1.0 for three-dimensional structures where the cost of a factorization is large compared to the matrix-vector operation. Moreover, as noted previosly we require substantially less storage space.

b) *finite deformation problem*

In figure 7, we show a plane strain rubber block subjected to large deformation. We employ a 4-node element and a Mooney-Rivlin material as described in

[7]. The rubber block is discretized by 144 elements (12 x 12 mesh) with 286 degrees of freedom. The stiffness matrix stored in profile form requires 7618 storage spaces of which only 1799 are nonzero; corresponding to a 76% saving in storage when using PCG.

The rubber block is stretched to 50% of its original length in load 5 steps. The cost comparison of both the PCG and Newton algorithms is summarized in the following table.

displ.	Nonlinear PCG		Newton
u	No. of Iterat.	Averg. No. Matrix op.	No. of Iter.
0.1	6	43	7
0.2	6	45	6
0.3	7	47	6
0.4	6	47	5
0.5	6	44	5

Table III. Comparision of PCG Algorithm and Newton Method (Rubber Block).

Again, both PCG and Newton require almost the same number of nonlinear steps to converge. What is more interesting is that the number of PCG iterations is quite constant, even for the highly nonlinear range. When comparing this test with the previous 132 degree-of-freedom building example, we notice that the number of iterations in the PCG algorithm, as expected, does not increase as fast as the number of degree-of-freedom.

6.3 Dynamic example

In this example, we wish to indicate the effectiveness of solving approximately, a linear elastic dynamic problem using the PCG algorithm. This is done for a series of time steps and tolerances. Since the PCG algorithm involves no factorization steps it can solve nonlinear dynamic problems with the same amount of effort as for the linear case. However, in this study we select a linear problem. To limit the computer costs, we selected a structure having 20 4-node plane stress elements defined in section 6.1(b). The dynamic problem consists of releasing the structure from an initially deformed configuration and letting it vibrate freely. The mesh, material properties and initial state are given in figure 8. The time steps chosen are Δt = 0.5, 0.2, 0.1, 0.05 and 0.025 seconds, which correspond to $1/120 < \Delta t / T < 1/6$, where $T = 3.0$ sec. is the fundamental period of vibration of the structure. For comparison, the critical time step for an explicit analysis would be Δt_{cr} = 0.01 sec., for a bulk wave velocity of 911/s. In order to see the effect of solving approximately the set of equations, we use the three following tolerances: $tol = 10^{-4}$, 10^{-2} and 10^{-1}. In figure 9, we plot the y-displacement of node 1 for the time step $\Delta t = 0.1$ for the three tolerances; in figure 9, we show the corresponding relative error e, i.e.

$$e = \frac{\delta_N(t) - \delta_{pcg}(t)}{\delta_0}$$

where δ_N is the displacement obtained using the Newmark method, δ_{pcg} is the corresponding results obtained using PCG and $\delta_0 = 0.172$ is the initial applied displacement. The results clearly show that the tolerance 10^{-1} is too large and leads to inaccurate results. The error using a tolerance of 10^{-2} is about 1%, while there is no visible error for $tol = 10^{-4}$ (less than 0.01 percent). When we reduce the time step to $\Delta t = 0.05$ (half the preceding), the results improve substantially: while we see no difference between tolerances 10^{-4} and 10^{-2}, there is only 1% error when using 10^{-1} (fig. 10). For smaller time steps, no differences are seen in the first five digits.

Finally, figure 11 shows the average number of iterations as a function of $T/\Delta t$. The reduction of the number of iterations as Δt tends to zero is quite interesting: for a tolerance of 10^{-2}, this number drops from 30 to 4 iterations when $\Delta t / T$ changes from 1/6 to 1/120. This reduction is totally due to the convergence of the preconditioning matrix to the dynamic stiffness matrix, $\mathbf{A}$. Such a small time step is not unusual in many applications, e.g. impact problems.

7. Closure

In this report we have described our initial efforts to construct a solution method for the algebraic equations arising from finite element solution of linear and non-linear problems. Both static and dynamic problems are considered. For nonlinear problems, Newton's method is used to generate a sequence of linear problems. A preconditioned conjugate gradient method is used to solve the linear set of equations. A method for constructing an effective preconditioning matrix in terms of an additive decomposition of the coefficient matrix is introduced separately for the static and dynamic cases. Several example problems are solved demonstrating the features of the proposed method.

In order to further evaluate the method additional work is required. In particular we recommend that the conjugate gradient part of the algorithm be replaced by the Lanczos method as described in [4,6]. This will permit consideration of indefinite problems, such as those resulting from use of Lagrange multiplier methods (e.g.,contact problems, etc.). In addition it is essential to test the method on larger problems, preferably some three-dimensional problems where sparsely populated coefficient matrices with rather large mean column heights occur. Further analyses for significant non-linear problems should also be performed. Finally, some efforts to adaptively compute an optimal value for ω should be explored.

REFERENCES

[1] Concus, P., G. H. Golub and D. P. O'Leary, "A Generalized Conjugate Gradient Method for the Numerical Solution of Elliptic Partial Differential Equations," *Sparse Matrix Computations*, ed. by J. R. Bunch and D. J. Rose, Academic Press, New York, 1976.

[2] Kershaw, D. S., "The Incomplete Cholesky-Conjugate Gradient Method for Solution of System of Linear Equations," *Journal of Computational Physics*, vol. 26, pp. 43-65, 1978.

[3] Gourlay, A. R., "Splitting Methods for Time Dependent Partial Differential Equations," *The State of the Art in Numerical Analysis*, ed. by D. Jacobs, Academic Press, New York, 1977.

[4] Nour-Omid, B., B. N. Parlett and R. L. Taylor, "A Newton- Lanczos Method for Solution of Nonlinear Finite Element Equations, *Computers and Structures*, vol. 16, No. 1-4, pp. 241-252, 1982.

[5] Park, K. C. and J. M. Housner, "Semi-Implicit Transient Analysis Procedures for Structural Dynamic Analysis," *International Journal for Numerical Methods in Engineering*, vol. 18, pp. 609-622, 1982.

[6] Parlett, B. N., "A New Look at the Lanczos Algorithm for Solving Symmetric Systems of Linear Equations, *Linear Algebraic Applications*, vol. 29, pp. 323-346, 1980.

[7] Simo, J. C. and R. L. Taylor, "Penalty Formulations for Rubber-like Elasticity, *Report No. UCB/SESM 81/03*, Department of Civil Engineering, University of California, Berkeley, November 1981.

[8] Zienkiewicz, O. C., *The Finite Element Method*, Third Edition , Mc-Graw Hill, Inc., London, 1977.

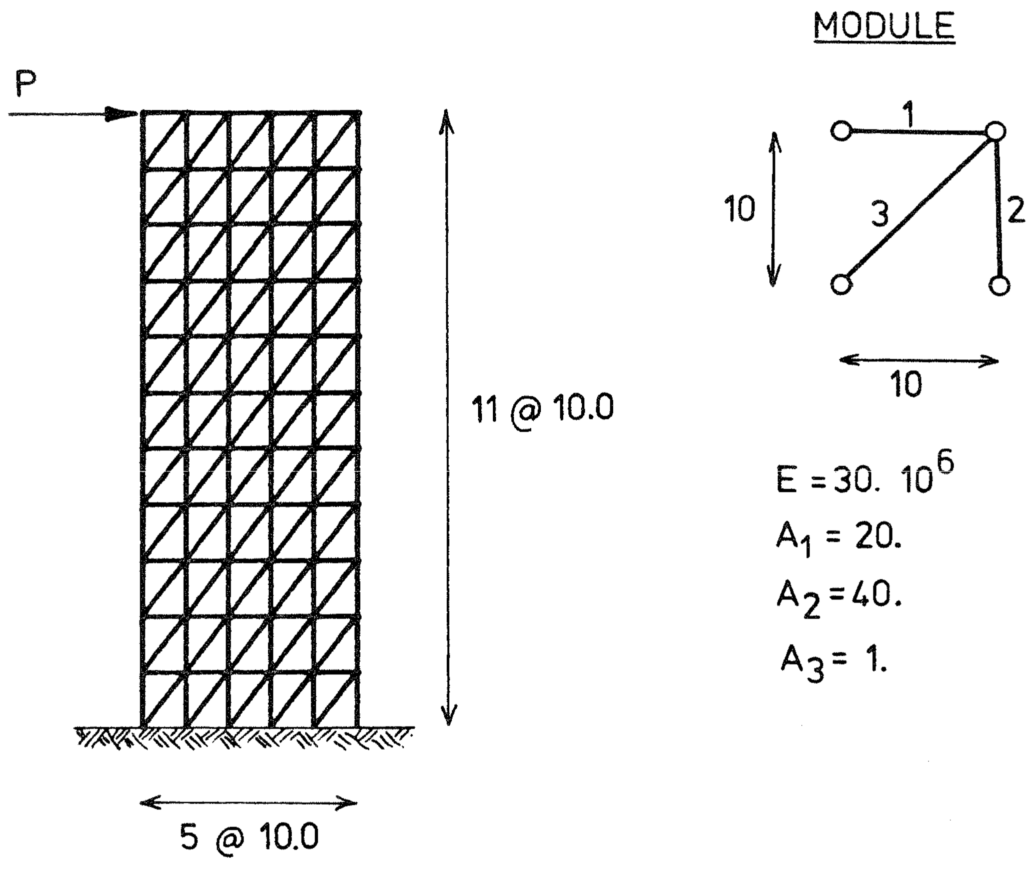


Figure 1. 132 Degrees of Freedom Truss Building.

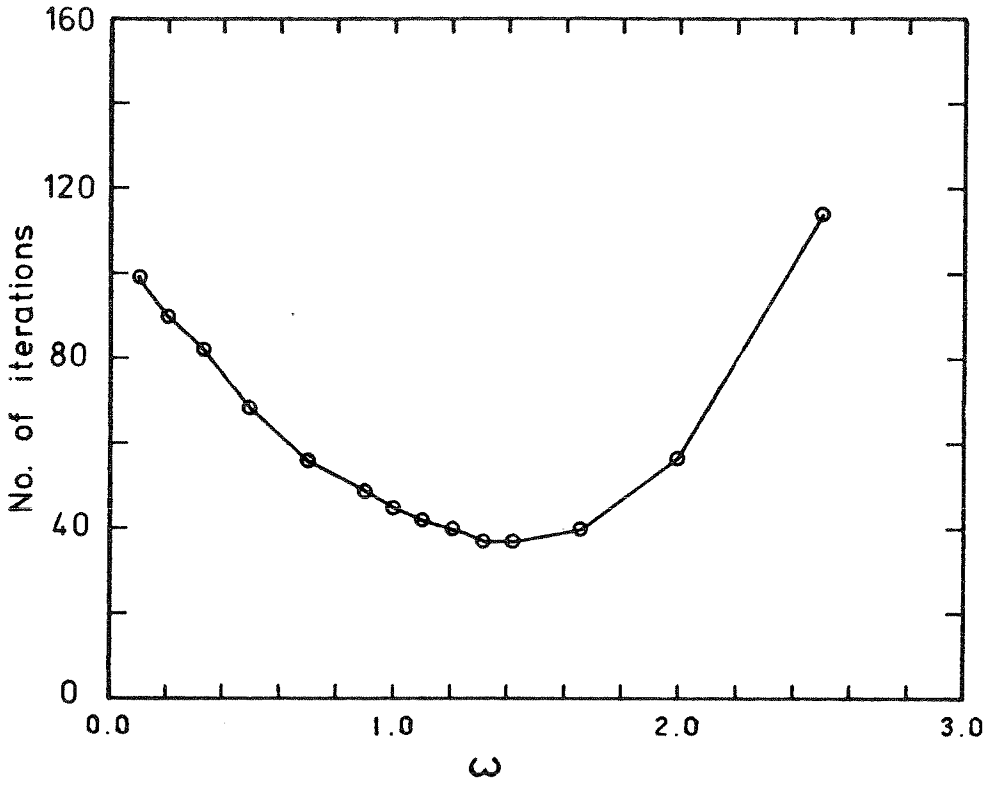


Figure 2. Effect of ω on the Number of Iterations.

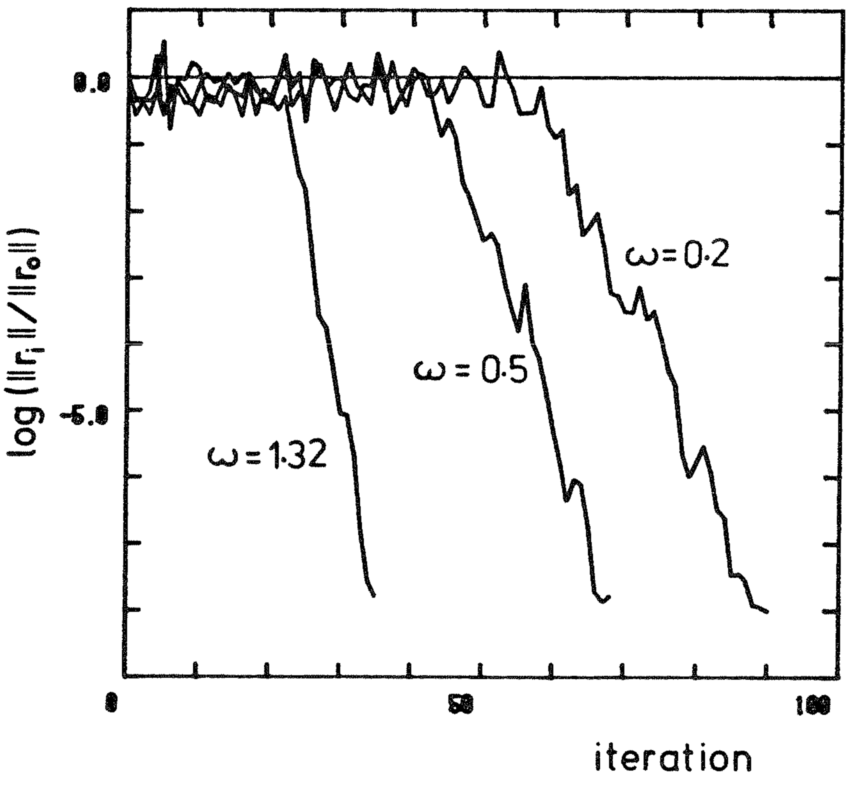


Figure 3. Reduction of the Residual Norm for PCG Iterations.

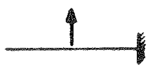
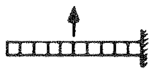
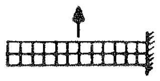
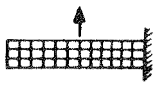
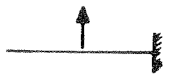
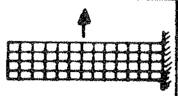
GEOMETRY						
ELEMENT TYPE	10 BEAM ELEMENTS	10 PLANE STRESS ELEMENTS	20 PL. ST. EL.	60 PL. ST. EL.	80 BEAM EL.	120 PL. ST. EL.
NUMBER OF DOFS	30	40	60	160	240	300

Figure 4. Summary of the Set of Examples Chosen for the Static Tests.

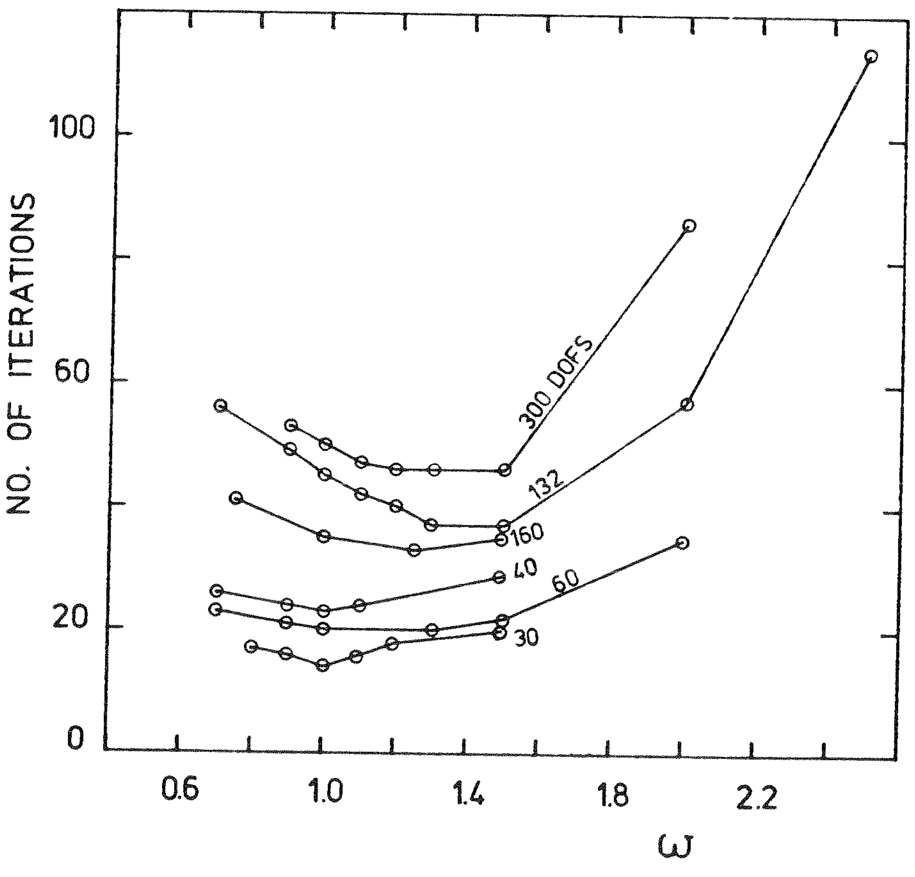


Figure 5. Variation of the Number of Iterations with ω.

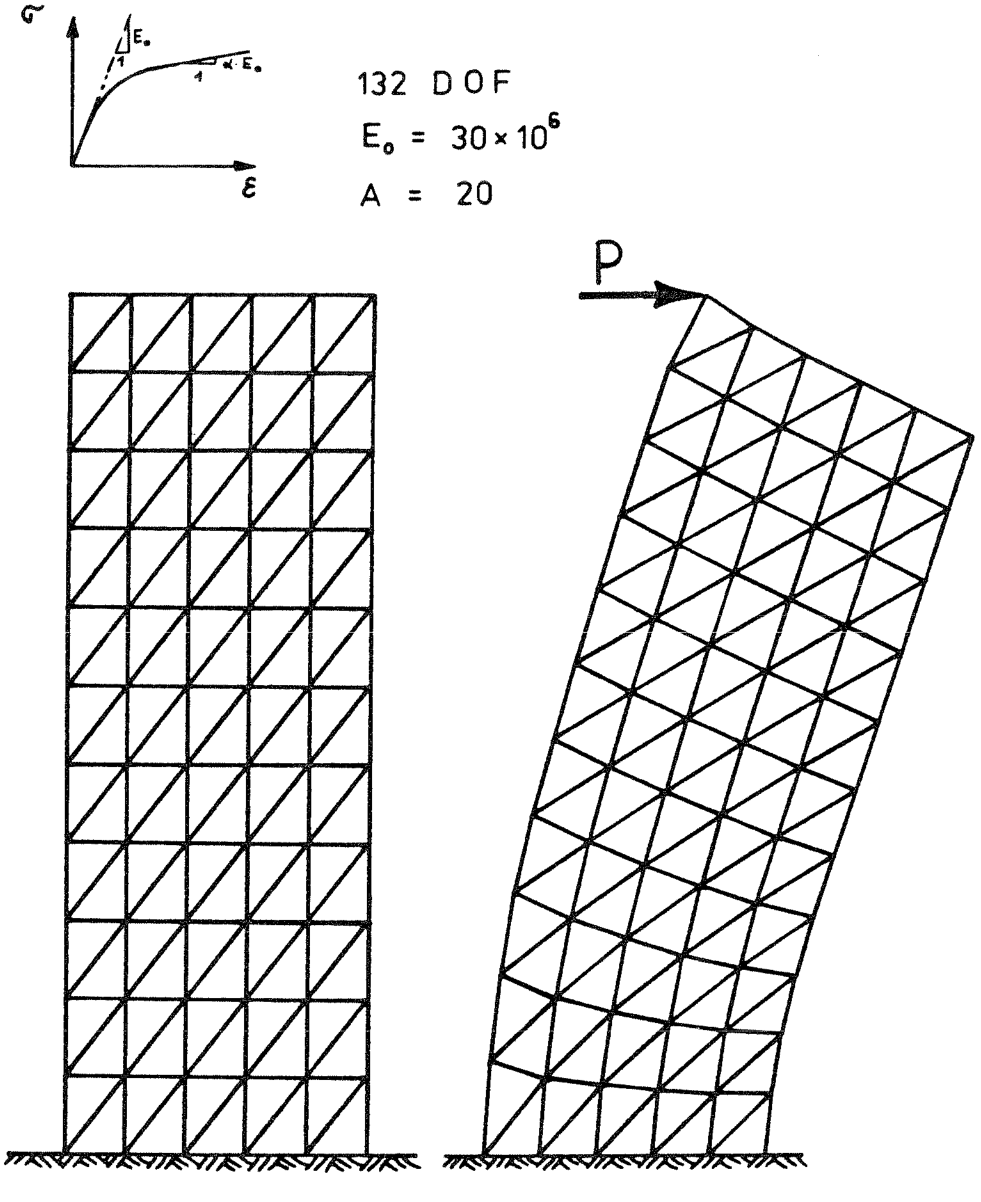


Figure 6. 132 Degrees of Freedom Nonlinear Truss Building.

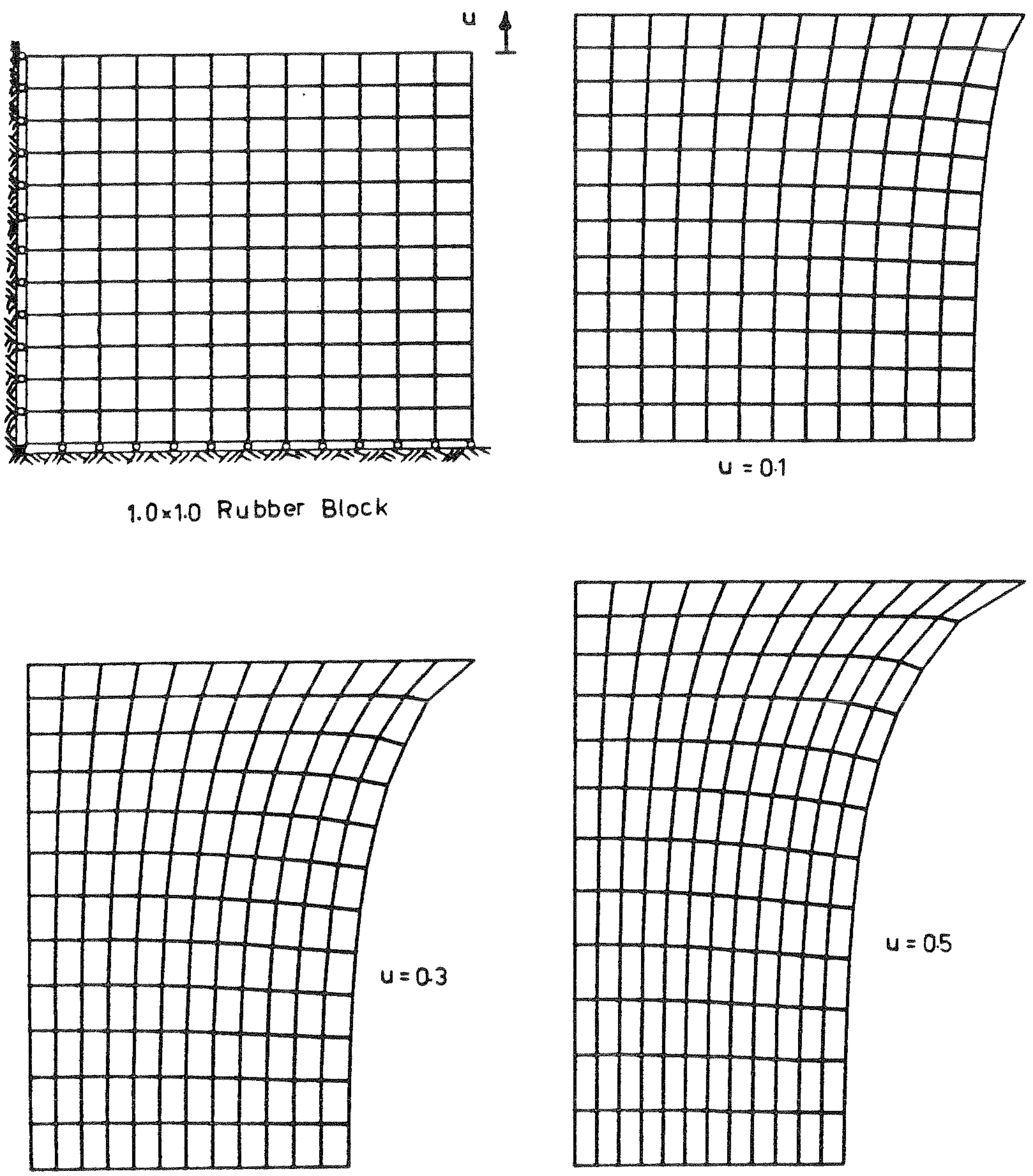


Figure 7. Large Deformation Analysis of the Rubber Block with Mooney-Rivlin Material Model.

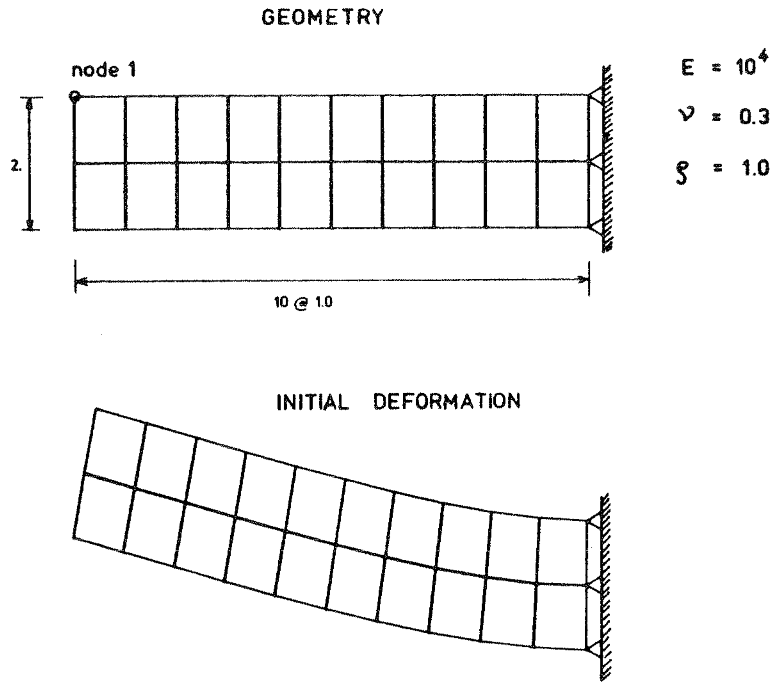


Figure 8. Beam Example Used in Dynamic Analysis.

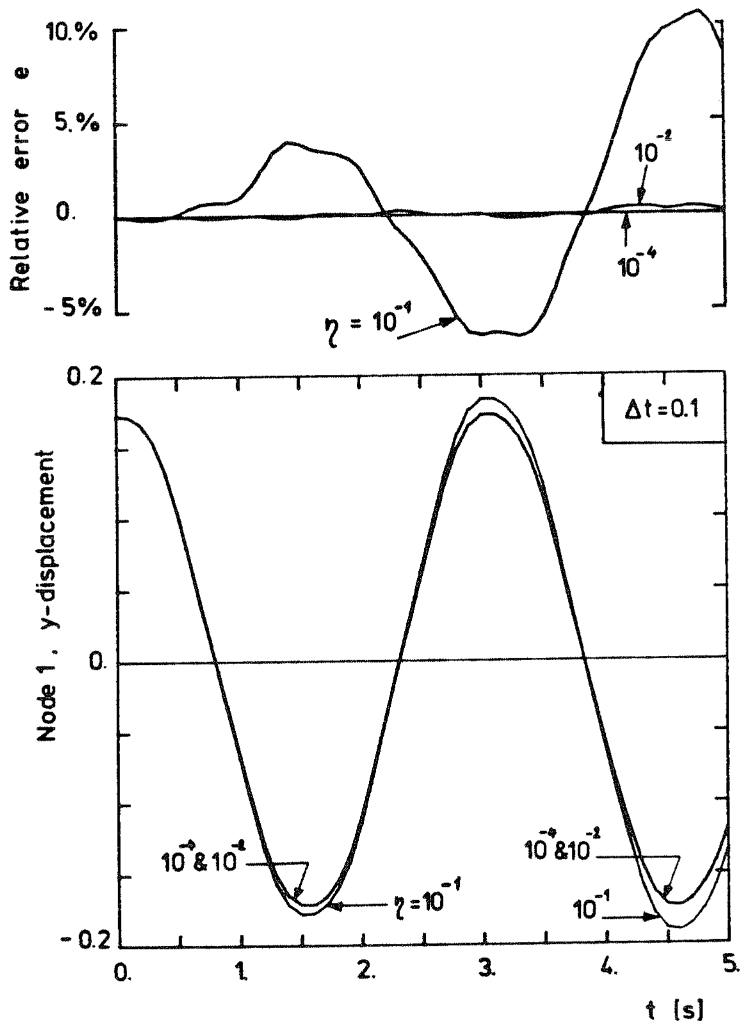


Figure 9. Response of Node 1 with Different PCG Tolerances and Large Time Step.

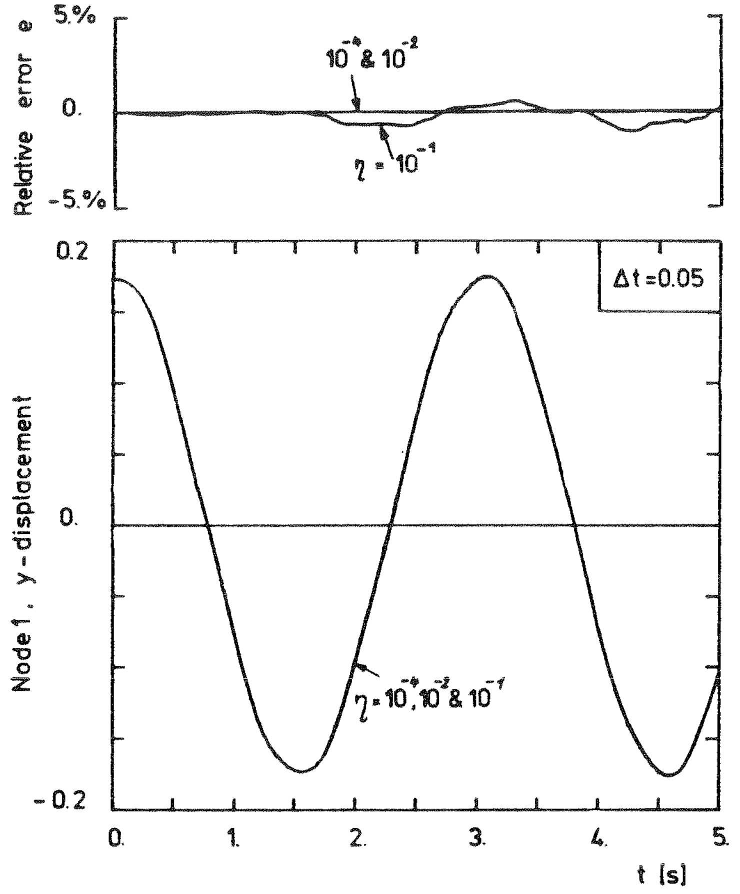


Figure 10. Response of Node 1 with Different PCG Tolerances and Small Time Step.

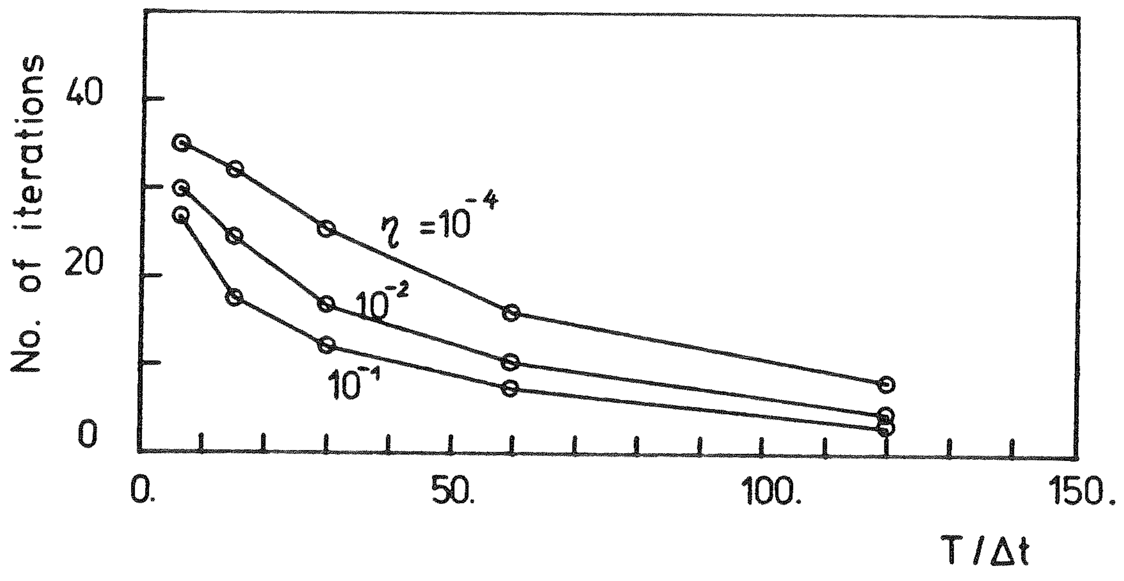


Figure 11. Improvement of Preconditioning Matrix with Reduction in Time Step.

PART II

THICK PLATE ANALYSIS USING VISCOPLASTIC FINITE ELEMENTS

CE-299 Report

Thick Plate Analysis by Viscoplastic Finite Elements

by

Carlos Rodriguez

Department of Civil Engineering
University of California
Berkeley, California 94720

December 1982

Contents

INTRODUCTION

In this paper, we present a formulation for a thick plate finite element with an elastic/viscoplastic material law. The constitutive law adopted is capable of treating material behaviors ranging from elasticity to pure plasticity and/or creep.

After a review of the elastic plate theory, we show that the response of a plate can be decoupled into membrane, plate bending and tranverse shear (section 1). Restricting our attention to problems with no in-plane forces, and assuming a linear elastic behavior for the transverse shears, we develop the viscoplastic law to model the plate bending part only (section 2). The resulting constitutive equation in rate form is integrated using an unconditionally stable generalized mid-point rule and is linearized by a truncated Taylor's expansion. In section 3, we make use of the Principle of Virtual Work to construct a weak form of the equation of motion; the resulting set of nonlinear algebraic equations is solved by a Newton-Raphson iteration scheme at each time step of the solution. We improve the rate of convergence of the Newton-Raphson scheme by forcing the constitutive equation to be satisfied within some very small tolerance, using an inexpensive iteration scheme at the integration point level.

Special attention is devoted to the numerical integration across the depth of the plate. It was expected that Gauss-Lobatto quadrature (with sampling points at the surface and bottom of the plate) would perform better than the usual Gauss-Legendre quadrature; suprisingly, the reverse happens to be true. We propose a slight modification of the Gauss-Legendre quadrature in order to better capture the behavior of the plate close to its collapse load.

In section 4, we present some numerical examples to illustrate the effectiveness of the viscoplastic formulation to simulate quasi-static elastic-perfectly plastic problems.

1. LINEAR ELASTIC PLATE THEORY

1.1 Basic equations

With the assumption of small displacements and deformations 1/ , the linear and angular momentum balance can be written (Mase, 70)

$$\sigma_{ij,j} + \rho b_i = \rho \ddot{u}_i \tag{1a}$$

$$\sigma_{ij} = \sigma_{ji} \tag{1b}$$

where σ_{ij} is the indicial notation of the 2nd order stress tensor, b_i or $\underset{\sim}{b}$ is the body force vector per unit mass, $\ddot{u}_i$ or $\underset{\sim}{\ddot{u}}$ is the acceleration and ρ is the mass density.

A comma denotes partial derivative, i.e.

$$(\phi)_{,i} = \partial\phi/\partial x_i$$

and repeated (dummy) indices is the summation convention:

$$a_i b_i = \sum_{i=1}^{3} a_i b_i = a_1 b_1 + a_2 b_2 + a_3 b_3$$

When neglecting the second order terms, the compatibility relations become

$$\varepsilon_{ij} = \frac{1}{2}\left(u_{i,j} + u_{j,i} - u_{k,i}\, u_{k,j}\right) \doteq \frac{1}{2}\left(u_{i,j} + u_{j,i}\right) \tag{2}$$

where $\underset{\sim}{\varepsilon}$ is the strain tensor and $\underset{\sim}{u}$ the displacement vector.

For linear elastic isotropic material, with no temperature effects, the constitutive relations are

$$\varepsilon_{ij} = \frac{1+\nu}{E}\sigma_{ij} - \frac{\nu}{E}\sigma_{kk}\,\delta_{ij} \tag{3}$$

where E is the Young's modulus, ν the Poisson's ratio and δ_{ij} the Kronnecker delta.

We shall now specialize the previous relations for the thick plate domain by introducing some kinematic and static constraints:

a) Kinematic constraints

(i) normals to the undeformed midplane of the plate remain straight lines after deformation (fig. 1: ab=straight line)

(ii) the distance between two points on a normal remains constant (fig. 1: $\|AB\| = \|ab\|$)

1/ The fundamental laws of continuum mechanics may be stated in either reference or a spatial current configuration. Here, we are only concerned with an infinitesimal deformation theory, hence there is no need to make the distinction between reference and current configurations.

Let $\underset{\sim}{U}o=(Uo,Vo,\ Wo)^T$ be the motion of the point O located at the midplane of the plate (fig. 1) and $\underset{\sim}{\varphi}=(\varphi x,\varphi y,0)^T$ be the rotation of the line initially normal to the midplane. We can represent the motion on any point of the line by

$$\underset{\sim}{u} = \underset{\sim}{u}_0 + z \cdot \underset{\sim}{\varphi} \tag{4}$$

Substitution into eq. (2) leads to

membrane

$$\begin{aligned}
\varepsilon_z &= 0 \\
\varepsilon_x &= \frac{\partial u_0}{\partial x} + z\,\frac{\partial \varphi_x}{\partial x} &&= \varepsilon_{x_0} + z\,\chi_x \\
\varepsilon_y &= \frac{\partial v_0}{\partial y} + z\,\frac{\partial \varphi_y}{\partial y} &&= \varepsilon_{y_0} + z\,\chi_y \\
\gamma_{xy} &= \frac{\partial u_0}{\partial y} + \frac{\partial v_0}{\partial x} + z\left(\frac{\partial \varphi_x}{\partial y} + \frac{\partial \varphi_y}{\partial x}\right) &&= \gamma_{xy_0} + z\,\chi_{xy}
\end{aligned} \tag{5}$$

plate bending

$$\begin{aligned}
\gamma_{xz} &= \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} &&= \varphi_x + w_{,x} \\
\gamma_{yz} &= \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} &&= \varphi_y + w_{,y}
\end{aligned}$$

transverse shear

where $\underset{\sim}{\chi}=(\chi x,\ \chi y,\ \chi xy)$ is the curvature vector, and γ_{ij} is the engineering strain, i.e. $\gamma_{ij} = \varepsilon_{ij} + \varepsilon_{ji} = 2\,\varepsilon_{ij}$
Note that the transverse shears γxz and γyz are independent of z, which contradicts the boundary conditions on the top and bottom surfaces. A shear correction factor $\kappa=5/6$ is usually used to correct this effect, in an energy sense only.

b) <u>Static constraint</u>

Assume that each layer of the plate is in state of <u>plane stress</u> in the plane of the lamina, i.e. σz is so much smaller than the other stresses, that we can neglect it. 1/
The constitutive relations (3) become

$$\begin{aligned}
\varepsilon_x &= \frac{1}{E}\left(\sigma_x - \nu\sigma_y\right) & \gamma_{xy} &= \frac{1}{G}\,\tau_{xy} \\
\varepsilon_y &= \frac{1}{E}\left(\sigma_y - \nu\sigma_x\right) & \gamma_{xz} &= \frac{1}{G}\,\tau_{xz} \\
\varepsilon_z &= \frac{-\nu}{E}\left(\sigma_x + \sigma_y\right) & \gamma_{yz} &= \frac{1}{G}\,\tau_{yz}
\end{aligned} \tag{6}$$

Where G is the shear modulus: $G = E/2(1+\nu)$

Note that $\varepsilon z \neq 0$ is in contradiction with the kinematic constraints (eq. 5); but its effects being small , we will simply ignore it.

1/ For an applied load $\bar{\underset{\sim}{t}}$, we have $\sigma_z = \bar{\underset{\sim}{t}}\cdot n_z$; that means that the magnitude of the applied load must be much smaller than the norms of the stresses $\sigma x, \sigma y, \tau xy,\ \tau yz$ and τxz.

Inverting relations (6), we have in matrix form:

$$\left\{\begin{array}{c} \sigma_x \\ \sigma_y \\ \tau_{xy} \\ \hdashline \tau_{xz} \\ \tau_{yz} \end{array}\right\} = \frac{E}{1-\nu^2} \left[\begin{array}{ccc:cc} 1 & \nu & 0 & & \\ \nu & 1 & 0 & \multicolumn{2}{c}{\underset{\sim}{0}} \\ 0 & 0 & \frac{1-\nu}{2} & & \\ \hdashline & & & \frac{1-\nu}{2} & 0 \\ & \underset{\sim}{0} & & 0 & \frac{1-\nu}{2} \end{array}\right] \cdot \left\{\begin{array}{c} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \\ \hdashline \gamma_{xz} \\ \gamma_{yz} \end{array}\right\} \qquad (7) \qquad \begin{array}{c} \text{plate bending} \\ \text{and} \\ \text{membrane} \\ \uparrow \\ \hdashline \downarrow \\ \text{transverse} \\ \text{shear} \end{array}$$

Before formulating the viscoplastic behavior, we shall take advantage of the splitting of the response of the plate, namely membrane, plate bending and transverse shear. This is done next. Then the viscoplatic description (section 2) and finite element discretization (section 3) will complete our formulation.

1.2 Uncoupling

Looking at equations (5) and (7), we see that the behavior of the plate is influenced by three different terms:

(a) membrane effects (in plane force resultants only)
(b) plate bending effects
(c) transverse shear effects.

It appears then natural to try to uncouple these effects, or at least to find the bounds between which they uncouple.

Let us look at the internal Virtual Work:

$$VW_I = \int_\Omega \delta \underset{\sim}{\varepsilon}^T \cdot \underset{\sim}{\sigma} \, d\Omega$$

where $\delta \underset{\sim}{\varepsilon}^T$ is an admissible virtual strain field and $\underset{\sim}{\sigma}$ the real stress field.

Using for the virtual strains the same splitting as for the real strains , we can use equation (5) and write

$$VW_I = \int_\Omega \langle \delta\varepsilon_{x_0}, \delta\varepsilon_{y_0}, \delta\gamma_{xy_0} \rangle \cdot \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} d\Omega \qquad \text{(8a) "membrane"}$$

$$+ \int_\Omega \langle \delta\chi_x, \delta\chi_y, \delta\chi_{xy} \rangle \cdot \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} \cdot z \cdot d\Omega \qquad \text{(8b) "plate bending"}$$

$$+ \int_\Omega \langle \delta\gamma_{xz}, \delta\gamma_{yz} \rangle \cdot \begin{Bmatrix} \tau_{xz} \\ \tau_{yz} \end{Bmatrix} d\Omega \qquad \text{(8c) "transverse shear"}$$

(a) "Membrane" part

Using the membrane part of the constitutive equation (7) and the compatibility relations (5), the membrane part of the internal virtual work is

$$(8a) = \int_\Omega \langle \delta\varepsilon_{x_0}, \delta\varepsilon_{y_0}, \delta\gamma_{xy_0} \rangle \cdot \underset{\sim}{D} \cdot \begin{Bmatrix} \varepsilon_{x_0} + z\,\chi_x \\ \varepsilon_{y_0} + z\,\chi_y \\ \gamma_{xy_0} + z\,\chi_{xy} \end{Bmatrix} \cdot d\Omega \qquad (9)$$

$$\text{where } \underset{\sim}{D} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \qquad (10)$$

now split the integral:

$$\int_\Omega (\bullet)\, d\Omega = \int_A \int_{-h/2}^{h/2} (\bullet)\, dz \; dA$$

where A= domain of Ω where z=0.
Because we integrate between $-\frac{h}{2}$ and $+\frac{h}{2}$, the terms with z in equation (9) will cancel.

$$(8a) = \int_{\Omega} \langle \delta \varepsilon_{x_0}, \delta\varepsilon_{y_0}, \delta\gamma_{xy_0} \rangle \cdot \underset{\sim}{D} \cdot \begin{Bmatrix} \varepsilon_{x_0} \\ \varepsilon_{y_0} \\ \gamma_{xy_0} \end{Bmatrix} \qquad (11)$$

that is only function of Uo, Vo. This implies that the membrane term fully uncouples from the plate bending and the transverse shear. Hence, it can be treated separately. This is true only if the function $(\bullet)$is odd in z, that is only if $\underset{\sim}{D}$ is symmetric. Practically, the membrane term will uncouple only for (i) linear elastic problems or (ii) nonlinear problems where $\underset{\sim}{D}(z)$ is symmetric (i.e. no in-plane forces). For the purpose of this study, we will restrict our attention to problems with no in-plane forces, hence we can drop the membrane term.

(b) "Plate bending" part

Using (7) and (5), the plate bending part of the virtual work is

$$(8b) = \int_{\Omega} \langle \delta\chi_x, \delta\chi_y, \delta\chi_{xy} \rangle \cdot \underset{\sim}{D} \cdot z \cdot \begin{Bmatrix} \varepsilon_{x_0} + z\,\chi_x \\ \varepsilon_{y_0} + z\,\chi_y \\ \gamma_{xy_0} + z\,\chi_{xy} \end{Bmatrix} d\Omega$$

again, splitting the integral and cancelling terms in z, we are left with

$$(8b) = \int_{\Omega} \langle \delta\chi_x, \delta\chi_y, \delta\chi_{xy} \rangle \cdot \underset{\sim}{D} \cdot z^2 \cdot \begin{Bmatrix} \chi_x \\ \chi_y \\ \chi_{xy} \end{Bmatrix} d\Omega \qquad (12)$$

where $\underset{\sim}{D}$=plane stress elasticity matrix (equ. 10)

(c) "Transverse shear" part

Similarly:
$$(8c) = \int_{\Omega} \langle \delta\gamma_{xz}, \delta\gamma_{yz} \rangle \cdot \underset{\sim}{G} \cdot \begin{Bmatrix} \gamma_{xz} \\ \gamma_{yz} \end{Bmatrix} d\Omega \qquad (13a)$$

$$\underset{\sim}{G} = \frac{E \cdot \kappa}{2(1+\nu)} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad (13b)$$

where κ=5/6 is the shear correction factor.

For linear elastic materials, we could directly go to the external virtual work and construct the finite element discretization. Our purpose is now to introduce the viscoplastic behavior. Before doing so, however, we must say a few comments on the transverse shear part.

"Locking" phenomenon

It has been known for some time that the stiffness resulting from a thick plate theory is dominated by the transverse shear term when the aspect ratio of the element (length/thickness) becomes large (Hughes et al., 77); the corresponding stiffness is overestimated and the ratio

$\hat{\phi} / \phi_{th}$ tends to zero.

where $\hat{\phi}$ = finite element displacements
ϕ_{th} = displacements resulting from the Kirchhoff plate theory.
In this study, we shall use the element proposed by Hughes (Hughes and Tezduyar, 80) which permits a very large aspect ratio to be used before the element becomes inaccurate (due to round-off errors).

2. VISCOPLASTIC BEHAVIOR

2.1 Viscoplastic model

With the assumption that the total strain rate tensor $\dot{\underset{\sim}{\varepsilon}}$ is the sum of an elastic and a viscoplatic parts, i.e.

$$\dot{\underset{\sim}{\varepsilon}} = \dot{\underset{\sim}{\varepsilon}}^{e} + \dot{\underset{\sim}{\varepsilon}}^{vp} \qquad (14)$$

Classical plasticity theory postulates the existence of a yield function F $(\dot{\underset{\sim}{\sigma}}, \underset{\sim}{q})$, where $\dot{\underset{\sim}{\sigma}}$ is the stress rate and $\underset{\sim}{q}$ some internal variables, such that:

$$F(\dot{\underset{\sim}{\sigma}}, \underset{\sim}{q}) < 0 \Rightarrow \dot{\underset{\sim}{\varepsilon}}^{vp} = \underset{\sim}{0} \qquad \text{elastic behavior}$$

$$F(\dot{\underset{\sim}{\sigma}}, \underset{\sim}{q}) \geqslant 0 \Rightarrow \dot{\underset{\sim}{\varepsilon}}^{vp} \neq \underset{\sim}{0} \qquad \text{has a plastic flow}$$

Introducing a MacCauley type operator

$$\begin{aligned} \langle f(F) \rangle &= 0 && \text{if} \quad F < 0 \\ \langle f(F) \rangle &= f(F) && \text{if} \quad F \geq 0 \end{aligned} \qquad (15)$$

and the usual normality rule, we can write (Naghdi, 60):

$$\dot{\underset{\sim}{\varepsilon}}^{vp} = \gamma \cdot \left\langle \Phi\left(\frac{F}{F_0}\right)\right\rangle \cdot \frac{\partial Q}{\partial \underset{\sim}{\sigma}} \qquad (16)$$

where γ = material parameter
$\Phi(F/F_0)$ = flow function
Q = viscoplastic potential
Substituting eq. (16) into eq. (14), we get for the general case:

$$\dot{\underset{\sim}{\varepsilon}} = \underset{\sim}{C}\,\dot{\underset{\sim}{\sigma}} + \gamma \cdot \left\langle \Phi\left(\frac{F}{F_0}\right)\right\rangle \cdot \frac{\partial Q}{\partial \underset{\sim}{\sigma}} \tag{17}$$

where $\underset{\sim}{C}$ is the elastic compliance matrix and $\underset{\sim}{C}\dot{\underset{\sim}{\sigma}} = \dot{\underset{\sim}{\varepsilon}}^e$ the elastic part of eq. (14).
For associated plasticity (Q = F), it can be shown that the resulting viscoplastic tangent matrix is symmetric, which is always preferable (Zienkiewicz, 77). For metal plasticity, the Huber-vonMises yield surface is often used

$$Q = F = \sqrt{3J_2} - \sigma_Y \tag{18}$$

where $J_2 = \frac{1}{2}\sigma'_{ij}\sigma'_{ij}$ = 2nd deviatoric stress invariant
σ_Y = uniaxial yield stress. 1/

We will also choose a plastic flow coefficient of the form

$$\Phi\left(\frac{F}{F_0}\right) = \left(\frac{F}{\sigma_Y}\right)^k \tag{19}$$

where $k \geq 1$ is a parameter to be chosen.
Substituting eq. (18) into eq. (19) leads to

$$\left\langle \Phi\left(\frac{F}{F_0}\right)\right\rangle = \left\langle \left(\frac{\sqrt{3J_2}}{\sigma_Y} - 1\right)^k \right\rangle \tag{20}$$

Where the MacCauley operator must be evaluated before performing the exponentiation.

Remarks : 1) due to the MacCauley operator, we don't have a smooth function; this may slow down the rate of convergence of an iterative process.
2) constant stress states outside the yield surface are allowed, but only with non-zero rates of strain (fig. 2). For quasi-static loads we get an elastic-perfectly plastic response; for high rates (or for σ_Y appropriately high) we get an elastic response.
3) the flow law selected (eq. 19) overestimates the apparent yield stress for high strain rates (Perzyna, 66).

1/ We implicitly assume that we have no hardening in our formulation, i.e. σ_Y =constant.

2.2 Specialization of the viscoplastic model to thick plates

Recalling the restrictions laid out in section 1.2, we consider a viscoplastic behavior for the plate bending part only. The transverse shear term will always be treated elastically, hence $\underset{\sim}{G}$ =constant in eq. (13a) and (13b). We expect that the good behavior of the element at the thin plate limit will be preserved.

With the yield surface (eq. 18) and the flow law (eq. 19), we are able to compute the different terms of the constitutive equation (eq. 17). The term $\partial Q/\partial \underset{\sim}{\sigma}$ becomes

$$\frac{\partial Q}{\partial \sigma_{kl}} = \frac{\partial Q}{\partial \sigma'_{ij}} \cdot \frac{\partial \sigma'_{ij}}{\partial \sigma_{kl}} \qquad \text{(where } \sigma'_{ij} = \text{deviatoric stresses)}$$

$$= \left(\frac{\sqrt{3}}{2 J_2} \cdot \sigma'_{ij}\right) \cdot \left(\delta_{ik}\delta_{jl} - \frac{1}{3}\delta_{ij}\delta_{kl}\right)$$

Using the relation $\sigma'_{ii} = 0$, we deduce that

$$\frac{\partial Q}{\partial \sigma_{kl}} = \frac{\sqrt{3}}{2 \cdot \sqrt{J_2}} \cdot \sigma'_{kl} \qquad (21)$$

In each lamina, the stress tensor can be replaced by its corresponding vector form (eq. 7, plate bending part only). When using this vector notation for quantities which are 2nd order tensor (i.e. $\underset{\sim}{\sigma}$ and $\underset{\sim}{\varepsilon}$) we can substitute eq. (20) and (21) into eq. (17) to write the rate form of the constitutive equation:

$$\dot{\underset{\sim}{\varepsilon}} = \underset{\sim}{C}\,\dot{\underset{\sim}{\sigma}} + \frac{1}{\sqrt{3J_2}} \cdot \gamma \cdot \left\langle \left(\frac{\sqrt{3J_2}}{\sigma_Y} - 1\right)^k \right\rangle \cdot \underset{\sim}{M} \cdot \underset{\sim}{\sigma} \qquad (22)$$

where

$$\dot{\underset{\sim}{\varepsilon}} = \begin{Bmatrix} \dot{\varepsilon}_x \\ \dot{\varepsilon}_y \\ \dot{\gamma}_{xy} \end{Bmatrix}, \quad \dot{\underset{\sim}{\sigma}} = \begin{Bmatrix} \dot{\sigma}_x \\ \dot{\sigma}_y \\ \dot{\tau}_{xy} \end{Bmatrix} \quad \text{and} \quad \underset{\sim}{\sigma} = \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} \qquad (23)$$

are the total strain rate, stress rate and stress vectors, respectively. The 2nd invariant of the deviatoric stresses becomes:

$$J_2 = \frac{1}{3}\left(\sigma_x^2 + \sigma_y^2 - \sigma_x\sigma_y\right) + \tau_{xy}^2$$

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

$$\underset{\sim}{M} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & 0 \\ 0 & 0 & 6 \end{bmatrix} \qquad (24)$$

and $\underset{\sim}{C}$ is the elastic compliance matrix:

$$\underset{\sim}{C} = \frac{1}{E}\begin{bmatrix} 1 & -\nu & 0 \\ -\nu & 1 & 0 \\ 0 & 0 & 2(1+\nu) \end{bmatrix} \qquad \text{and} \quad \underset{\sim}{C} = \underset{\sim}{D}^{-1} \tag{25}$$

It is worth noting that the shear strain γ_{xy} in eq. (23) is the usual engineering strain, i.e. $\gamma_{xy} = 2\varepsilon_{xy}$; the $\underset{\sim}{M}$ and $\underset{\sim}{C}$ matrices reflect this choice.

For each point in the plate, equation (22) is the constitutive relation for the plate bending part only. The transverse shear always remains elastic and is given by

$$\begin{Bmatrix} \gamma_{xz} \\ \gamma_{yz} \end{Bmatrix} = \frac{2(1+\nu)}{E \cdot \kappa} \cdot \begin{Bmatrix} \tau_{xz} \\ \tau_{yz} \end{Bmatrix} = \frac{1}{G'} \cdot \begin{Bmatrix} \tau_{xz} \\ \tau_{yz} \end{Bmatrix} \tag{26}$$

where $\kappa = 5/6$ is the shear correction factor.

Equations (22) and (26) fully describe the response of the plate. In the sequel, we will not discuss the elastic transverse shear since it is identical to (Hughes and Tezduyar, 80), but concentrate on the bending part only.

Remark: Note that the ratio -M (1,2)/M (1,1) = 0.5, which is Poisson's ratio for the plastic deformation; it clearly shows that we have an incompressible behavior during plastic flow.

2.3 Integration of the rate form

For computational purposes, the constitutive equation (22) is integrated over a time step.

Let us define
$$OP = \frac{1}{\sqrt{3J_2}} \cdot \left\langle \left(\frac{\sqrt{3J_2}}{\sigma_Y} - 1 \right)^k \right\rangle \tag{27}$$

We can then rewrite eq. (22) as

$$\dot{\underset{\sim}{\varepsilon}} = \underset{\sim}{C}\dot{\underset{\sim}{\sigma}} + \gamma \cdot OP \cdot \underset{\sim}{M} \cdot \underset{\sim}{\sigma} \tag{28}$$

Suppose that the solution at time t_n is known, we can integrate eq. (28) over a time step to obtain the solution at time t_{n+1}

$$\int_{t_n}^{t_{n+1}} \left(\dot{\underset{\sim}{\varepsilon}} - \underset{\sim}{C}\dot{\underset{\sim}{\sigma}} - \gamma \cdot OP \cdot \underset{\sim}{M} \cdot \underset{\sim}{\sigma} \right) dt = 0$$

or

$$\left(\underset{\sim}{\varepsilon}_{n+1} - \underset{\sim}{\varepsilon}_n\right) - \underset{\sim}{C} \cdot \left(\underset{\sim}{\sigma}_{n+1} - \underset{\sim}{\sigma}_n\right) - \gamma \cdot \underset{\sim}{M} \cdot \int_{t_n}^{t_{n+1}} OP \cdot \underset{\sim}{\sigma} \, dt = 0 \tag{29}$$

Where the subscript n indicates the "nth" time step. The last term of eq. (29) causes some difficulties. Since both OP and $\underset{\sim}{\sigma}$ are function of time, this term cannot, in general, be integrated exactly. However, using the mean value theorem, we can write

$$\left(\underset{\sim}{\varepsilon}_{n+1}-\underset{\sim}{\varepsilon}_{n}\right)-\underset{\sim}{C}\cdot\left(\underset{\sim}{\sigma}_{n+1}-\underset{\sim}{\sigma}_{n}\right)-\gamma\cdot\Delta t\cdot\left(OP\cdot\underset{\sim}{\sigma}\right)_{t_{n+\alpha}} = 0 \qquad (30)$$

where $\Delta t=t_{n+1}-t_{n}$ is the time step and $(OP.\underset{\sim}{\sigma})_{t_{n+\alpha}}$ means that both OP and $\underset{\sim}{\sigma}$ are evaluated at some time $t=t_{n+\alpha}$, $t_{n+\alpha}\in\left[t_n,t_{n+1}\right]$.

While the two first bracket terms of eq. (30) are easy to handle, the last term needs a special treatment because $t_{n+\alpha}$ is unknown. Assuming that the stress $\underset{\sim}{\sigma}$ has a smooth variation between t_n and t_{n+1} , we can use the generalized mid-point rule proposed by (Hughes and Taylor, 78) to describe its variation:

$$\underset{\sim}{\sigma}_{n+\alpha}=(1-\alpha)\underset{\sim}{\sigma}_{n}+\alpha\cdot\underset{\sim}{\sigma}_{n+1} \qquad 0\leq\alpha\leq 1 \qquad (31)$$

Once the stresses are defined, the variable OP is straightforwardly computed (eq. 27). This one-parameter family of algorithms is consistent for the given range of α, and has been shown to be unconditionally stable for $0.5\leq\alpha\leq 1.0$; α=0.5 is known as Crank-Nicholson (or mid-point) rule, α=1.0 as Euler backward difference (or implicit) and α=0 gives an explicit algorithm (also known as the "initial stress" technique (Zienkiewicz, 77)).

At this stage, we have fully described the computational form for the constitutive relation (eq. 30). However, this equation is nonlinear and requires linearization; its solution will then lead to an iterative scheme.

2.4 Linearization of the constitutive equation

To linearize the constitutive equation (30), we shall use a Taylor series in both $\underset{\sim}{\varepsilon}$ and $\underset{\sim}{\sigma}$ (this method is fully explained in (Hughes and Pister, 78)). Let us first rewrite eq. (30) introducing the function $\underset{\sim}{\phi}$:

$$\underset{\sim}{\phi}\left(\underset{\sim}{\varepsilon}_{n+1},\underset{\sim}{\sigma}_{n+1}\right)=\left(\underset{\sim}{\varepsilon}_{n+1}-\underset{\sim}{\varepsilon}_{n}\right)-\underset{\sim}{C}\left(\underset{\sim}{\sigma}_{n+1}-\underset{\sim}{\sigma}_{n}\right)-\gamma\Delta t\,\underset{\sim}{M}\left(OP\cdot\underset{\sim}{\sigma}\right)_{t_{n+\alpha}} = 0 \qquad (32)$$

we can write the first few terms of the Taylor's series:

$$\underset{\sim}{\phi}\left(\underset{\sim}{\varepsilon}_{n+1}+\Delta\underset{\sim}{\varepsilon},\underset{\sim}{\sigma}_{n+1}+\Delta\underset{\sim}{\sigma}\right)=\underset{\sim}{\phi}\left(\underset{\sim}{\varepsilon}_{n+1},\underset{\sim}{\sigma}_{n+1}\right)+\frac{\partial\underset{\sim}{\phi}}{\partial\underset{\sim}{\varepsilon}_{n+1}}\cdot\Delta\underset{\sim}{\varepsilon}+\frac{\partial\underset{\sim}{\phi}}{\partial\underset{\sim}{\sigma}_{n+1}}\cdot\Delta\underset{\sim}{\sigma}+\theta(m) \qquad (33)$$

where the term $\theta\,(\Delta\underset{\sim}{\varepsilon}^{m},\Delta\underset{\sim}{\sigma}^{m})$, with m > 1, regroups the higher order terms of the series.

The first three terms of eq. (33) defines the linear part of $\underset{\sim}{\phi}$. Setting the linear part to zero defines a Newton-Raphson iterative method for solving eq. (32).

Accordingly, let i be the iteration number of the Newton-Raphson scheme, then eq. (33) reads

$$\underset{\sim}{\phi}^{i}(\underset{\sim}{\varepsilon}^{i}_{n+1}, \underset{\sim}{\sigma}^{i}_{n+1}) + \frac{\partial \underset{\sim}{\phi}^{i}}{\partial \underset{\sim}{\varepsilon}^{i}_{n+1}} \Delta \underset{\sim}{\varepsilon}^{i}_{n+1} + \frac{\partial \underset{\sim}{\phi}^{i}}{\partial \underset{\sim}{\sigma}^{i}_{n+1}} \Delta \underset{\sim}{\sigma}^{i}_{n+1} = 0 \qquad (34)$$

with

$$\Delta \underset{\sim}{\varepsilon}^{i}_{n+1} = \underset{\sim}{\varepsilon}^{i+1}_{n+1} - \underset{\sim}{\varepsilon}^{i}_{n+1}$$
$$\Delta \underset{\sim}{\sigma}^{i}_{n+1} = \underset{\sim}{\sigma}^{i+1}_{n+1} - \underset{\sim}{\sigma}^{i}_{n+1} \qquad (35)$$

At the beginning of a new step (i=0), the best guess for the initial values are those of the converged previous step, i.e.

$$\underset{\sim}{\varepsilon}^{0}_{n+1} = \underset{\sim}{\varepsilon}_{n} \quad \text{and} \quad \underset{\sim}{\sigma}^{0}_{n+1} = \underset{\sim}{\sigma}_{n}$$

The partial derivatives of eq. (34) are:

$$\frac{\partial \underset{\sim}{\phi}}{\partial \underset{\sim}{\varepsilon}_{n+1}} = I \quad \text{the identity matrix} \qquad (36)$$

$$\frac{\partial \underset{\sim}{\phi}}{\partial \underset{\sim}{\sigma}_{n+1}} = -\underset{\sim}{C} - \gamma \cdot \Delta t \left(OP \cdot \underset{\sim}{M} \cdot \frac{\partial \underset{\sim}{\sigma}_{n+\alpha}}{\partial \underset{\sim}{\sigma}_{n+1}} + \underset{\sim}{M} \cdot \underset{\sim}{\sigma}_{n+\alpha} \cdot \frac{\partial\, OP}{\partial \underset{\sim}{\sigma}_{n+1}} \right) \qquad (37)$$

where

$$\frac{\partial \underset{\sim}{\sigma}_{n+\alpha}}{\partial \underset{\sim}{\sigma}_{n+1}} = \alpha \qquad (38)$$

$$\frac{\partial\, OP}{\partial \underset{\sim}{\sigma}_{n+1}} = \alpha \cdot \left\{ \frac{1}{3J_2} \left\langle \left(\frac{\sqrt{3J_2}}{\sigma_Y} - 1 \right)^{k-1} \right\rangle \cdot \left(\frac{k-1}{\sigma_Y} + \frac{1}{\sqrt{3J_2}} \right) \right\} \cdot \underset{\sim}{\sigma}^{T}_{n+\alpha} \cdot \underset{\sim}{M}^{T}$$

$$\frac{\partial\, OP}{\partial \underset{\sim}{\sigma}_{n+1}} = \alpha \cdot OP^{*} \cdot \underset{\sim}{\sigma}^{T}_{n+\alpha} \cdot \underset{\sim}{M}^{T} \qquad (39)$$

where $OP^{*} = \frac{1}{3J_2} \cdot \left\langle \left(\frac{\sqrt{3J_2}}{\sigma_Y} - 1 \right)^{k-1} \right\rangle \cdot \left(\frac{k-1}{\sigma_Y} + \frac{1}{\sqrt{3J_2}} \right) \qquad (40)$

Again, the MacCauley operator must be evaluated before performing the exponentiation.

Substituting eq. (36) through (39) into eq. (34) leads to the <u>incremental iterative constitutive equation</u>:

$$\underset{\sim}{\phi}^{i}(\underset{\sim}{\varepsilon}^{i}_{n+1}, \underset{\sim}{\sigma}^{i}_{n+1}) + \Delta \underset{\sim}{\varepsilon}^{i}_{n+1} - \underset{\sim}{C}^{i}_{n+\alpha} \cdot \Delta \underset{\sim}{\sigma}^{i}_{n+1} = 0 \qquad (41)$$

where $\underset{\sim}{C}^{i}_{n+\alpha}$ is the <u>tangent compliance matrix</u> (fully populated and symmetric) given by

$$\underset{\sim}{C}^{i}_{n+\alpha} = \underset{\sim}{C} + \gamma \Delta t \cdot \alpha \cdot \left\{ OP \cdot \underset{\sim}{M} + OP^{*} \cdot \underset{\sim}{M}\, \underset{\sim}{\sigma}\, \underset{\sim}{\sigma}^{T} \underset{\sim}{M}^{T} \right\}_{t_{n+\alpha}} \qquad (42)$$

and $\underset{\sim}{D}^{i}_{n+\alpha} = \left(\underset{\sim}{C}^{i}_{n+\alpha}\right)^{-1}$ is the tangent modulus matrix.
The constitutive equation (41) is satisfied when $\underset{\sim}{\phi}^{i} = \underset{\sim}{0}$.

We now have all the necessary relations to construct a finite element approximation for the plate problem.

3. FINITE ELEMENT DISCRETIZATION

We shall not discuss in detail the finite element method, which is described in many books (e.g. (Zienkiewicz, 77), (Bathe and Wilson, 76), etc.); the purpose of this section is to describe the _plate bending part of the response only_, being understood that the transverse part is treated elastically for all stress-strain states (its description can be found in detail in (Hughes and Tezduyar, 80)). The final result will be a displacement formulation for the viscoplastic plate element.

3.1 _Interpolation functions_

The element we shall use to implement our viscoplastic behavior is a _4-node isoparametric element_ (fig. 3) with three nodal parameters at each node: $(\varphi_x, \varphi_y, w)$, i.e. two slopes and the deflection. 1/

The displacement vector of any point inside each element is approximated by

$$\begin{Bmatrix} \varphi_x \\ \varphi_y \\ w \end{Bmatrix} = \sum_{I=1}^{NEL} N_I(\xi,\eta) \cdot \begin{Bmatrix} \varphi_x \\ \varphi_y \\ w \end{Bmatrix}_I = N_I \cdot \underset{\sim}{\varphi}_I \qquad (43)$$

where I = element node
NEL = number of nodes of the element (here NEL=4)
N_I = shape function
ξ, η =local coordinates (see fig. 3)

For the 4-node element, _the shape function is bilinear_ and is given by

$$N_I = \frac{1}{4}\left(1+\xi\cdot\xi_I\right)\cdot\left(1+\eta\cdot\eta_I\right)$$

where ξ_I and η_I are the local coordinates of the node I.

1/ The element is implemented in the FEAP program described in (Zienkiewicz, 77), chapter 24.

Looking only at the plate bending part, we have from eq. (5)

$$\underset{\sim}{\varepsilon} = z \cdot \underset{\sim}{\chi} = z \cdot \begin{Bmatrix} \varphi_{x,x} \\ \varphi_{y,y} \\ \varphi_{x,y} + \varphi_{y,x} \end{Bmatrix}$$

Substituting the values of $\underset{\sim}{\varphi}$ given by eq. (43), we obtain

$$\begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} = z \cdot \begin{bmatrix} N_{I,x} & 0 & 0 \\ 0 & N_{I,y} & 0 \\ N_{I,y} & N_{I,x} & 0 \end{bmatrix} \cdot \begin{Bmatrix} \varphi_x \\ \varphi_y \\ w \end{Bmatrix}_I \qquad (44)$$

Note that there is an implied sum due to the repeated index I (sum over the nodes).
Due to the decoupling of the plate bending from the transverse shear, we have zeros in the third column of the above matrix. Hence, we can drop it along with the deflection w :

$$\begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} = z \cdot \begin{bmatrix} N_{I,x} & 0 \\ 0 & N_{I,y} \\ N_{I,y} & N_{I,x} \end{bmatrix} \cdot \begin{Bmatrix} \varphi_x \\ \varphi_y \end{Bmatrix}_I = z \cdot \underset{\sim}{B}_I \cdot \underset{\sim}{\varphi}_I \qquad (45)$$

where $\underset{\sim}{B}_I$ (x,y) is the strain-displacement matrix for the node I.

3.2 Finite element equations

Using the relation (45), valid for the plate bending response, the internal virtual work can be expressed by

$$W_{int} = \int_{\Omega_e} \delta \underset{\sim}{\varepsilon}^T \cdot \underset{\sim}{\sigma} \, d\Omega = \int_{\Omega_e} \delta \underset{\sim}{\varphi}^T \cdot \underset{\sim}{B}^T \cdot z \cdot \underset{\sim}{\sigma} \, d\Omega \qquad (46)$$

where Ω_e represents the element volume.

Taking the virtual nodal values out of the integral and linearizing the stresses into known and unknown parts at a given iteration i and time t_{n+1} , eq. (46) becomes

$$W_{int} = \delta \underset{\sim}{\varphi}^T \cdot \int_{\Omega_e} \underset{\sim}{B}^T \cdot z \cdot \left(\underset{\sim}{\sigma}^i_{n+1} + \Delta \underset{\sim}{\sigma}^i_{n+1} \right) d\Omega \qquad (47)$$

which can be written

$$W_{int} = \delta \underset{\sim}{\varphi}^T \cdot \underset{\sim}{F}^{(b)\,i}_{n+1} \qquad (48)$$

where $\underset{\sim}{F}^{(b)i}_{n+1}$ are the internal forces corresponding to the plate bending part only:

$$\underset{\sim}{F}^{(b)i}_{n+1} = \int_{\Omega e} \underset{\sim}{B}^T \cdot z \cdot \left(\underset{\sim}{\sigma}^i_{n+1} + \Delta\underset{\sim}{\sigma}^i_{n+1} \right) d\Omega \qquad (49)$$

Using the constitutive incremental equation (41), we can write:

$$\Delta\underset{\sim}{\sigma}^i_{n+1} = \underset{\sim}{D}^i_{n+\alpha} \cdot \left(\Delta\underset{\sim}{\varepsilon}^i_{n+1} + \underset{\sim}{\phi}^i_{n+1} \right) \qquad (50)$$

where $\underset{\sim}{D}^i_{n+\alpha} = (\underset{\sim}{C}^i_{n+1})^{-1}$ is the tangent modulus matrix and $\underset{\sim}{\phi}^i_{n+1}$ is the out of balance of the constitutive equation (32).
Using eq. (45) and eq. (50), we can rewrite eq. (49) as

$$\underset{\sim}{F}^{(b)i}_{n+1} = \underset{\sim}{K}^i_{n+\alpha} \cdot \Delta\underset{\sim}{\varphi}^i_{n+1} + \int_{\Omega e} \underset{\sim}{B}^T \cdot z \cdot \left(\underset{\sim}{\sigma}^i_{n+1} + \underset{\sim}{D}^i_{n+\alpha} \cdot \underset{\sim}{\phi}^i_{n+1} \right) d\Omega \qquad (51)$$

and the out of balance forces as

$$\underset{\sim}{R}^i_{n+1} = \underset{\sim}{F}^{(e)i}_{n+1} - \left(\underset{\sim}{F}^{(b)i}_{n+1} + \underset{\sim}{F}^{(s)i}_{n+1} \right) \qquad (52)$$

where $\underset{\sim}{F}^{(e)}$ = external applied forces

$\underset{\sim}{F}^{(b)}$ = internal forces from the plate bending response
$\underset{\sim}{F}^{(s)}$ = internal forces from the transverse shear response
$\underset{\sim}{R}$ = out of balance forces

and $\underset{\sim}{K}$, in equation (51), is the tangent stiffness matrix:

$$\underset{\sim}{K}^i_{IJ\,n+\alpha} = \int_{\Omega e} \underset{\sim}{B}^T_I \cdot \underset{\sim}{D}^i_{n+\alpha} \cdot \underset{\sim}{B}_J \cdot z^2 \, d\Omega \qquad (53)$$

Equations (51) and (52) were developed at the element level.
Using the usual assembly process of the F.E. displacements approach, we will obtain similar expressions (51), (52) and (53) involving the whole domain of analysis; this standard step is not described here and can be found in many reference books. The goal now is to solve the set of algebraic equations $\underset{\sim}{K}\,\Delta\underset{\sim}{\varphi} = \underset{\sim}{R}$ using a Newton-Raphson scheme.

At this stage, we have finished with the development of the F.E. model. We shall first summarize our algorithm and then discuss in some detail the quadrature adopted for the numerical evaluation of the integrals. Finally, we shall present some numerical examples.

3.3 Finite element algorithm

i=Newton-Raphson iteration
n=time step

```
                                                                    see
                                                                    equations:
1.  Initialization :  set i=n=o,  form $\underset{\sim}{K}$ and $\underset{\sim}{F}o$ and          53
                      compute $\underset{\sim}{\varphi}o = \underset{\sim}{K}^{-1}\cdot\underset{\sim}{F}o$ and $\underset{\sim}{\sigma}_o = \underset{\sim}{D}\cdot\underset{\sim}{B}\cdot\underset{\sim}{\varphi}_o$      25, 45

2.  Select $\Delta t_{n+1}$ ; $\underset{\sim}{\varphi}^{o}_{n+1} = \underset{\sim}{\varphi}_n$  and  $\underset{\sim}{\sigma}^{o}_{n+1} = \underset{\sim}{\sigma}_n$ , set i=1

3.  At each Gauss point, for each nonlinear iteration i

    3a) update the stresses  $\underset{\sim}{\sigma}^{i}_{n+1} \leftarrow \underset{\sim}{\bar{\sigma}}^{i-1}_{n+1} + \underset{\sim}{D}^{i-1}_{n+\alpha}\cdot\Delta\underset{\sim}{\varepsilon}^{i}_{n+1}$     42, 35, 50

    3b) compute the elasticity matrix  $\underset{\sim}{D}^{i}_{n+\alpha} = (\underset{\sim}{C}^{i}_{n+\alpha})^{-1}$     42

    3c) compute the constitutive eq. out of balance $\underset{\sim}{\phi}^{i}_{n+1}$     32

    3d) compute the stresses  $\underset{\sim}{\bar{\sigma}}^{i}_{n+1} \leftarrow \underset{\sim}{\bar{\sigma}}^{i}_{n+1} + \underset{\sim}{D}^{i}_{n+\alpha}\cdot\underset{\sim}{\phi}^{i}_{n+1}$     42, 32, 50

    3e) check convergence on the constitutive eq. $\|\underset{\sim}{\phi}^{i}_{n+1}\| \overset{?}{\leqslant} tol\,1$
        if no, go to step 3b.  if yes, $\underset{\sim}{\sigma}^{i}_{n+1} \leftarrow \underset{\sim}{\bar{\sigma}}^{i}_{n+1}$
        and go to step 4.

4.  For each element

    4a) compute the tangent stiffness $\underset{\sim}{K}^{i}_{IJ\ n+\alpha}$  and     53
        assemble into the global $\underset{\sim}{K}^{i}_{n+\alpha}$

    4b) compute the out of balance forces $\underset{\sim}{R}^{i}_{I\,n+1}$     52
        and assemble into the global $\underset{\sim}{R}^{i}_{n+1}$

5.  Solve the set of equations  $\underset{\sim}{K}^{i}_{n+\alpha}\,\Delta\underset{\sim}{\varphi}^{i}_{n+1} = \underset{\sim}{R}^{i}_{n+1}$ for $\Delta\underset{\sim}{\varphi}^{i}_{n+1}$


6.  Compute the increment of deformation  $\Delta\underset{\sim}{\varepsilon}^{i}_{n+1}$     45
    and update   $\underset{\sim}{\varepsilon}^{i}_{n+1} \leftarrow \underset{\sim}{\varepsilon}^{i}_{n+1} + \Delta\underset{\sim}{\varepsilon}^{i}_{n+1}$     35
                 $\underset{\sim}{\varphi}^{i}_{n+1} \leftarrow \underset{\sim}{\varphi}^{i}_{n+1} + \Delta\underset{\sim}{\varphi}^{i}_{n+1}$     35

7.  increment i :  i ← i+1,  if i $\geqslant$ $i_{max}$ , stop

    Compute Euclidean norm and check: $\|\underset{\sim}{R}^{i}_{n+1}\| \overset{?}{<} tol\,2$
    if no, go to step 3.
    if yes, define $\underset{\sim}{\varphi}_{n+1} = \underset{\sim}{\varphi}^{i}_{n+1}$ , $\underset{\sim}{\sigma}_{n+1} = \underset{\sim}{\sigma}^{i}_{n+1}$ , $\underset{\sim}{\varepsilon}_{n+1} = \underset{\sim}{\varepsilon}^{i}_{n+1}$

8.  $n \leftarrow n+1$; if $\sum_{j=1}^{n} \Delta t_j < t_{max}$  go to step 2; otherwise, stop.
```

Remarks about the algorithm

1. In both steps 3a and 3d, we see a new symbol: $\bar{\underset{\sim}{\sigma}}$; it comes from the fact that at the step 3d, we cannot compute the exact stresses given by eq. (50) because $\Delta\underset{\sim}{\varepsilon}$ is unknown:

eq. (50) : $\Delta\underset{\sim}{\sigma} = \underset{=}{D} \cdot (\Delta\underset{\sim}{\varepsilon} + \underset{\sim}{\phi}) = \underset{=}{D} \cdot \Delta\underset{\sim}{\varepsilon} + \underset{=}{D} \cdot \underset{\sim}{\phi}$

then $\underset{\sim}{\sigma} \leftarrow \underset{\sim}{\sigma} + \underset{=}{D} \cdot \Delta\underset{\sim}{\varepsilon} + \underset{=}{D} \cdot \underset{\sim}{\phi}$; except $\Delta\underset{\sim}{\varepsilon}$, all the quantities are known; let us define

$\bar{\underset{\sim}{\sigma}} = \underset{\sim}{\sigma} + \underset{=}{D} \cdot \underset{\sim}{\phi}$ as a predictor type stress; then once the deformations are known for the current iteration , we shall update it. This corrector part is done at the beginning of the next step for ease of implementation (step 3a)

2) We have not yet discussed step 3e, which is conceptually of some importance. In our linearization of the viscoplatic constitutive equation, we note that a small error in $\underset{\sim}{\sigma}$ may lead to a very large change in $\underset{\sim}{\varepsilon}$; eventually, this will slow down the rate of convergence of the Newton-Raphson scheme.
For this reason, we iterate <u>at the Gauss point level</u> in order to get a sufficiently small out of balance $\underset{\sim}{\phi}$ (eq. 32) for the <u>current</u> $\underset{\sim}{\varepsilon}^{i}$.
Equation (41) becomes:

$$\underset{\sim}{\phi}^{i,j}_{n+1} + \Delta\underset{\sim}{\varepsilon}^{i}_{n+1} - \underset{\sim}{C}^{i,j}_{n+\alpha} \cdot \Delta\underset{\sim}{\sigma}^{i,j}_{n+1} = 0 \qquad (53b)$$

where j is the local iteration counter.
It is shown in section 4.2 that this substantially improves the global convergence.

3) In the present stage of development, our element uses a data base which records the stresses and strains for the current iteration and the previous converged time step. It also stores the current tangent elasticity matrix. This is done at each integration point (i.e. requires 18 storages per point, that is, 216 for 4×3=12 integration points: 4 in the x-y plane, 3 through the thickness).

Note that it is expected that the introduction of the convergence on the constitutive equation (step 3e) will allow us to skip step 3a, hence reducing the storage to only 6 values per integration point (stresses and strains for the previous time).

3.4 Numerical integration

One important aspect of the finite element method is the use of numerical integration (quadrature) to evaluate the integrals appearing in both the stiffness (eq. 53) and in the out of balance forces (eq. 51). These evaluations are usually done using a local coordinate system.

a) Isoparametric mapping

Splitting the volume integral into the plane of the plate and its thickness (h), the stiffness given in eq. (53) becomes

$$\underset{\sim}{K}_{IJ} = \int_{\Omega e} \underset{\sim}{B}_I^T \underset{\sim}{D} \underset{\sim}{B}_J z^2 d\Omega = \int_A \int_{-h/2}^{+h/2} \underset{\sim}{B}_I \underset{\sim}{D} \underset{\sim}{B}_J z^2 dz \, dA \tag{54}$$

using an isoparametric mapping, we have the following relations:

$$z = \frac{h}{2} \cdot \zeta \qquad \text{and} \qquad dz = \frac{h}{2} d\zeta \tag{55}$$

$$\text{also} \qquad dA = |\underset{\sim}{J}| \cdot d\xi \, d\eta \tag{56}$$

where $|\underset{\sim}{J}|$ is the determinant of the Jacobian matrix of the mapping:

$$\underset{\sim}{J} = \begin{bmatrix} x_{,\xi} & y_{,\xi} \\ x_{,\eta} & y_{,\eta} \end{bmatrix} \tag{57}$$

Using the relations (55) through (57), we can express the stiffness (eq. 54) as

$$\underset{\sim}{K}_{IJ} = \int_{-1}^{+1}\int_{-1}^{+1} \left\{ \int_{-1}^{+1} \underset{\sim}{B}_I^T \cdot \underset{\sim}{D} \cdot \underset{\sim}{B}_J \cdot |\underset{\sim}{J}| \cdot \left(\frac{h^2}{4} \zeta^2\right) \cdot \frac{h}{2} \cdot d\zeta \right\} d\xi \, d\eta \tag{58}$$

Similarly, the integral appearing in the out of balance forces (eq. 51), when using $\underline{\bar{\sigma}} = \underline{\sigma} + \underline{D} \cdot \underline{\phi}$, becomes

$$\int_{\Omega e} \underset{\sim}{B}_I^T z \underset{\sim}{\bar{\sigma}} d\Omega = \int_{-1}^{+1}\int_{-1}^{+1} \left\{ \int_{-1}^{+1} \underset{\sim}{B}_I^T \cdot \underset{\sim}{\bar{\sigma}} \cdot \left(\frac{h}{2} \zeta\right) \cdot \frac{h}{2} \, d\zeta \right\} d\xi \, d\eta \tag{59}$$

b) Quadratures

The integrals over the plane ξ, η of the plate will be carried out using Gauss-Legendre quadrature (2 x 2 points will suffice for our 4-node element).
For the integration through the thickness, the problem is different.
When the plastic zone spreads towards the midplane of the plate, the function to integrate looks more and more like a step function (fig. 4a).
Three different quadratures were tested, namely:

1) Gauss-Legendre (G): standard quadrature. N point integrate exactly polynomials up to the degree $2 \cdot N-1$.
2) Gauss-Lobatto (G-L): here, two sampling points must lie at the endpoints of the interval. N points integrate exactly polynomials up to the degree $2 \cdot N-3$.
3) Half Gauss-Legendre (H-G): same as (G) above, but applied to each half thickness of the plate (i.e. $-\frac{h}{2} \leq z \leq 0$ and $0 \leq z \leq h/2$). This allows us to consider a discontinuity of stresses at z=0.

Notice that an odd number of points for G or G-L will not be efficient because one point would lie at the midplane of the plate, where there are no stresses (due to the antisymmetry of the stresses). The same reason applies to reject the idea of having a half Gauss-Lobatto quadrature (sampling points at the ends, hence one at the midplane).

The idea of introducing a quadrature on half the thickness only was due to the fact that neither G nor G-L are able to compute exactly the "fully plastic" section (fig. 4b), no matter how many points we choose.
However, if we integrate separately the two halves of the thickness, the fully plastic section would be exactly computed, even with two point quadrature!

When using the H-G quadrature, the equations (58) and (59) must be slightly modified:

$$z = \frac{h}{4}(\zeta+1) \qquad dz = \frac{h}{4} \cdot d\zeta \tag{60}$$

$$\underset{\sim}{K}_{IJ} = \int_A \cdot \int_{-h/2}^{h/2} = \int_A \cdot 2 \cdot \int_0^{+h/2} = \int_{-1}^{+1}\int_{-1}^{+1} \cdot 2 \cdot \left\{ \int_{-1}^{+1} \underset{\sim}{B}_I^T \underset{\sim}{D} \underset{\sim}{B}_J \cdot |\underset{\sim}{J}| \cdot \left[\frac{h}{4}(\zeta+1)\right]^2 \cdot \frac{h}{4} d\zeta \right\} d\xi \, d\eta \tag{61}$$

$$\int_{\Omega e} \underset{\sim}{B}_I^T z \, \underset{\sim}{\bar{\sigma}} \, d\Omega = \int_{-1}^{+1}\int_{-1}^{+1} \cdot 2 \cdot \left\{ \int_{-1}^{+1} \underset{\sim}{B}_I^T \cdot \underset{\sim}{\bar{\sigma}} \cdot \left[\frac{h}{4}(\zeta+1)\right] \cdot \frac{h}{4} d\zeta \right\} d\xi \, d\eta \tag{62}$$

In order to compare the three quadratures, we define the following <u>test problem</u>:

Thickness=2, stresses $\sigma_y = \tau_{xy} = 0$, $\sigma_x = \sigma_x(z) = 0$ at midplane, linear for $0 \leqslant Z \leqslant Z_o$ and $\sigma_x = \sigma_Y$ for $z_o \leq z \leq h/2$ (see fig. 4a). The normalized moment is computed by

$$\hat{M}/\sigma_Y = \sum_{i=1}^{N} \sigma(\xi_i) \cdot \xi_i \cdot w_i$$

where ξ_i and w_i are the integration abscissae and weights, respectively (see Appendix I). N is the number of integration points and σ_Y the yield stress.

The following table gives the error $\frac{\hat{M}-M}{M} \cdot 100$ for Zo = 0, i.e. <u>fully plastic</u>. The exact moment is $M = \frac{h^2}{4} \cdot \sigma_Y$

#points through thickness	Gauss-Legendre (G)	Gauss-Lobatto (G-L)	Half Gauss-Legendre (H-G)
4	4.3 %	7.9 %	0.
6	2.0 %	2.9 %	0.
8	1.1 %	1.5 %	0.

Except for H-G which gives the exact result, all moments are overestimated. Since G-L is the first to detect any yielding (sampling points at top and bottom of the plate), it was expected to give better results than G; surprisingly they are worse, confirming the results found by Cormeau (Cormeau, 78).

In figure 5, we see the approximations given by the three quadratures, for an elastic to a fully plastic section. Again, the H-G quadrature is superior to the others, almost for all values of Zo. The result is true for both 4 and 6 points. This is confirmed in figure 6 where we plot the corresponding errors.

The importance of using H-G quadrature is most evident when we are close to a mechanism in a structure (i.e. when a small increase of load leads to a very large increase in deformations), as we shall see in the numerical examples presented in the next section.

4. NUMERICAL EXAMPLES

In this section we describe three examples, namely (i) a single plate element subjected to applied boundary displacements, (ii) a cylindrical bending of a clamped plate subjected to uniform loading and (iii) a simply supported square plate subjected to uniform loading.

All the tests were computed using K=1 in the plastic flow law (eq. 19). 1/

4.1 Single element test

The motivation to perform a single element test was to gain some insight of the viscoplastic behavior. By applying constant rates of deformations (here rotations), we see in Figure 7 that the response of the plate is almost elastic for very high rates and elastic-perfectly plastic for small rates of deformations. Figure 7 was obtained for a constant value of $\gamma\Delta t$ (where γ is the material parameter of eq. 16 and Δt the time step). The same type of test may be used to select $\gamma\Delta t$ by fitting experimental data.

When we are interested only in an elastic-perfectly plastic behavior, we can modify the test by imposing a large deformation in one step, and redo the test for different γ's. The result is shown in figure 8a, where we plot the moment M at the built-in end as a function of γ.
As expected, the moment M tends to M_u when γ increases; we could then select any $\gamma \geqslant 10^{-4}$ (fig. 8a). However, if the γ is too large, the number of the Newton-Raphson iterations needed to obtain a given accuracy increases (fig. 8b); for very large γ's, the tangent compliance matrix (eq. 42) can even become singular. The best γ is the smallest one still giving $M=M_u$, here: $\gamma\Delta t \doteq 10^{-4}$.

4.2 Cylindrical bending of a clamped plate

In this problem, we want to check if our plate element is accurate for a cylindrical bending problem, namely a plate clamped at both ends. The geometry and boundary conditions are given in figure 9a. The material properties selected were $E=30\cdot10^3$, $\nu=0.3$, $\sigma_Y=30$ and $\gamma\Delta t=10$ (selected as explained in section 4.1).
The uniform load was applied in 11 steps.

We can compute an upper bound estimate of the collapse load (Pu) using virtual work (fig. 9b):

$$Pu = \frac{16\, M_u}{l^2} = 24.1\cdot10^{-4} \quad (\text{for } l=240) \qquad (63)$$

where $M_u=(h^2/4)\cdot\sigma_Y\cdot\beta = 8.66$ ($\beta=1.155$ is used to account for cylindrical bending, i.e. stresses in the y direction, with $\nu_p=0.5$).

1/ In the sequel, we shall use the following terms: M_Y = uniaxial moment when the first fiber yields; φ_Y = corresponding rotation. Mu = ultimate uniaxial moment of the section: $Mu=(h^2/4)\cdot\sigma_Y = \frac{3}{2}\cdot M_Y$; φ_u = rotation at which continued flow starts.

Due to the mesh discretization, our solution will exceed this load; however, we can define two bounds, depending on the value of l in eq. (63):

a) lower bound: l =231.5, the distance between the two hinges close to the supports, assumed at the Gauss points (see fig. 9b)

then $p_{u_1} = 25.9 \quad 10^{-4}$

b) upper bound: l =220, the distance between the two hinges close to the supports, assumed at the center of the elements.

then $p_{u_2} = 28.6 \quad 10^{-4}$

These two bounds are indicated in figure 10a where we plot the load versus center displacement response for the H-G quadrature. The ultimate load (27.6 10^{-4}) compares well with the two bounds; moreover, both 4 and 6 integration points through the thickness give the same collapse load, which is very satisfactory.

In figure 10b we show the load-deflection curves obtained using 6 points through the thickness, for the three different quadratures studied. As expected, H-G quadrature gives better results than G or G-L, which still have a positive slope at w_o=20. In figure 10c, the same quadratures with 4 points through the thickness are shown; again, H-G gives the best results.

In the following table we summarize the results of the three last load steps for the 6 point quadratures:

step #	load P	vertical deflection at the center: Wo		
		G-L	G	H-G
9	27.5	14.19	15.29	16.55
10	29.1	20.83	23.40	375.0
11	30.6	1624.0	2096.0	3128.0

Of course, displacement of 3 digits or more means that we have reached the collapse mechanism shown in figure 9b.

<u>Local iterations for the constitutive equation</u>

In figure 10d, we plot the number of Newton-Raphson iterations needed at each load step to satisfy the convergence tolerance:

$$\|\underset{\sim}{R}_n\| < 10^{-9}$$

where $\|\underset{\sim}{R}_n\|$ is the Euclidean norm of the out of balance forces:

$$\|\underset{\sim}{R}_n\| = (\underset{\sim}{R}_n \cdot \underset{\sim}{R}_n)^{1/2}$$

Two curves are plotted: with and without the step 3e of the Finite element algorithm (section 3.3, eq. 53b), i.e. with and without iterations at the Gauss point level to approximate the constitutive equation (1 to 4 iterations are usually enough). The reduction of the number of Newton-Raphson iterations is quite significant, especially when close to the collapse load (fig. 10d, step 10).

4.3 Simply supported square plate

This last example concerns a square plate, simply supported, under uniform pressure applied step-by-step. Due to the symmetry of the geometry and load, one quadrant only was discretized with our 4-node element .
Figure 11 illustrates the mesh layout and boundary conditions; a coarse mesh (3 x 3 elements) was chosen to limit the computer cost.
The material parameters are: $E=2{\cdot}10^6$, $\nu=0.3$, $\sigma_Y=2000$.
Using the test mentioned in section 4.1, the "optimal" $\gamma\Delta t$ was found to be $\gamma\Delta t=10$.
The yield line theory gives the collapse load $p_u = 24\,M_p/\ell^2$ when we neglect the corner effects (i.e. the yield lines coincide exactly with the two diagonals of the plate). Here, Mp is the unidimensional plastic moment ($M_p = M_u = \frac{\ell^2}{4}\cdot\sigma_Y$).

When taking the corner effects into account, we have

upper bound: $\frac{p_u \cdot \ell^2}{M_p} = 25.056$ (Leckie and Ranaweera, 70)

lower bound: " $= 24.964$ (Hodge and Belytschko, 68)

The problem was solved for two quadratures (G and H-G), both for 4 and 6 points through the thickness.

The load-deflection diagram at the plate center is given on figure 12, along with the two above bounds. Despite the rather coarse mesh used, both 4 and 6 points H-G quadrature compare well with these bounds.
In the same figure, we also report the results from Cormeau (Cormeau, 78) who uses the same mesh but quadratic elements (2 layers of 8 nodes per element). Our "ultimate" load ($p\ell^2/M_p < 25.8$) exceeds the upper bound by about 3%. As expected, 6 point quadrature gives a smoother transition from the elastic state to the fully plastic regime than 4 point quadrature.

Figure 13 shows the spreading of the yield regions as the load increases. Starting from the center and corners of the plate, with some spreading away from the diagonal. At last, the elements along the diagonal are fully plastic as predicted by yield line theory.

Finally, figure 14 shows the effect of the quadrature on this problem. Both G and H-G (with 6 points/thickness) are reported, and we see that the G quadrature has still some load carrying capacity (at $p\ell^2/M_p=25.8$), while H-G has reached its ultimate load.
The last figure (fig. 15) illustrates the increase of Newton-Raphson iterations needed to converge when $\gamma\Delta t$ is selected too large. Throughout the analysis $\gamma\Delta t=10$ was chosen according to the procedure of section 4.1; the number of iterations as a function of the load steps is shown by a solid line. The dotted line was obtained for $\gamma\Delta t=1000$, leading to a substantial increase of iterations.

CLOSURE

We have presented a formulation for a thick plate finite element with an associated viscoplasticity and a Huber-von Mises yield surface. By treating the transverse shears elastically, it is expected that the good behavior of the element at the thin plate limit will be preserved in the nonlinear range.

The viscoplastic finite element has been shown to be reliable for the elastic-perfectly plastic problems solved; even when very coarse meshes are used.

The use of our modified quadrature rule (H-G) substantially improves the behavior of the element, especially when the structure is close to a collapse mechanism, i.e. when all the subsequent deformations will be concentrated into yield lines. It is then important to compute the exact moment of the fully plastic section. Moreover, the ultimate loads obtained with 4 or 6 H-G integration points across the depth of the plate were almost identical, which is very satisfactory.

Finally, the use of inexpensive local iterations in the linearization of the constitutive equation substantially reduces the number of Newton-Raphson iterations, reducing the cost of analysis.

REFERENCES

Abramowitz, M. and A. I. Stegun, 1964. Handbook of mathematical functions., National Bureau of Standards Applied Mathematics Series - 55, U.S. Department of Commerce.

Bathe, K.-J. and E. L. Wilson, 1976. Numerical methods in finite element analysis, Prentice-Hall, Inc.

Cormeau, I., 1978. Elastoplastic thick shell analysis by viscoplastic solid finite elements, International Journal of Num. Meth. Engineering, Volume 12.

Hodge, P. G. and T. Belytschko, 1968. Numerical methods for the limit analysis of plates, Journal of Applied Mechanics, Volume 35, pp. 796-802.

Hughes, T. J. R., R. L. Taylor and W. Kanoknukulchai, 1977. A simple and efficient finite element for plate bending, International Journal of Num. Meth. Engineering, Volume 11.

Hughes, T. J. R. and K. S. Pister, 1978. Consistent linearization in mechanics of solids and structures, Computers and Structures, Volume 8, pp. 391-397.

Hughes, T. J. R. and R. L. Taylor, 1978. Unconditionally stable algorithms for quasi-static elasto/viscoplastic finite element analysis, Computers and Structures, Volume 8, pp. 169-173.

Hughes, T. J. R. and T. E. Tezduyar, 1981. Finite elements based upon Mindlin plate theory with particular reference to the four-node bilinear isoparametric element, Journal of Applied Mechanics, Vol. 48, pp. 587-596.

Leckie, F. A. and M. P. Ranaweera, 1970. Bound methods in limit analysis, Proc. Sem. Finite Element Techniques in Structural Mechanics (ed., H. Tottenham), University of Southampton, pp. 259-282.

Mase, G. E., 1970. Continuum Mechanics, Schaum's Outline Series in Engineering, McGraw-Hill, Inc.

Naghdi, P. M., 1960. Stress-strain relations in plasticity and thermoplasticity, Plasticity (eds., E. H. Lee and P. S. Symonds), Pergammon Press, London, pp. 121-167.

Perzyna, P., 1966. Fundamental problems in viscoplasticity, Advance Applied Mechanics, Volume 9, pp. 243-377.

Scheid, 1968. Numerical analysis, Schaum's Outline Series in Mathematics, McGraw-Hill, Inc.

Zienkiewicz, O. C., 1977. The finite element method, Third Edition, McGraw-Hill, Inc., London.

Appendix I : Quadrature points and weights

Given below are the abscissae and weights for both Gauss-Legendre quadrature (up to 12 points) and Gauss-Lobatto (up to 10 points). If one needs more integration points, they can be found in (Abramowitz, 64) from which these tables are extracted.
Note: Whenever possible, analytical expressions were used instead of the numerical values. Reference (Scheid, 68) was found to be helpful in deriving them.

ABSCISSAS AND WEIGHT FACTORS FOR GAUSSIAN INTEGRATION

$$\int_{-1}^{+1} f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i)$$

Abscissas $= \pm x_i$ (Zeros of Legendre Polynomials) Weight Factors $= w_i$

$\pm x_i$		w_i
	$n=2$	
0.57735 02691 89626		1.00000 00000 00000
	$n=3$	
0.00000 00000 00000		0.88888 88888 88889
0.77459 66692 41483		0.55555 55555 55556
	$n=4$	
0.33998 10435 84856		0.65214 51548 62546
0.86113 63115 94053		0.34785 48451 37454
	$n=5$	
0.00000 00000 00000		0.56888 88888 88889
0.53846 93101 05683		0.47862 86704 99366
0.90617 98459 38664		0.23692 68850 56189
	$n=6$	
0.23861 91860 83197		0.46791 39345 72691
0.66120 93864 66265		0.36076 15730 48139
0.93246 95142 03152		0.17132 44923 79170
	$n=7$	
0.00000 00000 00000		0.41795 91836 73469
0.40584 51513 77397		0.38183 00505 05119
0.74153 11855 99394		0.27970 53914 89277
0.94910 79123 42759		0.12948 49661 68870
	$n=8$	
0.18343 46424 95650		0.36268 37833 78362
0.52553 24099 16329		0.31370 66458 77887
0.79666 64774 13627		0.22238 10344 53374
0.96028 98564 97536		0.10122 85362 90376
	$n=9$	
0.00000 00000 00000		0.33023 93550 01260
0.32425 34234 03809		0.31234 70770 40003
0.61337 14327 00590		0.26061 06964 02935
0.83603 11073 26636		0.18064 81606 94857
0.96816 02395 07626		0.08127 43883 61574
	$n=10$	
0.14887 43389 81631		0.29552 42247 14753
0.43339 53941 29247		0.26926 67193 09996
0.67940 95682 99024		0.21908 63625 15982
0.86506 33666 88985		0.14945 13491 50581
0.97390 65285 17172		0.06667 13443 08688
	$n=12$	
0.12523 34085 11469		0.24914 70458 13403
0.36783 14989 98180		0.23349 25365 38355
0.58731 79542 86617		0.20316 74267 23066
0.76990 26741 94305		0.16007 83285 43346
0.90411 72563 70475		0.10693 93259 95318
0.98156 06342 46719		0.04717 53363 86512

ABSCISSAS AND WEIGHT FACTORS FOR LOBATTO INTEGRATION

$$\int_{-1}^{+1} f(x)dx \approx w_1 f(-1) + \sum_{i=2}^{n-1} w_i f(x_i) + w_n f(1)$$

Abscissas $= \pm x_i$ Weight Factors $= w_i$

n	$\pm x_i$	w_i
3	1.00000 000	0.33333 333
	0.00000 000	1.33333 333
4	1.00000 000	0.16666 667
	0.44721 360	0.83333 333
5	1.00000 000	0.10000 000
	0.65465 367	0.54444 444
	0.00000 000	0.71111 111
6	1.00000 000	0.06666 667
	0.76505 532	0.37847 496
	0.28523 152	0.55485 838
7	1.00000 000	0.04761 904
	0.83022 390	0.27682 604
	0.46884 879	0.43174 538
	0.00000 000	0.48761 904
8	1.00000 000	0.03571 428
	0.87174 015	0.21070 422
	0.59170 018	0.34112 270
	0.20929 922	0.41245 880
9	1.00000 00000	0.02777 77778
	0.89975 79954	0.16549 53616
	0.67718 62795	0.27453 87126
	0.36311 74638	0.34642 85110
	0.00000 00000	0.37151 92744
10	1.00000 00000	0.02222 22222
	0.91953 39082	0.13330 59908
	0.73877 38651	0.22488 93420
	0.47792 49498	0.29204 26836
	0.16527 89577	0.32753 97612

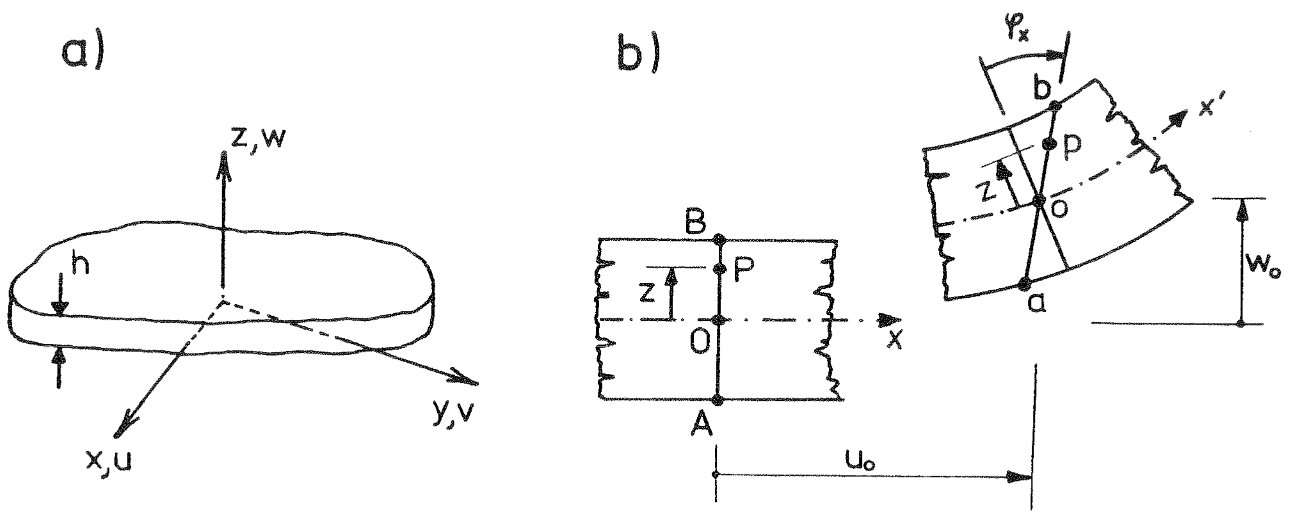


Figure 1 Coordinate system and displacements

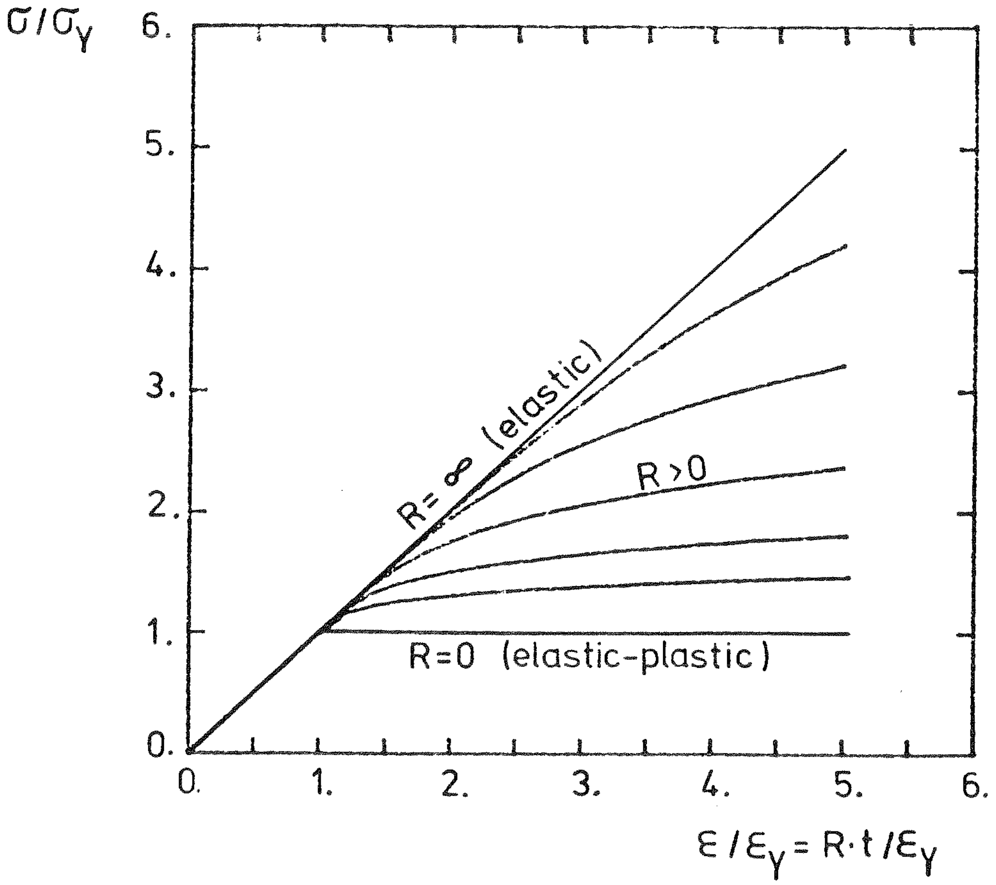


Figure 2 Rate of deformation effect on the response

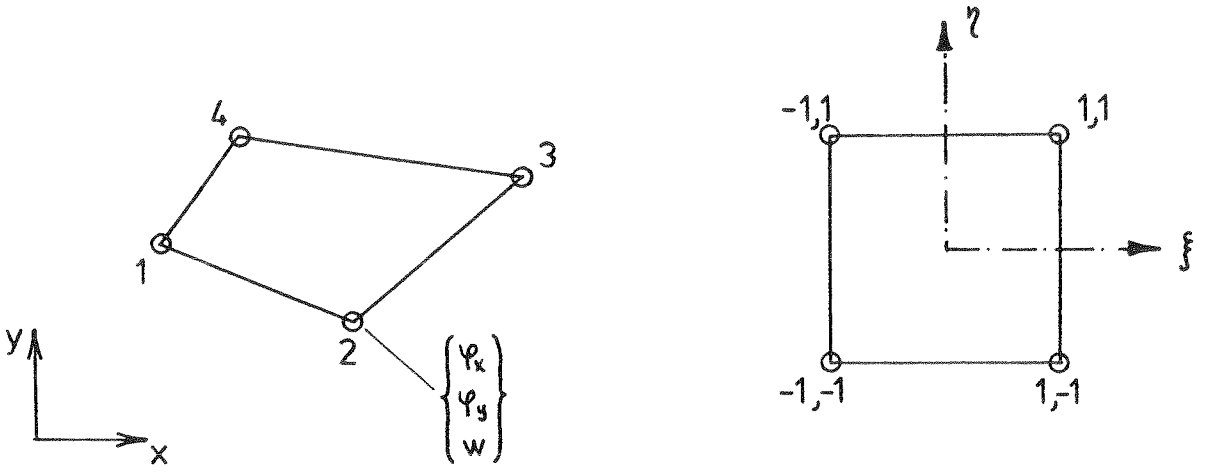


Figure 3 4-node element: nodal parameters and local coordinates

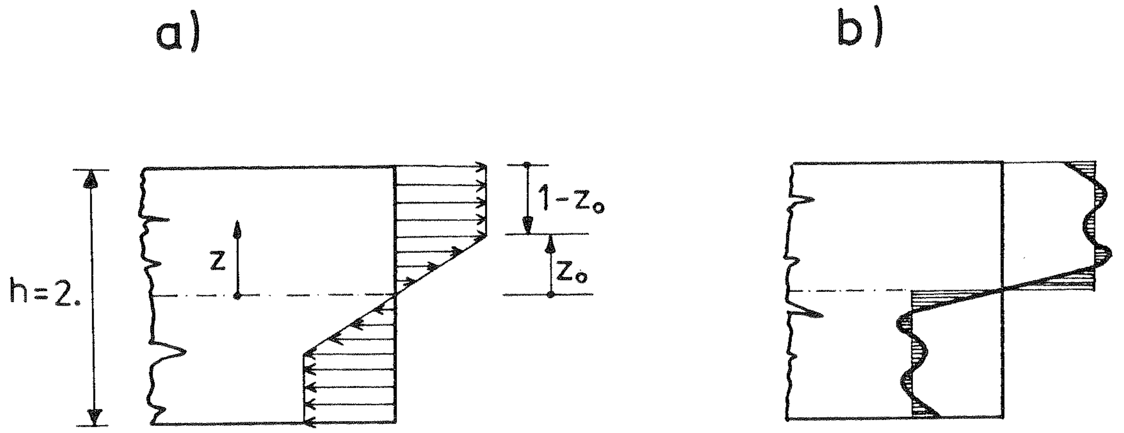
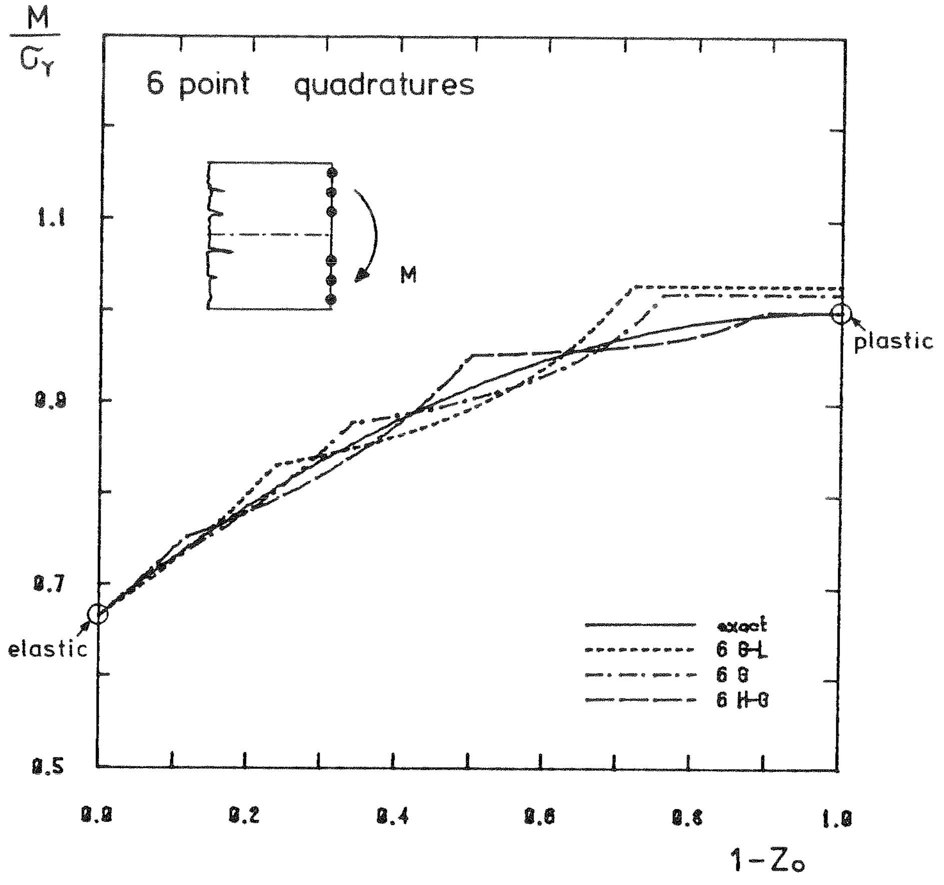


Figure 4 a) Variation of stresses through the thickness
b) Approximation of a step function using polynomials

a)



b)

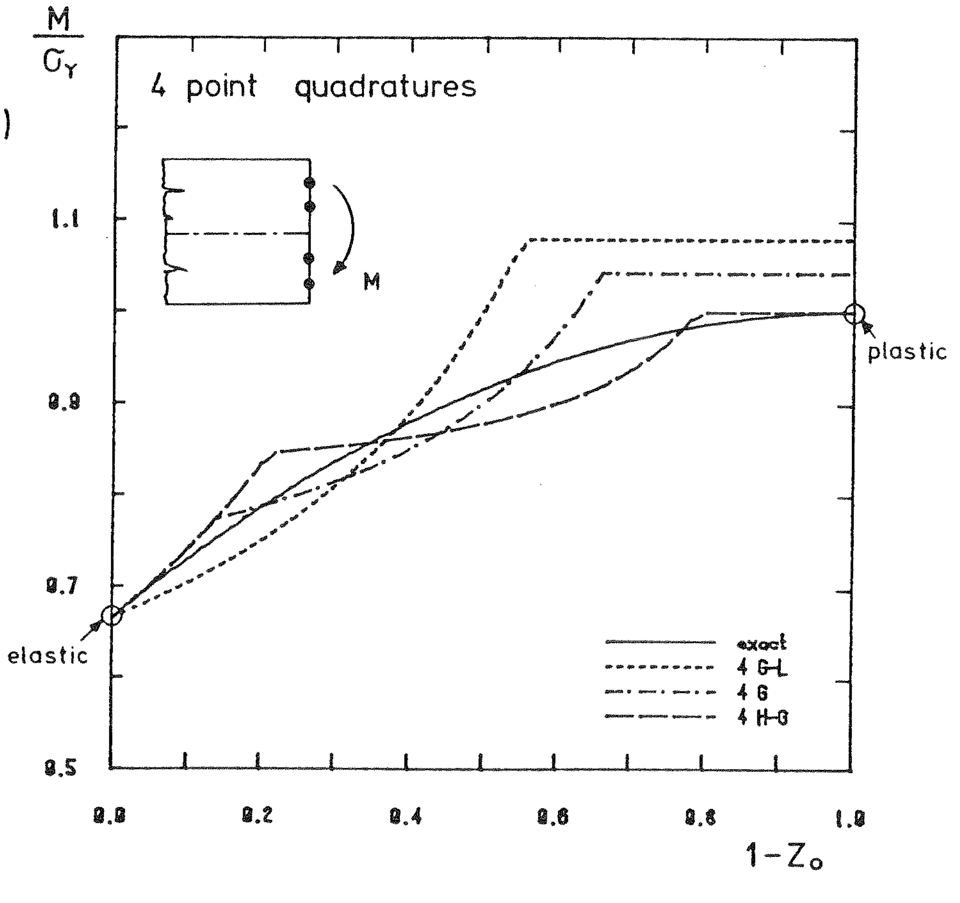
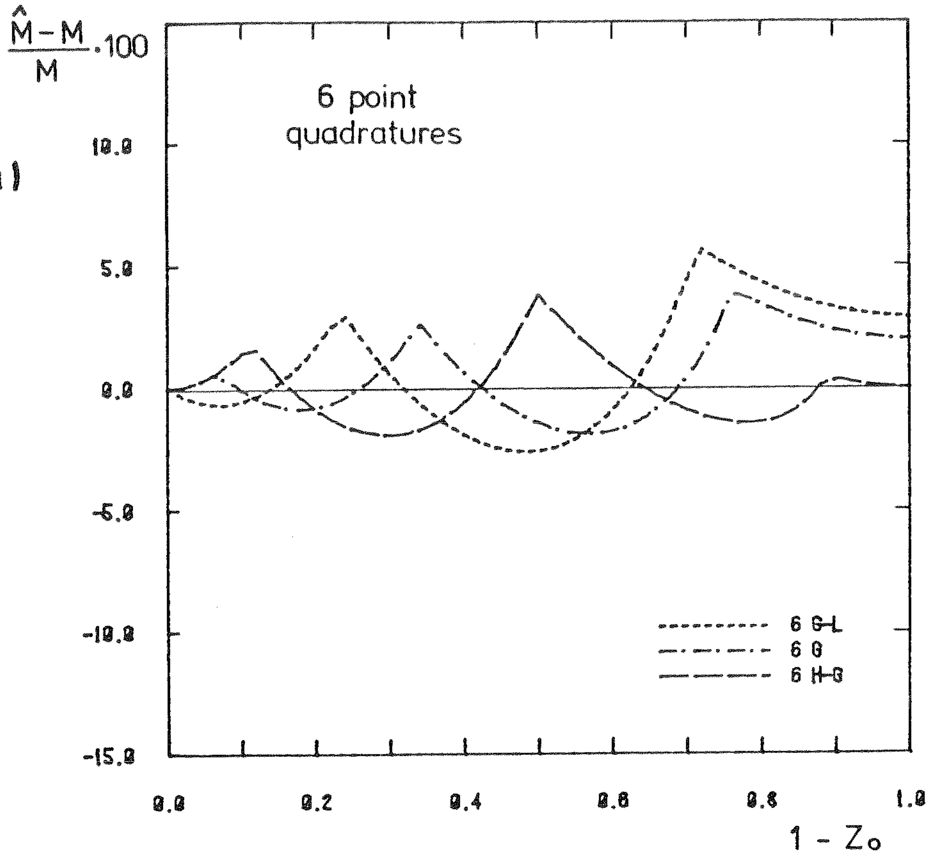


Figure 5 Numerical approximation of the moments
a) 6 points quadratures
b) 4 " "

a)



b)

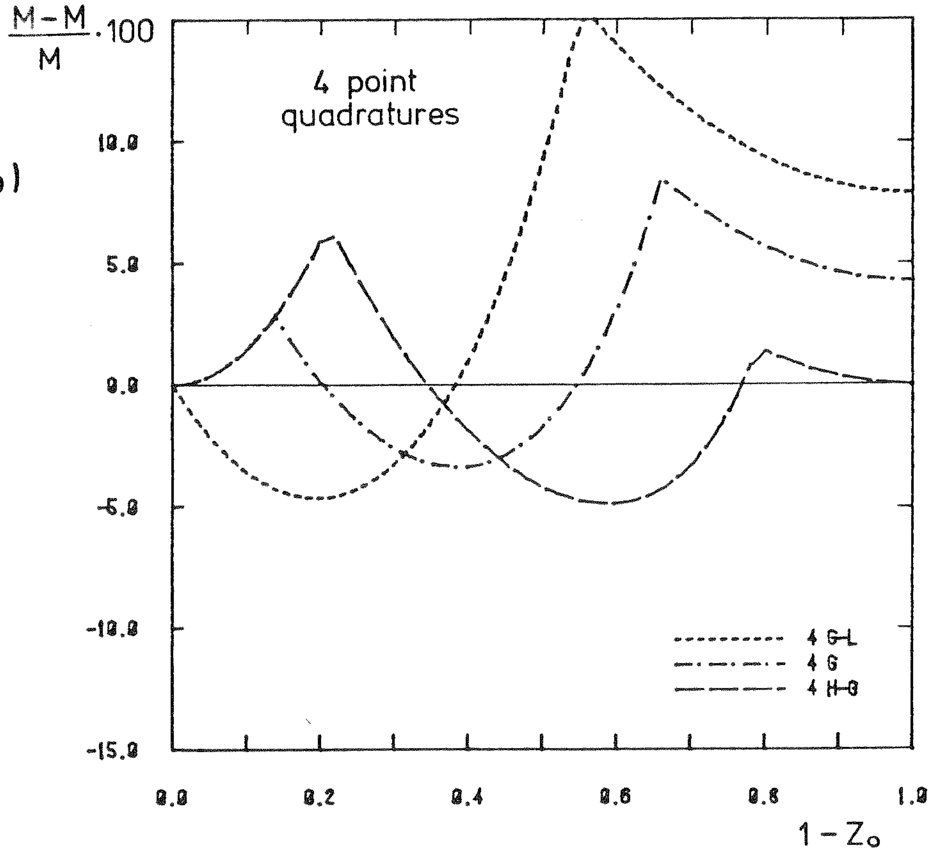


Figure 6 Relative error due to the quadrature

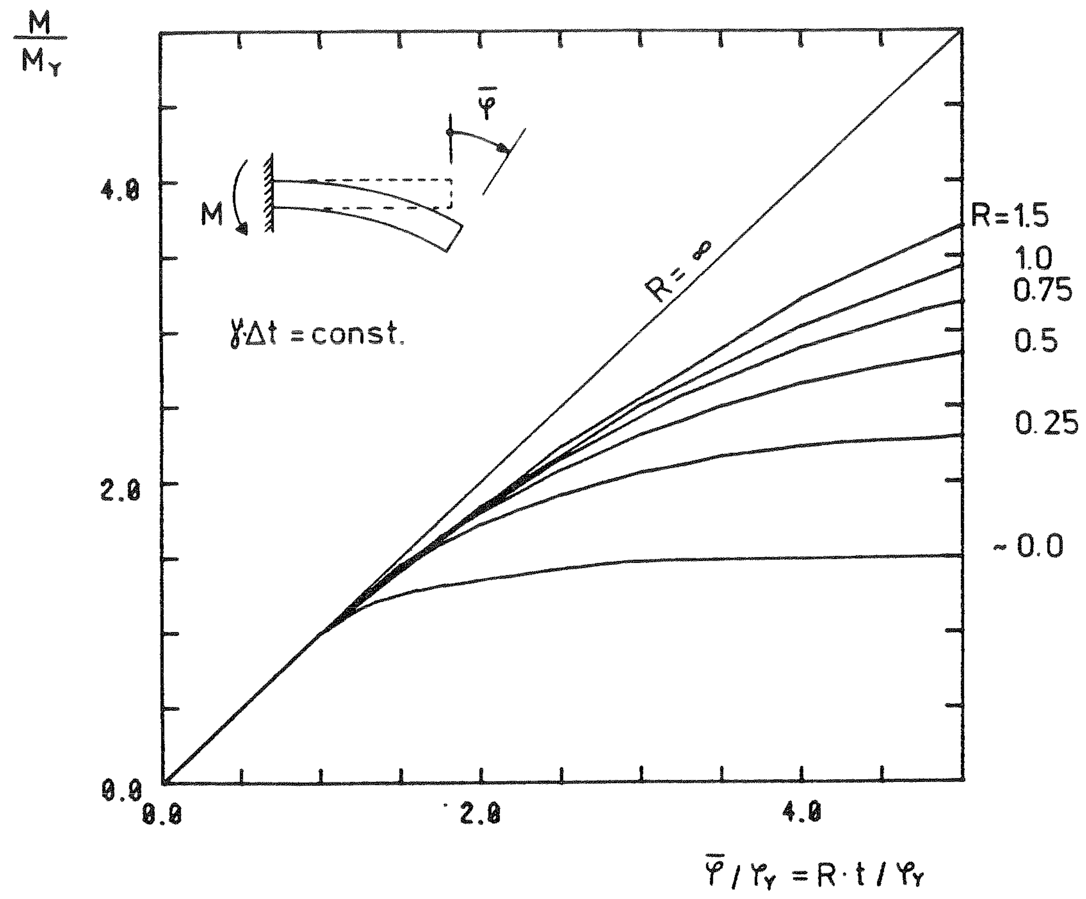
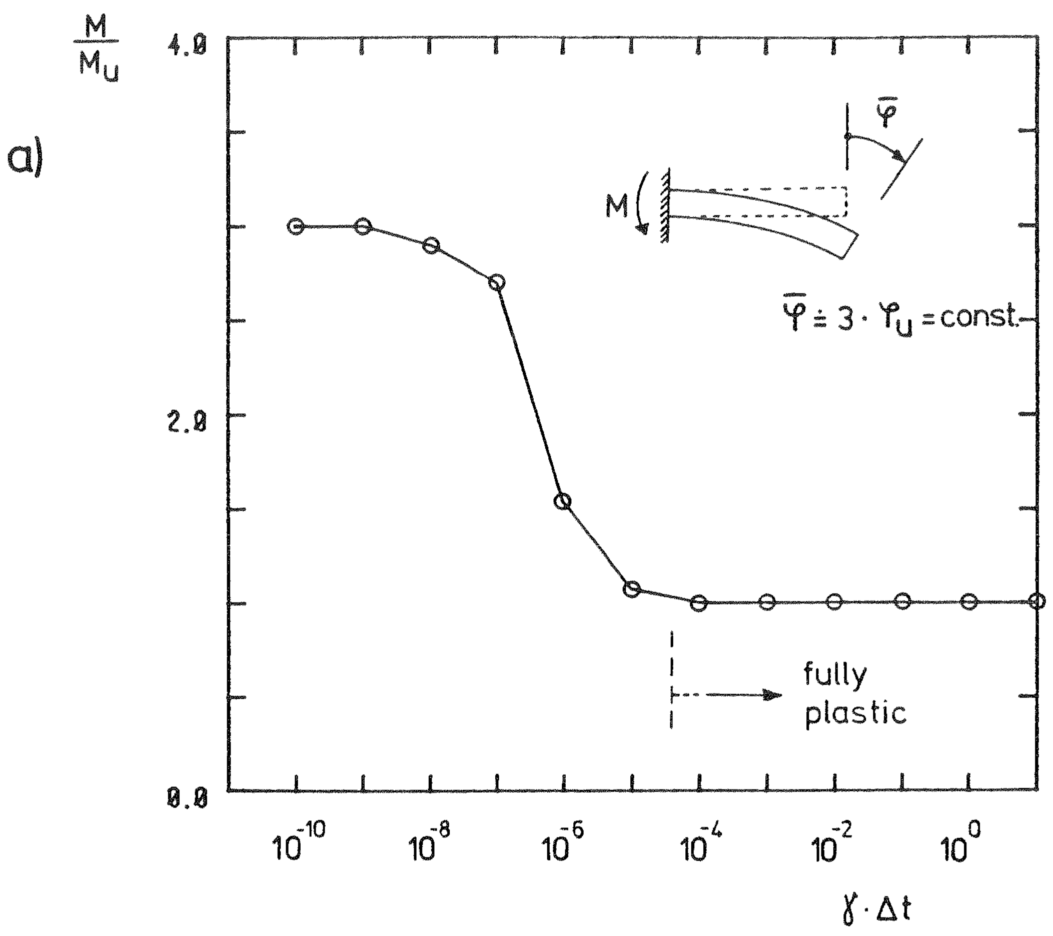


Figure 7   Single element results for data fitting



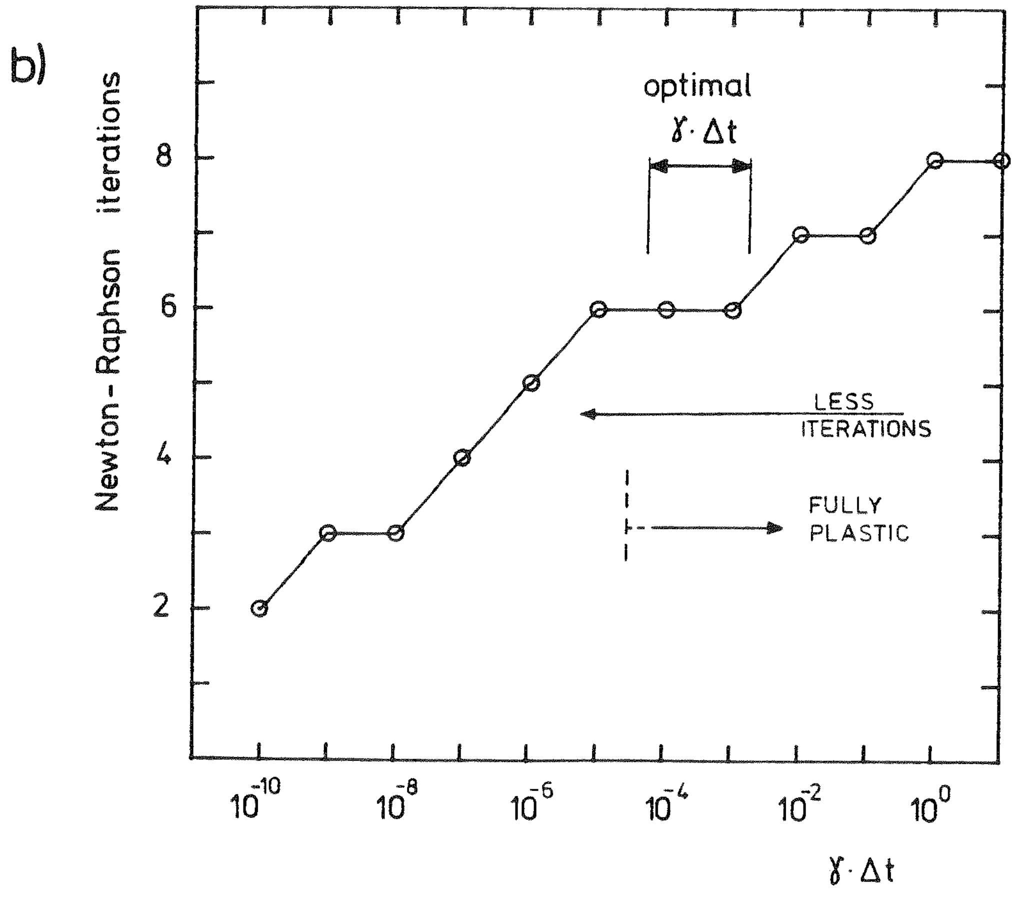
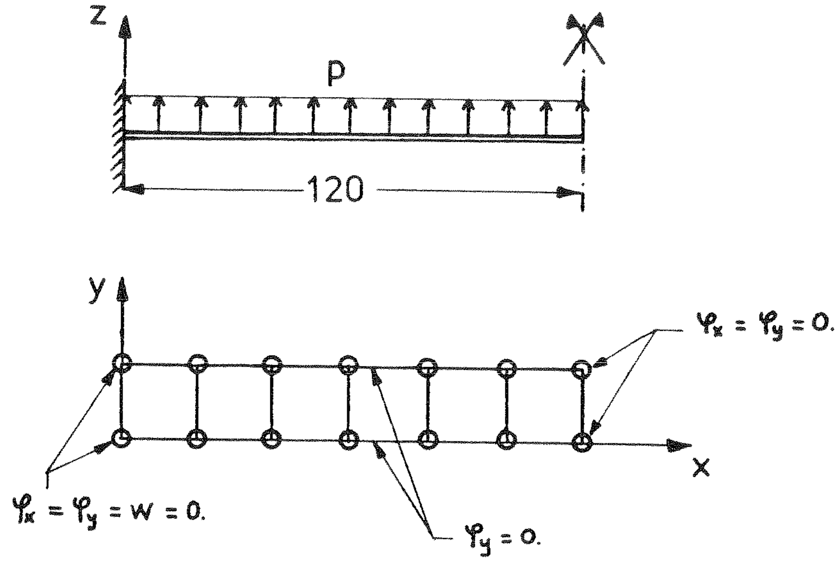


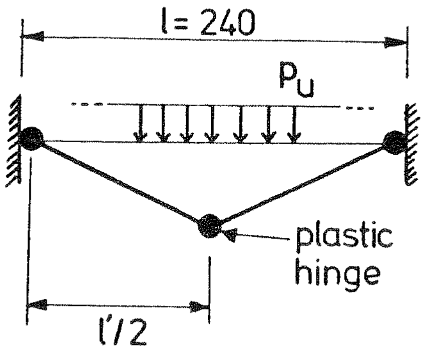
Figure 8 a) Quasi-static response as a function of $\gamma \Delta t$
b) Selection of $\gamma \Delta t$

a)


z
p
120
y
x
$\varphi_x = \varphi_y = 0.$
$\varphi_x = \varphi_y = w = 0.$
$\varphi_y = 0.$

6 Elements @ 20 x 20
thickness = 1.0

b)

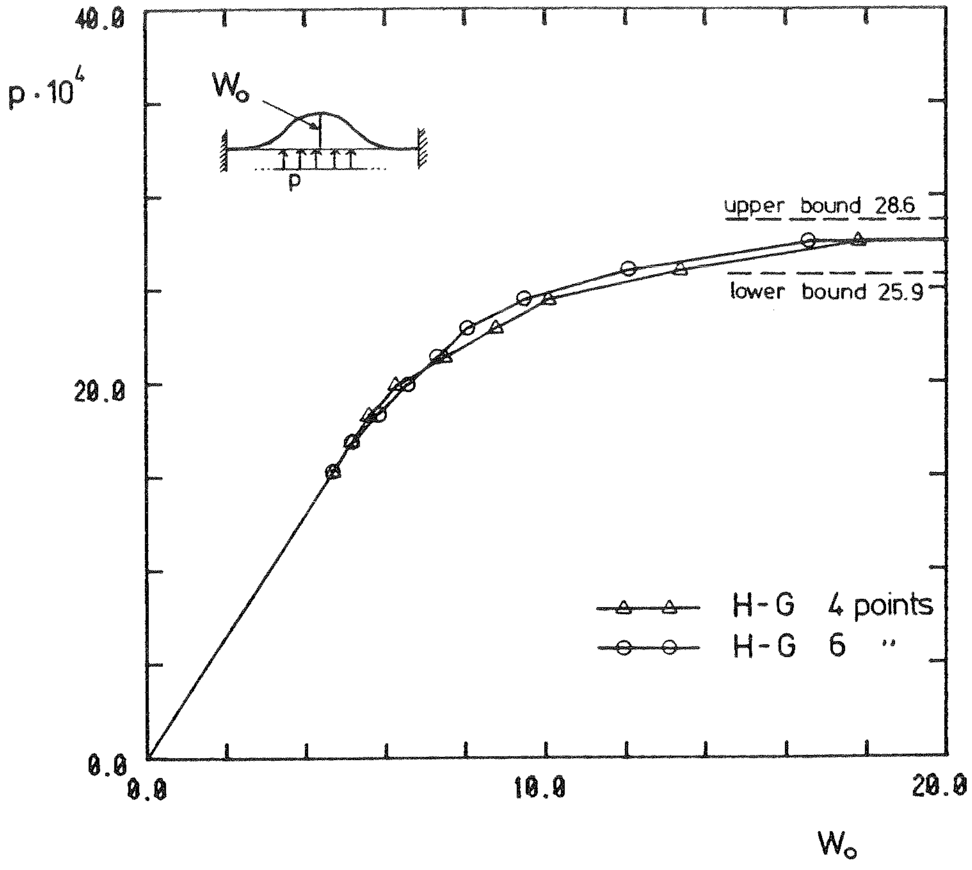


Collapse mechanism

$$p_u = \frac{16 M_u}{l^2}$$

Figure 9 a) Geometry and boundary conditions
b) Collapse mechanism

a)



b)

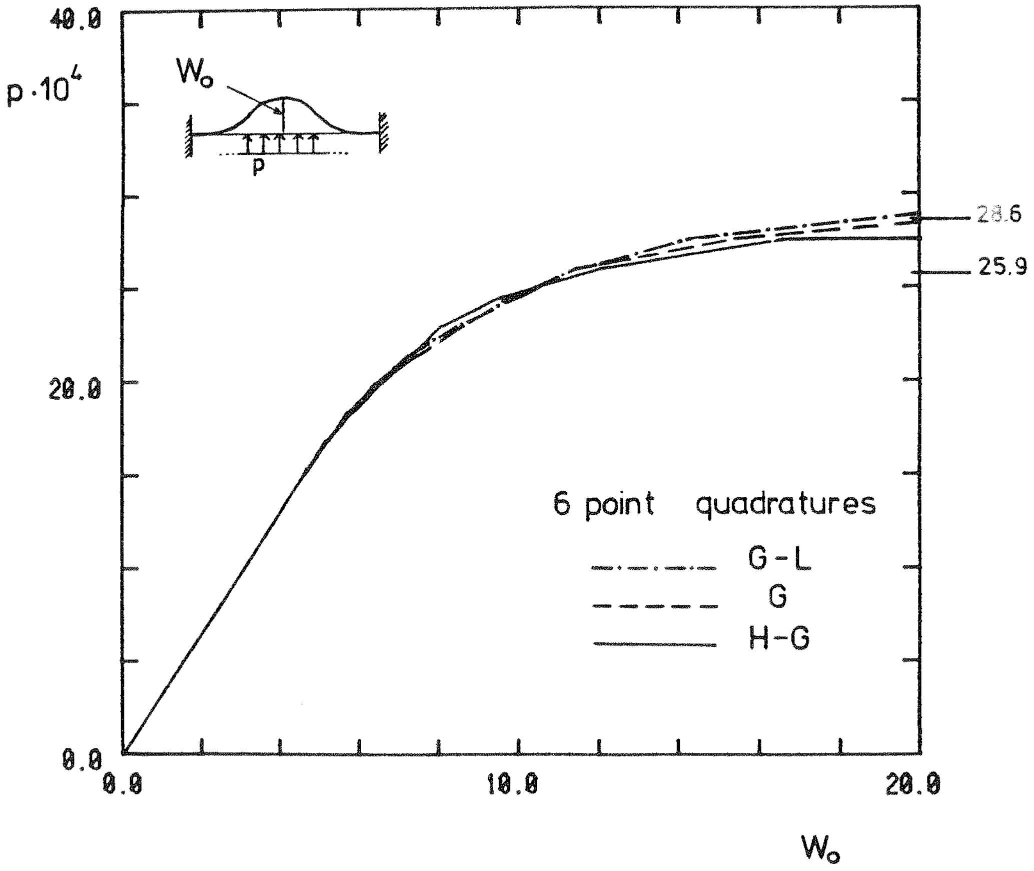
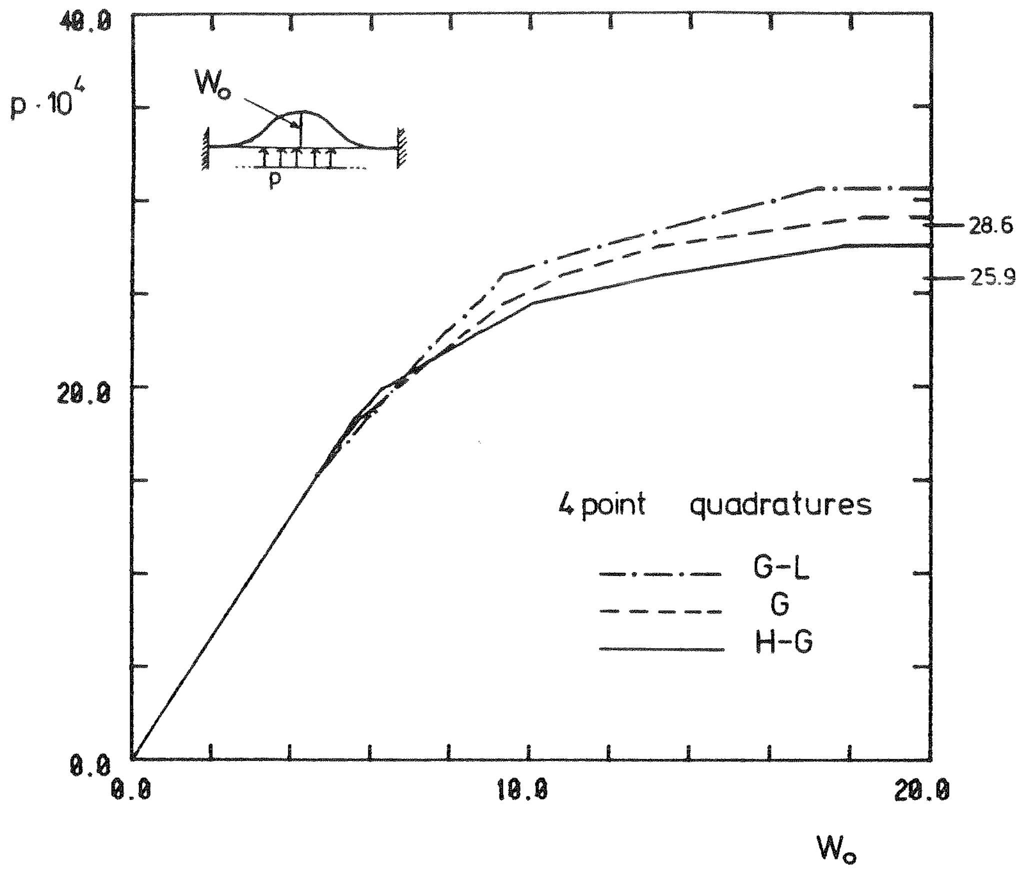


Figure 10 a) Load-deflection diagram at the center
b) Effect of the quadrature (6 points)

c)



d)

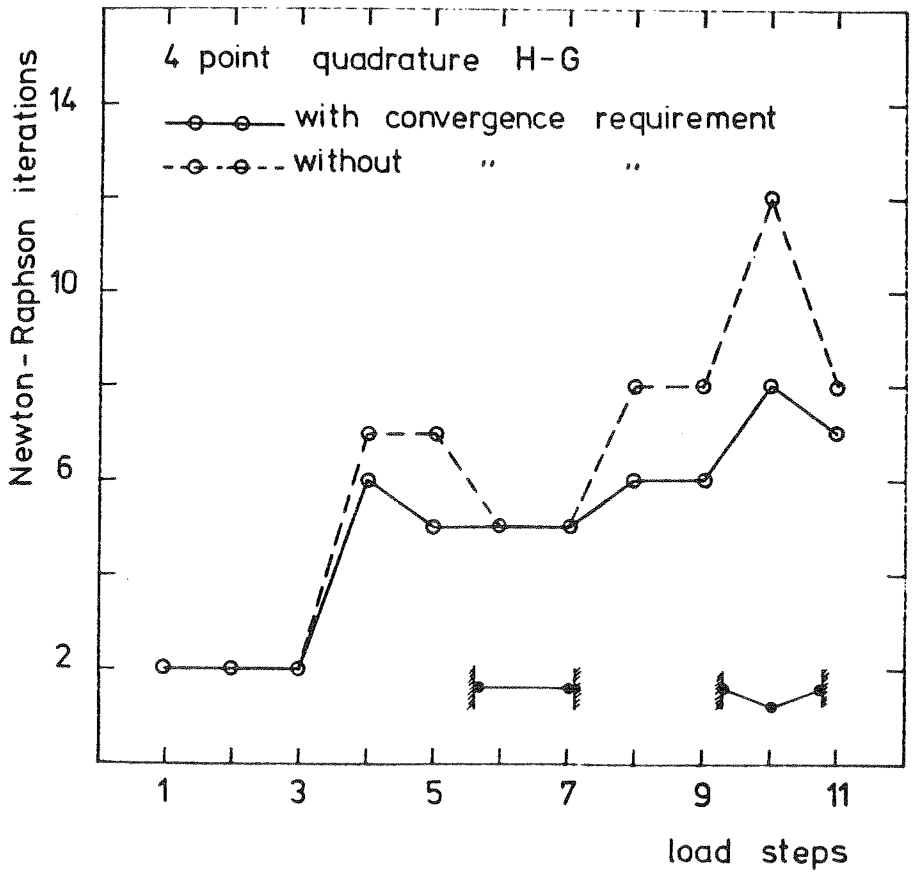


Figure 10 c) Effect of the quadrature (4 points)
d) Effect of the local iterations

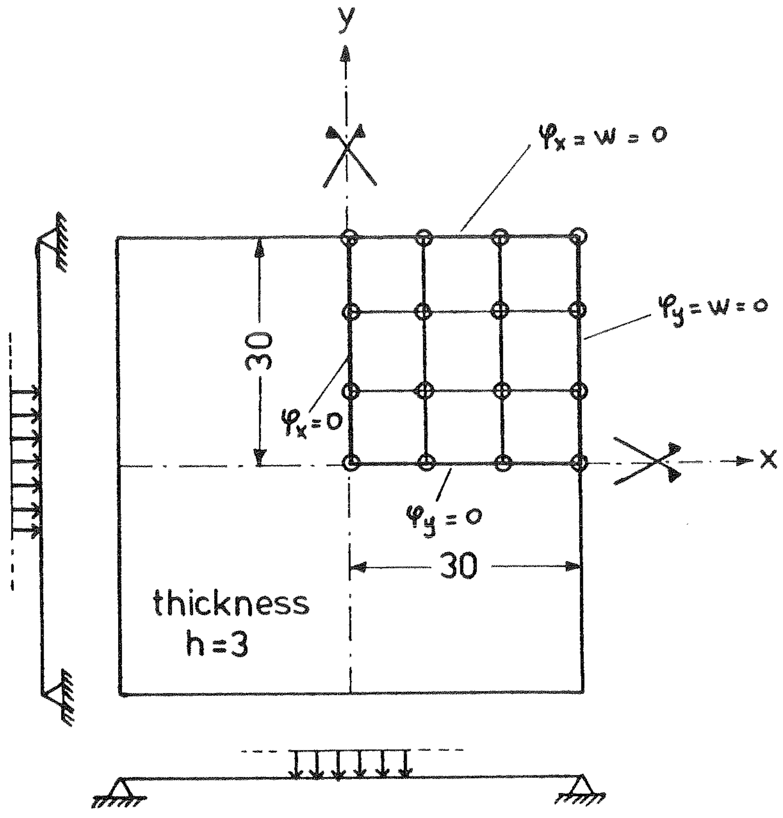


Figure 11 Geometry and boundary conditions

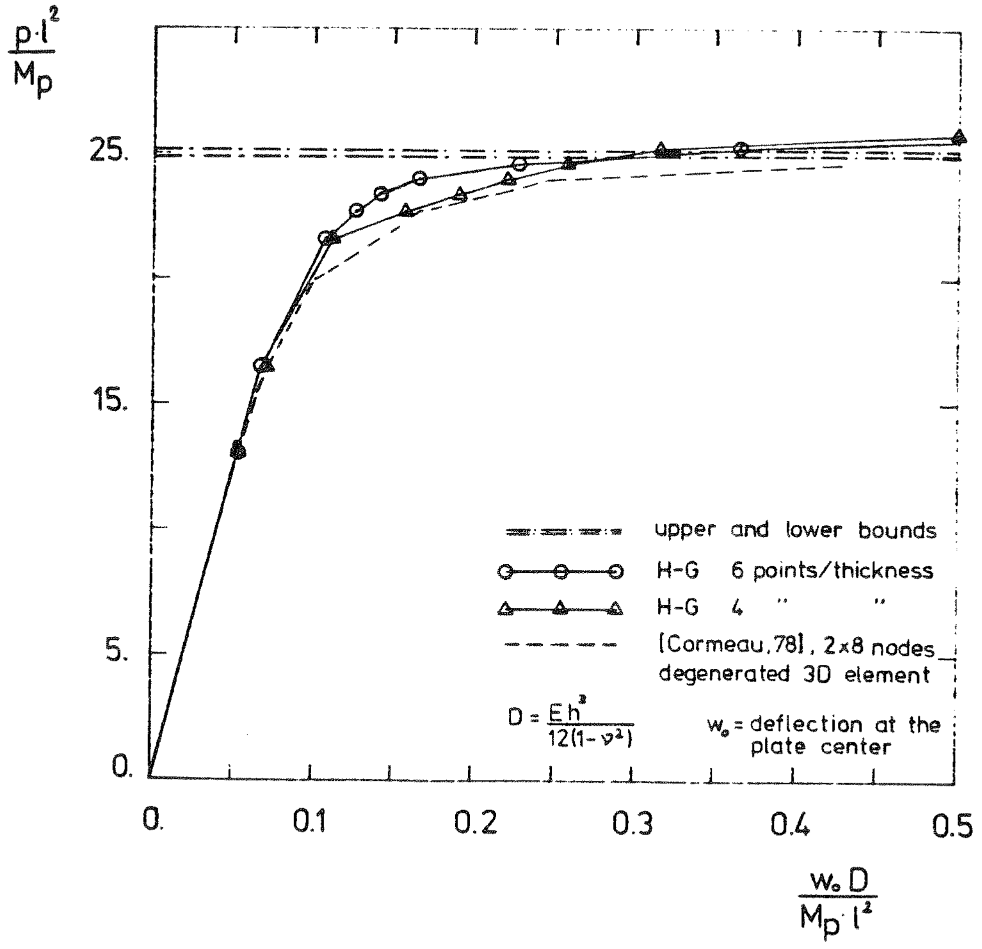
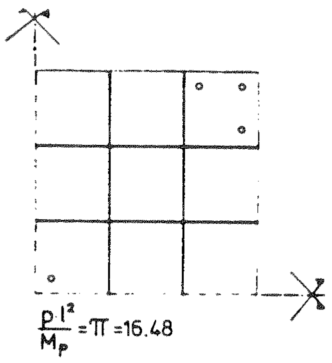
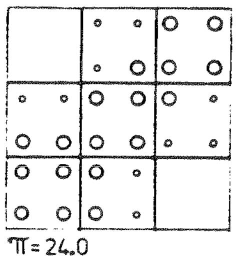
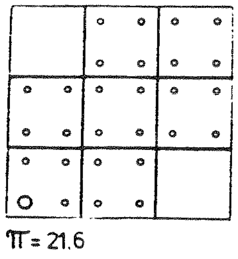
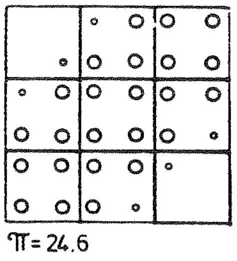


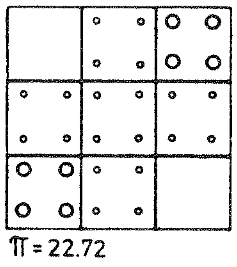
Figure 12 Load-deflection diagram at the plate center

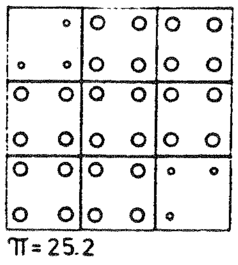


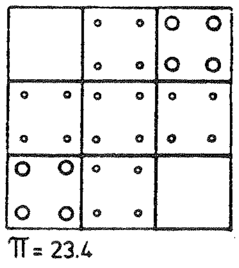


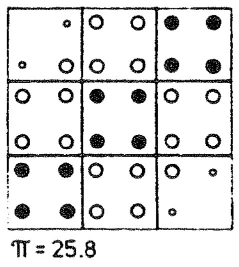












Notation :

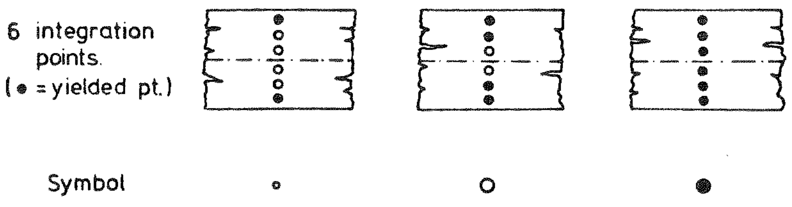


Figure 13 Spread of the yield regions

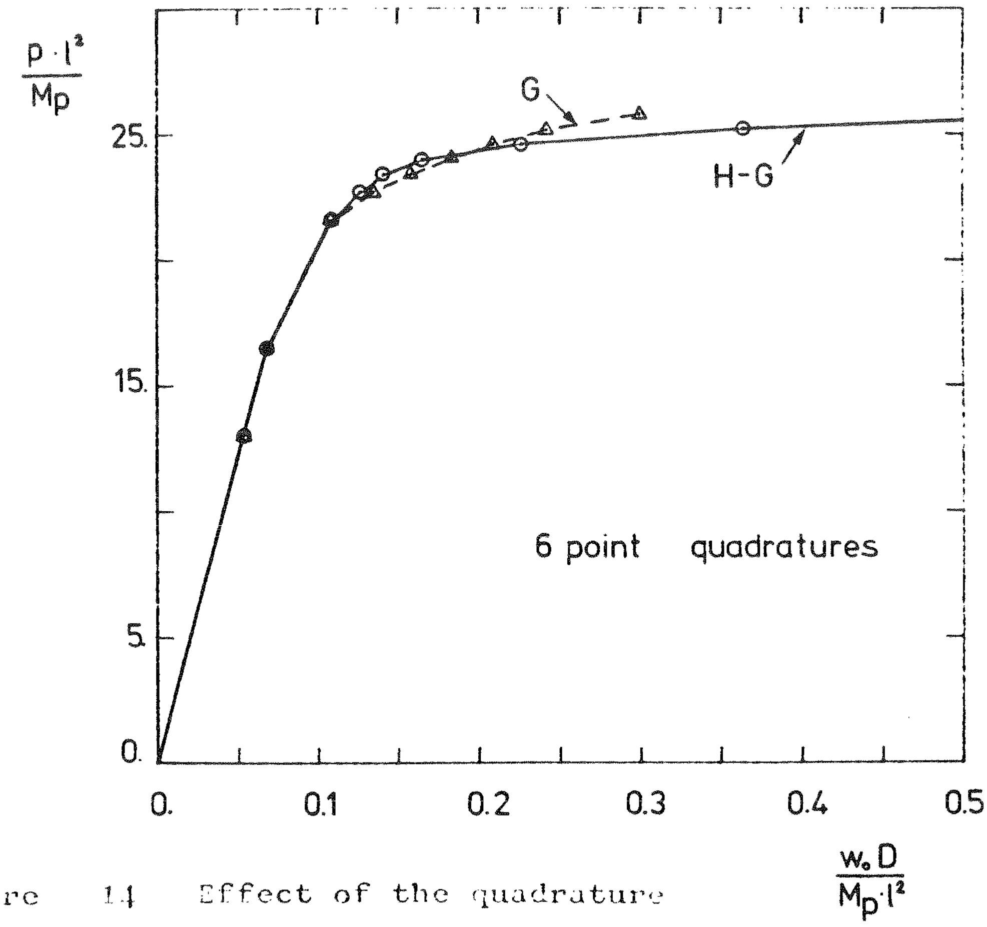


Figure 14 Effect of the quadrature on the square plate response

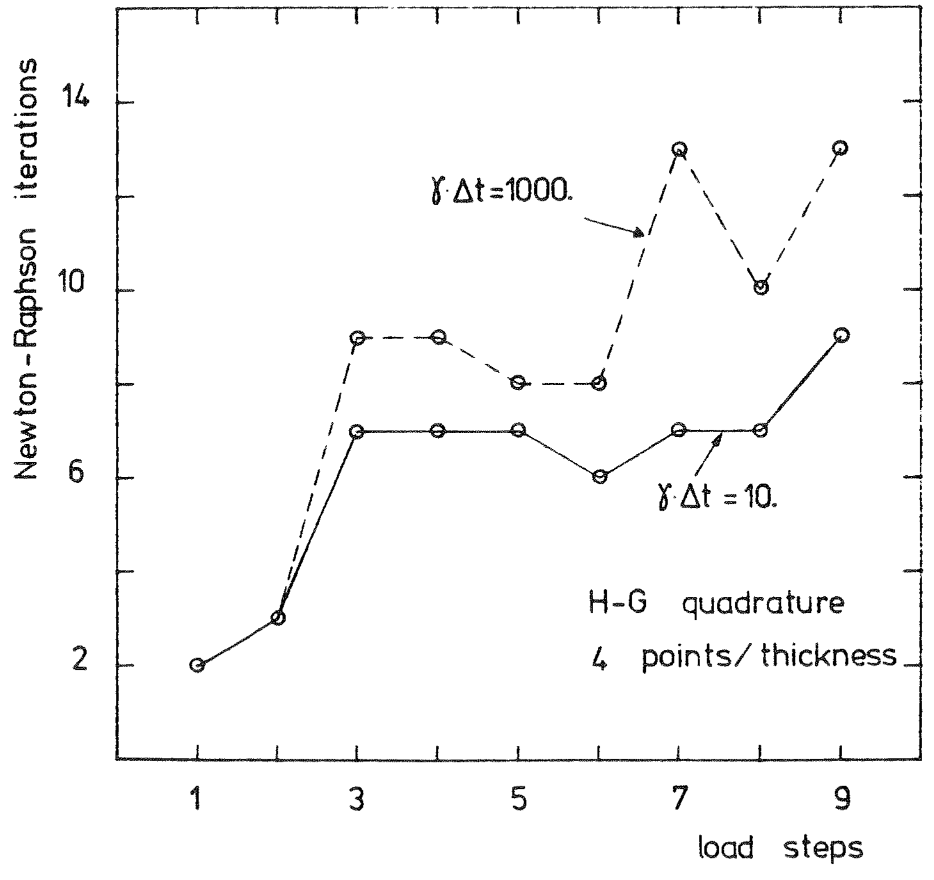


Figure 15 Effect of the material parameter on the number of N-R iterations

```
      subroutine elmt12(d,ul,xl,ix,tl,s,p,ndf,ndm,nst,isw)
c
c.... four node viscoplastic displacement model plate element
c.... with TJR Hughes displacement model shear stiffness
c.... using mindlin plate theory and von mises yield criterion
c.... Infinitesimal theory only (small deforma and displ)
c.... No membrane forces; antisymmetry in z: f(z)=-f(-z)
c
c.... input data:
c....                  young's modulus                                e
c....                  poisson's ratio                                xnu
c....                  mass density                                   d(4)
c....                  number of quadrature points (x-y)              l
c....                  number of points for stress output (x-y)       k
c....                  plate thickness                                thick
c....                  coeff of thermal expansion (not used)          alp
c....                  temperature reference (not used)               t0
c  (next card)
c....                  uniform pressure load (pos. upward)            q0
c....                  quadrature type (1=gauss-lobatto,2=gauss,
c....                            3=half-gauss)                        iparam
c....                  points thru the thickness (4 or 6)             lz
c....                  exponent for viscoplastic power law            kflow
c....                  mid-point rule coeff (0=explicit,1=implicit)   alpha
c....                  fluidity factor                                gamma
c....                  yield stress in tension                        yield
c.... default values:
c....                  l = 2
c....                  k = 1
c....                  iparam = 3
c....                  lz = 2
c....                  kflow = 1
c....                  alpha = 1.d0
c.... nodal dof:
c....                  1) rotation about x axis    theta x
c....                  2) rotation about y axis    theta y
c....                  3) transverse displacement  w
c.... interpolation:
c....                  displacement and rotations use bilinear functions
c.... integration:
c....                 >in the plane of the plate (x-y), Gauss-Legendre
c....                  quadrature is used. use at least 2x2 for the stiffness
c....                  and 2x2 for the stresses (since we have bilinear
c....                  interpolation functions, we gain nothing using
c....                  2x2 for the stresses. this should be modified)
c....                 >for the integration through the thickness, we can
c....                  use gauss-lobatto, gauss or half-gauss quadrature,
c....                  depending on input "iparam" (=1,2 or 3 respectively).
c....                  use 4 or 6 points ("lz"). if iparam=3, 4 points
c....                  are usually enough.
c....                  the number of points computed is half the input # due
c....                  to the antisymmetry sig(z)=-sig(-z).  hence linz=lz/2
c.... sign convention:
c....                  rcc coordinate system with x and y in plane of plate
c....                  stress resultants are determined according to positive
c....                  stress convention: a >0 stress in the + face
```

```
c....                produces a >0 stress resultant.
c.... list of routines:
c....                elmt12    driver of the element
c....                stifvp    computes the tangent stiffness + updates
c....                          the current stresses
c....                forcvp    computes the internal forces
c....                viscop       "    the viscoplastic state (gauss pt)
c....                momts        "    the stress resultants (at x-y gauss pt
c....                gaulob       "    the coord and weights (quadrature)
c....                pform   ### slight modif of Taylor's :
c....                        the common /bound/bcode is used in stifvp to che
c....                        whether or not we are computing applied B.C.
c.... the MACRO order is:
c
c....                dt  ,     ,xxxx
c....                loop,     ,i                 as many as load steps
c....                time
c....                loop,     ,j                 Newton-Raphson loops
c....                tang
c....                form
c....                solv
c....                next
c....                disp
c....                stre
c.... programmed by:
c....                Carlos Rodriguez
c....                January 1983 (UC Berkeley)
c - ----- -------- -- ---------------------------------------------
      implicit double precision (a-h,o-z)
c
      logical prt,debug
      logical mflg,rflg
      common /itcseq/maxit,tolceq
      common /debugp/debug,iparam
      common /cdata/ o,head(20),numnp,numel,nummat,nen,neq,ipr
      common /eldata/ dm,n,ma,mct,iel,nel
      common /print/ prt
      common /tdata/time,dt,c1,c2,c3,c4,c5
      common /mixed/ rm,mflg,rflg
      common /visco/nelem,npel,nstp,nste,nelemz,nelmax,iterno
      dimension d(1),ul(ndf,1),xl(ndm,1),ix(1),tl(1),s(nst,1),p(1)
     1  ,shp(3,9),sg(16),tg(16),wg(16),sig(6)
     2  ,zg(3),zwg(3),dnl(6),stk(13824)
      data nstp,nste/18,216/,nelemz/64/
      data tolceq/1.d-10/,maxit/20/
      data istart,iterno/0,0/
      data debug/.false./
c---------------------- ------------------------------------
c     modify nelemz and stk if >64 elemt12-type
c     dimension stk(nelemz*nste)
c------------------ -----------------------------------------
c.... go to correct array processor
      go to(1,2,3,4,5,6,7,2,9,2), isw
c.... input material properties
1     read(5,1000) e,xnu,d(4),l,k,thick,alp,t0
      read(5,1001) q0,iparam,lz,kflow,alpha,gamma,yield
      if (iparam.le.0) iparam=3
      if (l.le.0) l=2
      if (k.ne.1) k=1
      cappa = 5./6.
      d(1) = e/(1.-xnu*xnu)*thick**3/12.0
      d(2)=xnu*d(1)
      d(3) = e/24./(1.+xnu)*thick**3
      d(7) = e/2./(1.+xnu)*thick*cappa
      d(8) = q0
      l = min0(4,max0(-1,l))
      d(5) = l
      k = min0(4,max0(1,k))
      d(6) = k
      lint = 0
5001  write(6,2000) e,xnu,d(4),l,k,alp,t0,thick,q0
      write(6,2003) kflow,alpha,gamma,yield,iparam,lz,maxit,tolceq
      d(4) = d(4)*thick
      d(9) = t0
      d(10)= e*alp/(1.-xnu)*thick**3/12.
      d(11)=thick
      d(12)=e
      d(13)=xnu
      d(14)=lz
      d(15)=kflow
      d(16)=alpha
      d(17)=gamma
      d(18)=yield
      rflg=.true.
      if (istart.ne.0) return
      linz=0
      call pzero(stk,nelemz*nste)
      istart=1
      npel=numel
      nelmax=0
      return
2     return
c
c.... form tangent stiffness matrix
c
3     continue
      call stifvp (dt,d,xl,s,zg,zwg,dnl,stk,linz
     1             ,lint,shp,sg,tg,wg,ndm,ix,ndf,nst,ul)
      return
c
c.... compute element moments and strains
c     ---forced at the stiffness integration points---
c
 4    continue
c     define current element
      if (n.lt.npel) nelem=0
      if (numel.eq.1) nelem=0
      npel=n
      nelem=nelem+1
      ipos=(nelem-1)*nste
      l=d(5)
      lz=d(14)
      if(l*l.ne.lint) call pgauss(l,lint,sg,tg,wg)
      if (linz.ne.lz/2) call gaulob (lz,linz,zg,zwg)
      do 440 l = 1,lint
c.... compute element shape functions
      call shape(sg(l),tg(l),xl,shp,xsj,ndm,nel,ix,.false.)
c.... compute  coordinates
      xx =0.d0
      yy =0.d0
      do 420 j = 1,nel
      xx = xx + shp(3,j)*xl(1,j)
 420  yy = yy + shp(3,j)*xl(2,j)
c.... compute moments  (sig=mx,mxy,my)
      call momts (linz,zg,zwg,d(11),stk,l,   sig)
      call pstres(sig,sig(4),sig(5),sig(6))
c.... output moments and sigmas-strains thru thickness
      mct = mct - 2
      if(mct.gt.0) go to 430
      write(6,2001) o,head
      mct = 50
430   write(6,2002) n,ma,(sig(ii),ii=1,5),xx,yy,sig(6)
      mct=mct-linz
      do 431 j=1,linz
      zz=zg(j)*d(11)/2.
      if (iparam.eq.3) zz=(zg(j)+1.d0)*d(11)/4.
      write (6,2004) zz,(stk(ipos+ii),ii=1,6)
 431  ipos=ipos+nstp
440   continue
      return
c
c.... compute consistent mass matrix
5     l = d(5)
      if(l*l.ne.lint) call pgauss(l,lint,sg,tg,wg)
      do 500 l = 1,lint
c.... compute shape functions
      call shape(sg(l),tg(l),xl,shp,xsj,ndm,nel,ix,.false.)
      dv = wg(l)*xsj*d(4)
c.... for each node j compute db = rho*shape*dv
      j1 = 3
      do 500 j = 1,nel
      w11 = shp(3,j)*dv
c.... compute a lumped mass
      p(j1) = p(j1) + w11
c.... for each node k compute mass matrix (upper triangular part)
      k1 = j1
      do 510 k = j,nel
      s(j1,k1) = s(j1,k1) + shp(3,k)*w11
510   k1 = k1 + ndf
500   j1 = j1 + ndf
c.... compute missing parts and lower part by symmetries
      nsl = nel*ndf
      d11=d(11)**2/12.
      do 520 j = 1,nsl,ndf
      p(j) = p(j+2)*d11
      p(j+1) = p(j)
      do 520 k = j,nsl,ndf
      s(j,k) = s(j+2,k+2)*d11
      s(j+1,k+1) = s(j,k)
      s(k+1,j+1) = s(j,k)
      s(k+2,j+2) = s(j+2,k+2)
520   s(k,j) = s(j,k)
      return
c
c.... compute right hand side (forces)
c
6     continue
      call forcvp (d,xl,zg,zwg,stk,linz
     1             ,lint,shp,sg,tg,wg,ndm,ix,ndf,nst,ul,    p)
      return
7     call plot9(ix,xl,ndm,nel)
      return
c.... make the switch of sigma and eps for next time step
9     iterno=0
      l=d(5)
      lz=d(14)
      if(l*l.ne.lint) call pgauss(l,lint,sg,tg,wg)
      if (linz.ne.lz/2) call gaulob (lz,linz,zg,zwg)
      ipos=0
      nval=nelmax*lint*linz
      do 480 i=1,nval
      iposp=ipos+6
      do 475 j=1,6
 475  stk(iposp+j)=stk(ipos+j)
 480  ipos=ipos+nstp
      return
c.... formats for input-output
1000  format(3f10.0,2i10,3f10.0)
1001  format(f10.0,3i10,3f10.0,i10)
2000  format(/5x,'plate bending viscoplastic element'//
     1 10x,7hmodulus,e18.5/10x,13hpoisson ratio,f8.5/10x,
     2 7hdensity,e18.5/10x,13hgauss pts/dir,i3/10x,10hmoment pts,i6/
     3 10x,5halpha,e20.5/10x,9hbase temp,e16.5/
     4 10x,'thickness',e16.5/10x,'uniform load',e13.5/)
2001  format(a1,20a4//5x,16helement stresses//18h  element material
     1   ,3x,9h11-moment,3x,9h12-moment,3x,9h22-moment,4x,
     2   8h1-moment,4x,8h2-moment/2x,7h1-coord,2x,7h2-coord,3x,
     3   45x,5hangle/7x,7hz-coord,7x,8h11-sigma,4x,8h12-sigma,4x,
     4   8h22-sigma,17x,6h11-eps,4x,8h12-gamma,6x,6h22-eps)
 2002 format (/2i9,5e12.3/2f9.3,36x,f18.2)
 2003 format (10x,
     1 17hflow-law exponent,i13/10x,17halpha (time lin.),e13.3/
     2 10x,18hgamma (visco-law) ,e12.3/10x,
     3 18huniax yield stress,e12.3/10x,
     4 18hquadrature type   ,i12/
     5 10x, '   1.... Gauss-Lobatto'
     6/10x, '   2.... Gauss'
     7/10x, '   3.... Gauss (half thick)'
     8/10x,18hquadrature points ,i12/
     9 10x,  'Local convergence max iter',i4/
     1 10x,  '  "       "  tolerance',e10.3)

 2004 format (f14.3,4x,3e12.3,10x,3e12.3)
      end
      subroutine forcvp (d,xl,zg,zwg,stk,linz
     1                  ,lint,shp,sg,tg,wg,ndm,ix,ndf,nst,ul,p)
      implicit double precision (a-h,o-z)
c
c.... plate bending viscoplastic. element force routine
c.... computes internal forces
c
      common /cdata/ o,head(20),numnp,numel,nummat,nen,neq,ipr
      common /eldata/ dm,n,ma,mct,iel,nel
      common /visco/nelem,npel,nstp,nste,nelemz,nelmax,iterno
      common /prlod/prop
      dimension d(1),xl(ndm,1),ix(1),s(12,12),ul(ndf,1),p(1)
     1  ,shp(3,9),sg(16),tg(16),wg(16),bs(2,3,4)
     2  ,e1(4,2),e2(4,2),h(4),cs(4),sn2(4),g1(4,2),g2(4,2),g(4,2)
     4  ,zg(3),zwg(3),sig(3),stk(1)
c----------------------------------------------------------------
c     define current element
      if (n.lt.npel) nelem=0
      if (numel.eq.1) nelem=0
      npel=n
      nelem=nelem+1
      if (nelmax.lt.nelem) nelmax=nelem
      if (nelmax.le.nelemz) go to 100
      write (6,2000)
      stop
 100  continue
      l = d(5)
      lz= d(14)
      call pzero (s,144)
c.... form base vectors at nodes
      do 302 i = 1,nel
      j = mod(i,nel) + 1
      do 301 k = 1,2
301   e1(i,k) = xl(k,j) - xl(k,i)
      h(i) = sqrt(e1(i,1)**2+e1(i,2)**2)
      do 302 k = 1,2
      e1(i,k) = e1(i,k)/h(i)
302   e2(j,k) = -e1(i,k)
c.... form sines and cosine terms
      do 303 i = 1,nel
      cs(i) = e1(i,1)*e2(i,1) + e1(i,2)*e2(i,2)
303   sn2(i) = 1. - cs(i)*cs(i)
c.... form parts of g-vectors independent of space
      do 304 i = 1,nel
      j = mod(i,nel) + 1
      do 304 k = 1,2
      g1(i,k) = (e1(i,k) - cs(i)*e2(i,k))/sn2(i)
304   g2(i,k) = (e2(j,k) - cs(j)*e1(j,k))/sn2(j)
c
c.... force computation (first, loop over x-y plane)
      ipos=(nelem-1)*nste
      if(l*l.ne.lint) call pgauss(l,lint,sg,tg,wg)
      if (linz.ne.lz/2) call gaulob (lz,linz,zg,zwg)
      do 320 l = 1,lint
      call shape(sg(l),tg(l),xl,shp,xsj,ndm,nel,ix,.false.)
c
```

```
c....  integrate flexure forces thru thick (sig=moments mx,mxy,my)
       call nomts (linz,zg,zwg,d(11),stk,1,    sig)
c
c....  now define element shear stiffness, external forces
c                                           and flexure forces
       xsj = xsj*wg(1)
       fac = sqrt(d(7)*xsj)
       do 305 i = 1,nel
       j = mod(i,nel) + 1
       do 305 k = 1,2
305    g(i,k) = (shp(3,i)*g1(i,k) - shp(3,j)*g2(i,k))*fac
       do 306 i = 1,nel
       j = mod(i,nel) + 1
       do 306 k = 1,2
       bs(k,3,j) = g(i,k)/h(i) - g(j,k)/h(j)
       bs(k,1,j) = (g(i,k)*e2(j,1) - g(j,k)*e1(j,1))/2.
306    bs(k,2,j) = (g(i,k)*e2(j,2) - g(j,k)*e1(j,2))/2.
c....  loop over rows   (bdij = (b)t * d)
       j1 = 1
       do 320 j = 1,nel
       j2 = j1 - 1
c
c....  minus flexural forces
       p(j1  )=p(j1  ) - (shp(1,j)*sig(1)+shp(2,j)*sig(2))*xsj
       p(j1+1)=p(j1+1) - (shp(1,j)*sig(2)+shp(2,j)*sig(3))*xsj
c
c....  add external forces
       p(j1+2)=p(j1+2) + d(8)*shp(3,j)*xsj*prop
c....  loop over columns (symmetry noted) to define shear stiff
       k1 = j1
       do 310 k = j,nel
       k2 = k1 - 1
       do 307 jr = 1,3
       do 307 kc = 1,3
       do 307 i = 1,2
307    s(jr+j2,kc+k2) = s(jr+j2,kc+k2) + bs(i,jr,j)*bs(i,kc,k)
310    k1 = k1 + ndf
320    j1 = j1 + ndf
c....  form lower part by symmetry
       nsl = nel*ndf
       do 330 j = 1,nsl,ndf
       do 330 k = j,nsl,ndf
       s(k  ,j  ) = s(j  ,k  )
       s(k  ,j+1) = s(j+1,k  )
       s(k  ,j+2) = s(j+2,k  )
       s(k+1,j  ) = s(j  ,k+1)
       s(k+1,j+1) = s(j+1,k+1)
       s(k+1,j+2) = s(j+2,k+1)
       s(k+2,j  ) = s(j  ,k+2)
       s(k+2,j+1) = s(j+1,k+2)
       s(k+2,j+2) = s(j+2,k+2)
330    continue
c
c....  minus transverse shear forces
       do 340 i=1,nst
       do 340 j=1,nst
```

```
 340  p(i)=p(i) - s(i,j)*ul(j,1)
c
      return
 2000 format (////' (elmt12)- too many elements. increase'/
     1           '             nelemz  and  stk array')
      end
      subroutine gaulob (lz,linz,zg,wg)
      implicit double precision (a-h,o-z)
c
c.... quadrature points and weights
c     for iparam = 1... Gauss-Lobatto (symm. versus z)
c                  2 .. Gauss          (  "      "    )
c                  3... Gauss (full quadr. over half thick)
c
      logical debug
      common /debugp/debug,iparam
      dimension zg(3),wg(3)
c      ----------------------------------------------------
      linz=lz/2
      call pzero (zg,3)
      call pzero (wg,3)
      go to (1,2,3),iparam
c.... 6 Gauss-Lobatto points quadrature (symm. versus z)
1     if (lz.ne.6) go to 100
      sq7=dsqrt(7.0d0)
      zg(1)=1.0d0
      zg(2)=dsqrt((7.d0+2.d0*sq7)/21.d0)
      zg(3)=dsqrt((7.d0-2.d0*sq7)/21.d0)
      wg(1)=1.d0/15.d0
      wg(2)=(14.d0-sq7)/30.d0
      wg(3)=(14.d0+sq7)/30.d0
      return
 100  if (lz.ne.4) stop 'gaulob'
      zg(1)=1.0d0
      zg(2)=1.d0/dsqrt(5.d0)
      wg(1)=1.d0/6.d0
      wg(2)=5.d0/6.d0
      return
c.... Gauss quad. (symm. versus z)
2     if (lz.ne.6) go to 200
      zg(1)=0.932469514203152d0
      zg(2)=0.661209386466265d0
      zg(3)=0.238619186083197d0
      wg(1)=0.171324492379170d0
      wg(2)=0.360761573048139d0
      wg(3)=0.467913934572691d0
      return
200   if (lz.ne.4) stop 'gaulob'
      zg(1)=0.861136311594053d0
      zg(2)=0.339981043584856d0
      wg(1)=0.347854845137454d0
      wg(2)=0.652145154862546d0
      return
c.... Gauss full quad. on half thick
3     if (lz.ne.6) go to 300
      zg(1)=dsqrt(0.6d0)
      zg(2)=0.d0
      zg(3)=-zg(1)
      wg(1)=5.d0/9.d0
      wg(2)=8.d0/9.d0
      wg(3)=wg(1)
      return
300   if (lz.ne.4) stop 'gaulob'
      zg(1)=1.d0/dsqrt(3.d0)
      zg(2)=-zg(1)
      wg(1)=1.d0
      wg(2)=1.d0
      return
      end
      subroutine nomts (linz,zg,zwg,thick,stk,lg,    sig)
      implicit double precision (a-h,o-z)
c
c.... prog to compute stress resultants (sig=mx,my,mxy) at a
c     given gauss point (x,y) by integr thru thickness
c
      logical debug
      common /visco/nelem,npel,nstp,nste,nelemz,nelmax,iterno
      common /debugp/debug,iparam
      dimension zg(3),zwg(3),stk(1),sig(3)
c
c.... compute starting address of stresses (in stk) : ipos+1
c     for current gauss (x,y) point : lg
c
      ipos=(nelem-1)*nste +(lg-1)*linz*nstp
c
c     constant t2s2= t**2/4  *2 (symm)
      t2s2=thick**2/2.
      if (iparam.eq.3) t2s2=thick**2/8.d0
      call pzero (sig,3)
c
c.... integration through the thickness  (stk contains sigma)
c
      do 110 i=1,linz
      xsj=zg(i)*zwg(i)*t2s2
      if (iparam.eq.3) xsj=(zg(i)+1.d0)*zwg(i)*t2s2
      do 100 j=1,3
 100  sig(j)=sig(j) - stk(ipos+j)*xsj
 110  ipos=ipos+nstp
c
      return
      end
      subroutine stifvp (dt,d,xl,s,zg,zwg,dnl,stk,linz
     1                  ,lint,shp,sg,tg,wg,ndm,ix,ndf,nst,ul)
      implicit double precision (a-h,o-z)
c
c.... plate bending viscoplastic element stiffness routine
c.... computes the tangent stiffness
c
      logical bcode,debug
      common /debugp/debug,iparam
      common /bound/bcode
      common /cdata/ o,head(20),numnp,numel,nummat,nen,neq,ipr
      common /eldata/ dm,n,ma,mct,iel,nel
      common /visco/nelem,npel,nstp,nste,nelemz,nelmax,iterno
      dimension d(1),xl(ndm,1),ix(1),s(nst,1)
     1  ,shp(3,9),sg(16),tg(16),wg(16),bs(2,3,4)
     2  ,e1(4,2),e2(4,2),h(4),cs(4),sn2(4),g1(4,2),g2(4,2),g(4,2)
     4  ,zg(3),zwg(3),dnl(6),stk(1),eps(3),sig(3),ul(ndf,1)
      dimension istate(3,4)
c-------------------------------------------------------------------
      do 10 i=1,3
      do 10 j=1,4
 10   istate(i,j)=1h
c     define current element
      if (bcode) go to 550
      if (n.lt.npel) nelem=0
      if (numel.eq.1) nelem=0
      npel=n
      nelem=nelem+1
      if (nelmax.lt.nelem) nelmax=nelem
      if (nelmax.le.nelemz) go to 550
      write (6,2000)
      stop
  550 continue
      l = d(5)
      lz= d(14)
c.... form base vectors at nodes
      do 302 i = 1,nel
      j = mod(i,nel) + 1
      do 301 k = 1,2
301   e1(i,k) = xl(k,j) - xl(k,i)
      h(i) = sqrt(e1(i,1)**2+e1(i,2)**2)
      do 302 k = 1,2
      e1(i,k) = e1(i,k)/h(i)
302   e2(j,k) = -e1(i,k)
c.... form sines and cosine terms
      do 303 i = 1,nel
      cs(i) = e1(i,1)*e2(i,1) + e1(i,2)*e2(i,2)
303   sn2(i) = 1. - cs(i)*cs(i)
c.... form parts of g-vectors independent of space
      do 304 i = 1,nel
      j = mod(i,nel) + 1
      do 304 k = 1,2
      g1(i,k) = (e1(i,k) - cs(i)*e2(i,k))/sn2(i)
304   g2(i,k) = (e2(j,k) - cs(j)*e1(j,k))/sn2(j)
      if(l*l.ne.lint) call pgauss(l,lint,sg,tg,wg)
      if (linz.ne.lz/2) call gaulob (lz,linz,zg,zwg)
      call pzero (s,144)
      locitr=0
c.... stiffness computation , first loop over 'x-y' plane
      ipos=(nelem-1)*nste + 1
      iposm1=ipos-1
      t3s4=d(11)**3/4.
      if (iparam.eq.3) t3s4=d(11)**3/32.d0
      do 320 l = 1,lint
      call shape(sg(l),tg(l),xl,shp,xsj,ndm,nel,ix,.false.)
      call pzero (dnl,6)
c.... skip calculation of sig, eps and call viscop when computing
```

```
c     the forces for the boundary conditions (bcode=.true.)
      if (bcode) go to 426
c
c.... compute strains through the thickness
      do 425 ith=1,linz
      call pzero (eps,3)
      zz=zg(ith)*d(11)/2.
      if (iparam.eq.3) zz=(zg(ith)+1.d0)*d(11)/4.d0
      do 420 j = 1,nel
      eps(1) = eps(1) - shp(1,j)*ul(1,j)
      eps(3) = eps(3) - shp(2,j)*ul(2,j)
420   eps(2) = eps(2) - shp(1,j)*ul(2,j) - shp(2,j)*ul(1,j)
c...  increment of strain (substract previous eps)
      ieps=iposm1+3
      do 421 j=1,3
 421  eps(j)=eps(j)*zz -stk(ieps+j)
c.... compute stress correction (dsig=dnl*deps)
      idnl=iposm1+12
      sig(1)=stk(idnl+1)*eps(1)+stk(idnl+2)*eps(2)+stk(idnl+4)*eps(3)
      sig(2)=stk(idnl+2)*eps(1)+stk(idnl+3)*eps(2)+stk(idnl+5)*eps(3)
      sig(3)=stk(idnl+4)*eps(1)+stk(idnl+5)*eps(2)+stk(idnl+6)*eps(3)
c.... update sigma and eps for current iteration
      do 422 j=1,3
      stk(iposm1+j)=stk(iposm1+j) + sig(j)
 422  stk(ieps+j)=stk(ieps+j) + eps(j)
 425  iposm1=iposm1+nstp
 426  continue
c
c.... integration through the thickness
c
      do 645 i=1,linz
      zgwg=zg(i)**2*zwg(i)
      if (iparam.eq.3) zgwg=(zg(i)+1.d0)**2*zwg(i)
c     call viscoplastic routine to get sbar,dtg (stored in stk array)
      kflow=d(15)
      if (.not.bcode)
     1   call viscop (d(16),d(17),kflow,d(18),dt,d(12),d(13)
     1             ,stk(ipos),stk(ipos+3),stk(ipos+6),stk(ipos+9)
     1             ,stk(ipos+12),nelem,l,i,istate(i,l),iterno,locitr)
      do 565 j=1,6
      jj=ipos+11+j
 565  dnl(j)=dnl(j)+zgwg*stk(jj)
 645  ipos=ipos+nstp
c     mult dnl  by t**3/8  & by 2 for symm
c     or by t**3/32 for iparam=3
      do 566 ij=1,6
566   dnl(ij)= dnl(ij)*t3s4
c
c.... now define element stiffness
      xsj = xsj*wg(l)
      fac = sqrt(d(7)*xsj)
      do 305 i = 1,nel
      j = mod(i,nel) + 1
      do 305 k = 1,2
305   g(i,k) = (shp(3,i)*g1(i,k) - shp(3,j)*g2(i,k))*fac
      do 306 i = 1,nel
      j = mod(i,nel) + 1
      do 306 k = 1,2
      bs(k,3,j) = g(i,k)/h(i) - g(j,k)/h(j)
      bs(k,1,j) = (g(i,k)*e2(j,1) - g(j,k)*e1(j,1))/2.
306   bs(k,2,j) = (g(i,k)*e2(j,2) - g(j,k)*e1(j,2))/2.
c.... loop over rows   (bdij = (b)t * d)
      j1 = 1
      do 320 j = 1,nel
      j2 = j1 - 1
      bd11=(dnl(1)*shp(1,j) + dnl(2)*shp(2,j))*xsj
      bd12=(dnl(2)*shp(1,j) + dnl(3)*shp(2,j))*xsj
      bd13=(dnl(4)*shp(1,j) + dnl(5)*shp(2,j))*xsj
      bd21=(dnl(2)*shp(1,j) + dnl(4)*shp(2,j))*xsj
      bd22=(dnl(3)*shp(1,j) + dnl(5)*shp(2,j))*xsj
      bd23=(dnl(5)*shp(1,j) + dnl(6)*shp(2,j))*xsj
c.... loop over columns (symmetry noted)
      k1 = j1
      do 310 k = j,nel
      k2 = k1 - 1
      do 307 jr = 1,3
      do 307 kc = 1,3
      do 307 i = 1,2
307   s(jr+j2,kc+k2) = s(jr+j2,kc+k2) + bs(i,jr,j)*bs(i,kc,k)
      s(j1  ,k1  ) = s(j1  ,k1  ) + bd11*shp(1,k) + bd12*shp(2,k)
      s(j1  ,k1+1) = s(j1  ,k1+1) + bd12*shp(1,k) + bd13*shp(2,k)
      s(j1+1,k1  ) = s(j1+1,k1  ) + bd21*shp(1,k) + bd22*shp(2,k)
      s(j1+1,k1+1) = s(j1+1,k1+1) + bd22*shp(1,k) + bd23*shp(2,k)
310   k1 = k1 + ndf
320   j1 = j1 + ndf
c.... form lower part by symmetry
      nsl = nel*ndf
      do 330 j = 1,nsl,ndf
      do 330 k = j,nsl,ndf
      s(k  ,j  ) = s(j  ,k  )
      s(k  ,j+1) = s(j+1,k  )
      s(k  ,j+2) = s(j+2,k  )
      s(k+1,j  ) = s(j  ,k+1)
      s(k+1,j+1) = s(j+1,k+1)
      s(k+1,j+2) = s(j+2,k+1)
      s(k+2,j  ) = s(j  ,k+2)
      s(k+2,j+1) = s(j+1,k+2)
      s(k+2,j+2) = s(j+2,k+2)
330   continue
c
c...   write state of element
      if (bcode) return
      if (nelem.eq.1) write (6,2001)
      avloc=1.d0*locitr/(lint*linz)
      write (6,2002) n,istate,avloc
      return
 2000 format (///' (elmt12)-too many elements. increase'/
     1            '                nelemz  and  stk array')
 2001 format(//' state of elemts (e=elastic, p=plastic) and local'
     1 ,' iter average')
 2002 format(1x,' element',i4,'   state :    ',4(3a1,2x),
     1 '         loc it av.',f4.1)
```

```
      end
      subroutine viscop (alpha,gamma,k,y,dt,yg,xnu,snp1,e1,s,e,dtg
     1                  ,nelem,ngauss,nlobat,istate,iterno,locitr)
c
c     program to compute phi,d tangent and sigma bar
c     at given integration point
c     stresses and strains are stored : x,xy,y
c...  WITH CONSTITUTIVE LINEARIZATION (iterations till phi=0)
c
      implicit double precision (a-h,o-z)
      logical debug
      common /itcseq/maxit,tolceq
      common /debugp/debug,iparam
      dimension s1(3),e1(3),s(3),e(3),dtg(6),snp1(3),phi(3)
c
      data rm11,rm13,rm22/1.d0,-0.5d0,3.d0/
      data phimax/0.d0/
c      ---------- - -- ----------------------------------------
c
c     define current iteration of the time step
      if (nelem*ngauss*nlobat.ne.1) go to 10
      iterno=iterno+1
      write (6,2001) iterno
 10   continue
      do 15 i=1,3
15    s1(i)=snp1(i)
c.... for first iter of each time step, we want a linear behavior
      if (iterno.eq.1) go to 100
      iter=0
20    iter=iter+1
      if (iter.le.maxit) go to 25
      go to 80
25    continue
c     compute stresses at time tn+alpha
      sx  =(1.d0-alpha)*s(1) + alpha*s1(1)
      sxy =(1.d0-alpha)*s(2) + alpha*s1(2)
      sy  =(1.d0-alpha)*s(3) + alpha*s1(3)
c
c     3* 2nd deviatoric stress invariant + its square root
c                       (at time tn+alpha)
      tj2=sx*sx + sy*sy - sx*sy +3.d0*sxy*sxy
      stj2=dsqrt(tj2)
c
c     test if the point is nonlinear
c
      val=stj2/y -1.d0
      if (val.le.0.d0) go to 100
c...  point remains plastic once plastic
c
c.... point is nonlinear
c     *****************
c
      istate=1hp
c
c     compute oper and oper* at time tn+alpha
c
```

```
      oper=(val**k)/stj2
      opstar=(val**(k-1))*((k-1)/y+1.d0/stj2)/tj2
c
c     compute phi at time tn+1   (product m*s at time tn+alpha)
c
      t1=rm11*sx + rm13*sy
      t3=rm13*sx + rm11*sy
      t2=rm22*sxy
      fac=gamma*dt*oper
      phi(1)=e1(1)-e(1) -(s1(1)-s(1)-xnu*(s1(3)-s(3)))/yg -fac*t1
      phi(3)=e1(3)-e(3) -(s1(3)-s(3)-xnu*(s1(1)-s(1)))/yg -fac*t3
      phi(2)=e1(2)-e(2) -2.d0*(1.d0+xnu)*(s1(2)-s(2))/yg  -fac*t2
c
c     compute c tangent (symm) at time tn+alpha  (ti*tj = m*s*(s)t*m)
c
      fac=gamma*dt*alpha
      c11=          1.d0/yg    +fac*(oper*rm11 + opstar*t1*t1)
      c12=                      fac*(            opstar*t1*t2)
      c13=          -xnu/yg    +fac*(oper*rm13 + opstar*t1*t3)
      c22=2.d0*(1.d0+xnu)/yg +fac*(oper*rm22 + opstar*t2*t2)
      c23=                      fac*(            opstar*t2*t3)
      c33=          1.d0/yg    +fac*(oper*rm11 + opstar*t3*t3)
c
c     compute d tangent at time tn+alpha (stored : 11,12,22,13,23,33)
c                   (by inversion of c tg)
c
      det=c11*c22*c33 +2.d0*c12*c23*c13 -c13*c22*c13 -c23*c23*c11
     1    -c33*c12*c12
      dtg(1)=(c22*c33-c23*c23)/det
      dtg(2)=(c13*c23-c12*c33)/det
      dtg(3)=(c11*c33-c13*c13)/det
      dtg(4)=(c12*c23-c13*c22)/det
      dtg(5)=(c12*c13-c11*c23)/det
      dtg(6)=(c11*c22-c12*c12)/det
c
c     compute sigma bar at time tn+1  (predictor for iter i+1)
c
      ds1= dtg(1)*phi(1) + dtg(2)*phi(2) + dtg(4)*phi(3)
      ds2= dtg(2)*phi(1) + dtg(3)*phi(2) + dtg(5)*phi(3)
      ds3= dtg(4)*phi(1) + dtg(5)*phi(2) + dtg(6)*phi(3)
      s1(1)=s1(1)+ds1
      s1(2)=s1(2)+ds2
      s1(3)=s1(3)+ds3
c
c.... check convergence
      locitr=locitr+1
      phicur=dot(phi,phi,3)
      if(phimax.lt.phicur) phimax=phicur
      if(phicur.gt.phimax*tolceq) go to 20
80    continue
      do 90 i=1,3
90    snp1(i)=s1(i)
      go to 200
c
100   istate=1he
101   continue
```

```
c
c....  point is linear
c      ***************
c
       locitr=locitr+1
       fac=yg/(1.d0-xnu**2)
       dtg(1)= fac
       dtg(2)= 0.d0
       dtg(3)= fac*(1.d0-xnu)/2.d0
       dtg(4)= fac*xnu
       dtg(5)= 0.d0
       dtg(6)= fac
c
       phi(1)=e1(1)-e(1) -(snp1(1)-s(1)-xnu*(snp1(3)-s(3)))/yg
       phi(3)=e1(3)-e(3) -(snp1(3)-s(3)-xnu*(snp1(1)-s(1)))/yg
       phi(2)=e1(2)-e(2) -2.d0*(1.d0+xnu)*(snp1(2)-s(2))/yg
c
c      compute sigma bar at time tn+1  (predictor for iter i+1)
c
       snp1(1)=snp1(1) + dtg(1)*phi(1) + dtg(2)*phi(2) + dtg(4)*phi(3)
       snp1(2)=snp1(2) + dtg(2)*phi(1) + dtg(3)*phi(2) + dtg(5)*phi(3)
       snp1(3)=snp1(3) + dtg(4)*phi(1) + dtg(5)*phi(2) + dtg(6)*phi(3)
c
 200   continue
       return
 2001  format (/' #### iteration =',i3)
       end
```