TIME INTEGRATION SCHEMES FOR RATE DEPENDENT ELASTO-PLASTIC CONSTITUTIVE EQUATIONS

by
Derek Hulley
BSc Eng (Mechanical)

Department of Mechanical Engineering
University of Cape Town

A thesis submitted in partial fulfilment of the requirements for the degree of
Master of Science in Engineering

January 1997

Declaration

I hereby declare that the modelling, calculations and results presented in this thesis are essentially my own work and that no part of this work has been submitted for a degree at any other university

Derek Hulley
January 1997

Signed by candidate

Signature Removed

Acknowledgements

I would like to thank the following parties for their assistance in completing this work:

Dr Greg Mitchell[1] for supervising this thesis.

FRD and CERECAM for their financial assistance.

Atomic Energy Corporation for granting me leave to complete my MSc.

[1] FRD/UCT Centre for Research in Computational and Applied Mechanics

SYNOPSIS

The purpose of this thesis is to set out the results of an investigation into the commonly used methods of performing material update calculations within the framework of the Finite Element Method, as well as an investigation into possible new methods of performing the material update procedures within the context of a rate dependent plastic material obeying the Von Mises yield condition. Material update procedures which have been used and analysed frequently are the Generalised Midpoint Algorithm, including the Midpoint Method, the Trapezoidal Rule and the Backward Euler Method with Radial Return. Each method displays its own advantages when applied to different input parameters (being material properties, initial stresses and strains, and increments in time and strain).

It was found that the Trapezoidal Rule could not perform as well as the other methods for most initial conditions, and that it should be disregarded as a means of analysing rate dependent material models obeying the Von Mises yield condition. The Midpoint Rule, it was found, could lead to vast under or over prediction of the yield surface radius, except for small increments. For small increments, the Midpoint Rule has the advantage that it ensures second order accuracy of the solution. The standard Backward Euler Method with Radial Return is more accurate in general than the aforementioned algorithms, but does not ensure second order accuracy for small increments. It was also found that the optimal algorithm should separate the yield surface radius calculations from the change in angle of the stress point in deviatoric space. The error in the yield surface given by the Backward Euler Method with Radial Return is significantly lower than that given by the Midpoint Rule, but lacks the second order accuracy for small increments.

A new algorithm which splits the stress update procedure into calculation of the yield surface, and calculation of the angular change of the stress point in deviatoric space is investigated. Based on a precise calculation of the change in angle devised by Krieg and Krieg for rate independent perfect plasticity, a method is devised which dynamically chooses the return direction from which to make a radial return. An explicit approximation is also found to dynamically predict the collation point, thus giving the algorithm the name - the Optimised Midpoint Method with Variable Return. It was found that the OMM gave significantly better results than all other methods, particularly in predicting the change in angle of the stress point. The improvement over the Backward Euler Method with Radial Return in predicting the radius of the yield surface was not significant, but the OMM did result in utilisation of the Midpoint Method when small increments were used. The computational effort involved in dynamically predicting a collation point and the return angle is not significantly greater than when these two are assumed to coincide at a fixed value.

Part of the derivation of the OMM resulted in an additional material parameter, a reference strain rate, being required. The determination of this parameter adds additional initial effort to obtaining an accurate solution and it was therefore recommended that the OMM be further investigated in order to eliminate the need for this parameter, which affects the results significantly.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

The Finite Element Method (FEM) is now well established as the preferred method for the computational simulation of a wide range of applications, such as stress analysis and metal forming. This is mainly due to the modular nature and straight forward discretisation procedures that are a feature of the FEM.

The choice of element (the basic building block of the FEM), characterised by interpolation functions, determines to a large extent the quality of the solution. Whatever method is used to derive the set of FEM equations (see Appendix C), the end result is a set of simultaneous equations that must be solved using standard techniques (Newton-Raphson, indirect solution schemes).

The solution of the simultaneous equations is usually (for displacement based FEM) a vector of nodal displacements. From these the strains and stresses must be calculated. How this is done depends on the class of problem being analysed. A linear elastic problem, for example, is solved directly using a few simple formulae.

In the context of material non-linearity (especially elasto-plastic problems) this calculation requires the integration of the constitutive equations which are usually cast in a rate form. The integration of these equations is completed by dividing the problem into a discrete number of time increments, and calculating the increments in stress and strain (elastic and plastic). For rate independent problems, the time incrementation is only used as a ratio of completion of the problem, i.e. pseudo time. For rate dependent problems, the time incrementation has to occur in real time.

Although the solution of the global system of simultaneous equations can be speeded up by having a more accurate consistent tangent modulus, it does not play an important role in the accuracy of the solution. It is the constitutive equations, as applied to each element integration point, which determine the accuracy of the stresses and strains predicted.

The integration of the constitutive equations has to contend with potentially highly non-linear functions, with only the strain increment, the time increment and the starting values being known. One such non-linear situation, is the solution of rate dependent material constitutive equations obeying the Von Mises yield criterion. The underlying material may itself exhibit non-linear hardening. The challenge of developing an integration scheme which can deal with these problems has resulted in the conception of several schemes which have been analysed for their accuracy and efficiency many times. In many cases, certain schemes are best applied to specific conditions, while performing poorly under others. Some of the more popular schemes are the Backward Euler Method with Radial Return, the Generalised Midpoint Method (including the Midpoint Method) and the Trapezoidal Rule.[1,2,3,4,5,6,7,8,9,10,13,14,15, 18,19]

The objectives of this thesis are:

1. To describe the framework, of the FEM, into which the material update algorithms are applied in order to understand the necessity of calculating the various output parameters.
2. To investigate some of the popular material update algorithms and to highlight their shortcomings and advantages.
3. To introduce a new algorithm recently proposed by P.A. Fotiu,[15] the Optimised Midpoint Method with Variable Return (OMM), which attempts to incorporate the advantages of the popular algorithms in use today into one single material update procedure.
4. To extend the series of tests performed on the OMM to include the widely used Isoerror maps, and to produce test results which are directly applicable to the FEM.
5. To compare the accuracy of this proposed algorithm against some well used procedures.

We will first see how the stress update procedures and calculation of the consistent tangent modulii fit into the broader scheme of the FEM. Some standard techniques for handling rate dependent Von Mises yielding will be examined before looking at some of the common methods of updating stresses in a body, given material data and strain history. We will initially concentrate on the traditional Backward Euler Method with Radial Return, before being introduced to a relatively new algorithm, the OMM. The range of tests originally performed on the OMM by P.A. Fotiu[15] will be extended to include a wide range of isoerror maps using dimensioned variables, and these will be compared to the traditionally used algorithms.

2. BASIC RELATIONS

Appendix C gives a brief overview of the framework of the FEM into which the subsequent work will be applied, and briefly shows how the material calculations fit into a displacement based Finite Element Analysis.

2.1 General Elasticity and Plasticity

The realm of elastic material response has been examined in many works and most applications of materials such as structural metals can be modelled using purely elastic, temperature independent relationships.[11] Adverse or extreme conditions of loading could however occur, and it is necessary to find the response of materials to these loads. However, the realm of plasticity has become more popular as a legitimate design tool, and much work has been dedicated to plasticity orientated design.[12]

The advantages of designing metal components to strain plastically are:

1. Most metals continue to harden as they start to exhibit plastic strain. This increase in strength can be used by allowing components, which do not need to unload, to achieve elevated strength with less material use.
2. The plastic deformation in metals serves to absorb large amounts of energy, dissipating it in other forms such as heat. This has made the solution of plasticity problems very popular in fields involving impact, such as the motor vehicle industry.
3. Most ductile metals display hardening in response to large strain rates[11] and this property can be utilised in several applications, such as the deep drawing of sheet metal. In this application, the use of high strain rates allows the metal to withstand the severe loading experienced.

Since we are primarily concerned with the behaviour of metals for the purposes of this study, we limit ourselves to the class of material which displays small strains until yield, thereafter exhibiting large strains. Before we examine the principles behind plastic strains, we need to know the basic elastic constitutive behaviour characterising most metals.

Since the elastic modulus of metals is typically three orders of magnitude greater than the yield stress, elastic strains are usually small, of the order of 10^{-3}.[11] This means that any small non-linearities which result from high strain rates, temperatures, etc. are negligible, which allows us to break the strain, and strain rate into an elastic $(\sim)^{el}$ and plastic $(\sim)^{pl}$ component for infinitesimal strain[13]. This is known as the additive strain rate decomposition.[11,14]

$$\dot{\varepsilon}_{ij} = \dot{\varepsilon}_{ij}^{el} + \dot{\varepsilon}_{ij}^{pl} \qquad 2.1.1$$

Assuming the body under scrutiny begins at a state of zero stress and strain, and that the material has not yielded:

$$\dot{\sigma}_{ij} = D^{el}_{ijkl}\dot{\varepsilon}^{el}_{kl} \qquad 2.1.2$$

In many instances, the stresses and strains can be further divided into volumetric and non-volumetric components. The non-volumetric components of stress and strain are known as *deviatoric* stresses and strains. These can be extracted from the stress and strain tensors as follows:

$$\dot{S}_{ij} = \dot{\sigma}_{ij} - \frac{1}{3}\dot{\sigma}_{kk}\delta_{ij} \qquad 2.1.3$$

$$\dot{e}_{ij} = \dot{\varepsilon}_{ij} - \frac{1}{3}\dot{\varepsilon}_{kk}\delta_{ij} \qquad 2.1.4$$

where δ_{ij} is known as the Kronecker delta and:

$$\begin{aligned} \delta_{ij} &= 1 \quad \text{if} \quad i = j \\ \delta_{ij} &= 0 \quad \text{if} \quad i \neq j \end{aligned} \qquad 2.1.5$$

The volumetric and deviatoric stresses are related to the strains by:

$$\dot{\sigma}_{kk} = K\dot{\varepsilon}_{kk} \qquad 2.1.6$$

$$\dot{S}_{ij} = 2G\dot{e}_{ij} \qquad 2.1.7$$

Where $\dot{S}_{ij}$ and $\dot{e}_{ij}$ are deviatoric components of stress and strain rate respectively, and the shear and bulk modulii are defined by:

$$K = \frac{E}{3(1-2\nu)} \qquad G = \frac{E}{2(1+\nu)} \qquad 2.1.8$$

The bulk modulus K quantifies the relation between volumetric strains and volumetric stress, while the shear modulus G defines the relation between deviatoric strains and deviatoric stress in the elastic domain.

So in the case of zero plastic strain, Equation 2.1.2 to Equation 2.1.7 can fully define the stress update and tangent modulus. In the case of plastic strain, the amount of stress developed depends only on the proportion of elastic strain. Plastic strain gives rise to no stress at all.

We can define a yield function f which allows us to determine if yielding has occurred or not:

$$f \rightarrow f\left(\sigma_{ij}, \kappa\right) \qquad 2.1.9$$

And κ is determined by the material properties of the body, such as the yield stress or temperature.

We can also define a plastic flow potential g such that:

$$\dot{e}_{ij}^{pl} = \dot{\lambda}\frac{\partial g}{\partial S_{ij}} \qquad 2.1.10$$

Where λ is a plastic multiplier acting on the normal to the surface g in stress space. We will focus on associative flow, which assumes colinearity between the plastic strain rate and the yield function ($g \equiv f$),[15,11] i.e. the plastic strain occurs normal to the yield surface for each infinitesimal strain increment.

$$\dot{e}_{ij} = \dot{\lambda}\frac{\partial f}{\partial S_{ij}} \qquad 2.1.11$$

The conditions for determining whether plastic loading, elastic loading, plastic unloading or elastic unloading occur, can be expressed in terms of the yield function and a plastic multiplier:[16]

$$\left.\begin{array}{l} f \le 0 \\ \lambda \ge 0 \\ f\lambda = 0 \end{array}\right\} \qquad 2.1.12$$

Each yield condition has its own yield function which controls the development of stress as a result of strain. One such condition is the Von Mises yield criterion.

2.2 The Von Mises Yield Criterion

The Von Mises yield criterion is commonly used in engineering applications for yield in ductile metals.[14]

The standard Von Mises condition assumes that the volumetric components of strain remain elastic regardless of loading conditions, thus making yield calculations necessary for only the deviatoric components S_{ij} and e_{ij}.

The Von Mises yield criterion fall into the general group of J_2 elasto-plasticity (J_2 is the second invariant of the stress tensor), where J_2 as a measure of the equivalent stresses and strains.[16] This is particularly useful in the sense that it allows the gathering of all the yield data from simple single stress tests (tensile, compressive, pure torsion) which provide nominal stress-strain data.[17] A typical Von Mises yield function will be of the form:

$$f = q - \overline{S} \qquad 2.2.1$$

where $\overline{S}$ is the maximum allowable equivalent stress. An equivalent deviatoric stress q can be calculated from the state of stress at any time, as well as equivalent total, elastic and plastic strains.

$$q = \sqrt{\frac{3}{2} S_{ij} S_{ij}} \quad 2.2.2$$

$$\overline{e} = \sqrt{\frac{2}{3} e_{ij} e_{ij}} \qquad \overline{e}^{el} = \sqrt{\frac{2}{3} e_{ij}^{el} e_{ij}^{el}} \qquad \overline{e}^{pl} = \sqrt{\frac{2}{3} e_{ij}^{pl} e_{ij}^{pl}} \quad 2.2.3$$

where the plastic strain is cumulative, i.e.:

$$\overline{e}^{pl} = \int_0^t \dot{\overline{e}}^{pl}\, dt \quad 2.2.4$$

Depending on the requirements of the solution algorithm, the yield stress may be given as:

$$H \rightarrow H(\overline{e}) \quad 2.2.5$$

$$H \rightarrow H(\overline{e}^{pl}) \quad 2.2.6$$

Where H denotes the yield stress. Obtaining data for Equation 2.2.6 involves converting nominal stress-nominal strain curves to true stress-logarithmic (true) strain curves.[12,11,17]

$$\sigma_{true} = \sigma_{nominal}(1 + \varepsilon_{nominal}) \quad 2.2.7$$

$$\varepsilon_{ln} = \ln(1 + \varepsilon_{nominal}) \quad 2.2.8$$

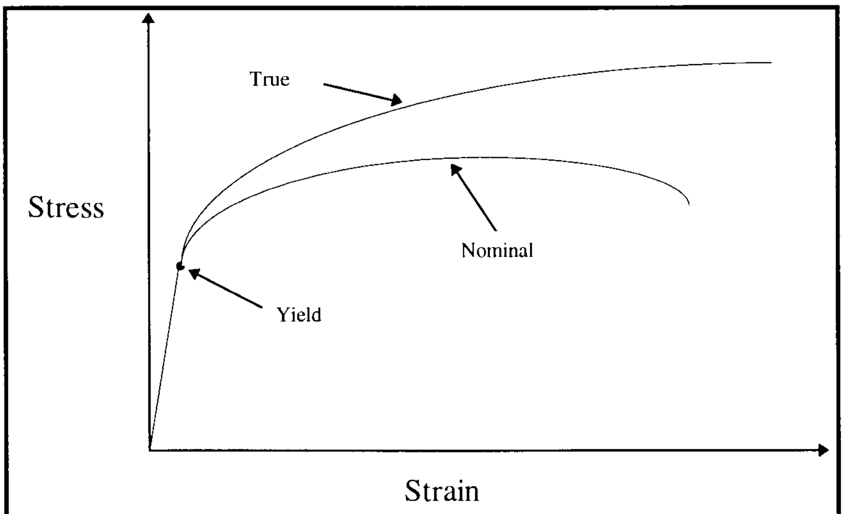

Figure 2.2.1 Comparison of nominal and true stress-strain curves.[12]

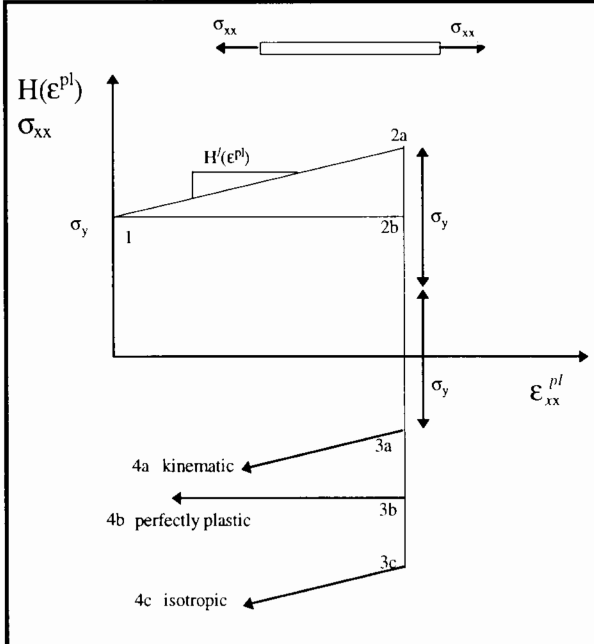

Figure 2.2.2 Stress versus plastic strain in uniaxial stress.[18]

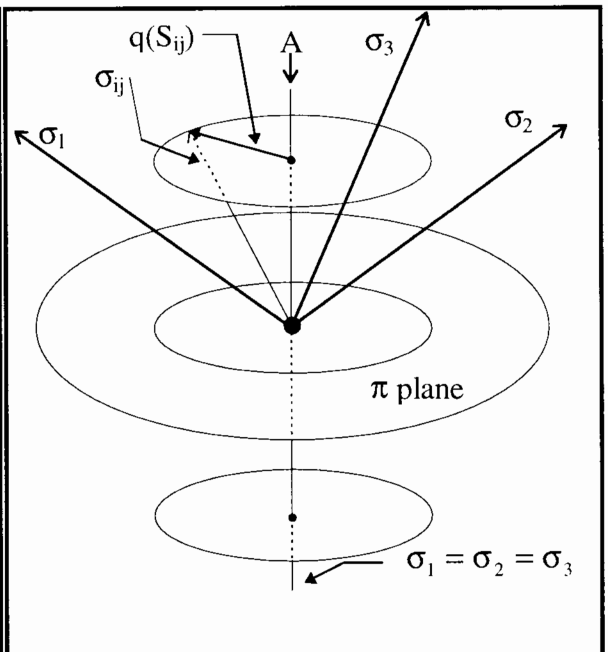

Figure 2.2.3 Geometric representation of state of stress on the yield surface on the π plane[17]

Figure 2.2.2 shows the idealised plastic behaviour of three types of materials. Figures in { } refer to the path of the stress-strain response in Figure 2.2.2:

1. Perfect plasticity {1,2b,3b,4b}
2. Isotropic hardening (linear or non-linear) {1,2a,3c,4c}
3. Kinematic Hardening {1,2a,3a,4a}

The removal of volumetric components from the yield condition leads to the formulation of a yield surface in the π plane. The yield surface can travel up and down axis A (the line of equal principal stress in Figure 2.2.3) without affecting the yield condition according to the Von Mises equations. All representations of yielding can thus be shown from the point of A, i.e. a circular yield surface in 2D space.

Figure 2.2.4 shows how f varies as a result of loading using the basic plasticity material models. The Bauschinger effect is observed primarily during kinematic hardening. The centre of the yield surface displaces such that the zero reference state moves relative to its original position. The centre of the yield surface is known as the back stress B_{ij}. It is therefore possible to eliminate first the volumetric and then the back stress components from the total stress, and to perform the stress update for each of these components separately.

$$S_{ij} = S^B_{ij} - B_{ij} \qquad 2.2.9$$

and S^B_{ij} is the deviatoric stress including the components of the back stress.

From Equation 2.1.11, the unit normal direction vector at **P** in Figure 2.2.4 gives the direction of plastic flow at that point. Another important assumption is that the strain rate remains constant throughout the increment. We define the direction of strain as:

$$\eta_{ij} = \frac{\dot{e}_{ij}}{\dot{\bar{e}}} \qquad 2.2.10$$

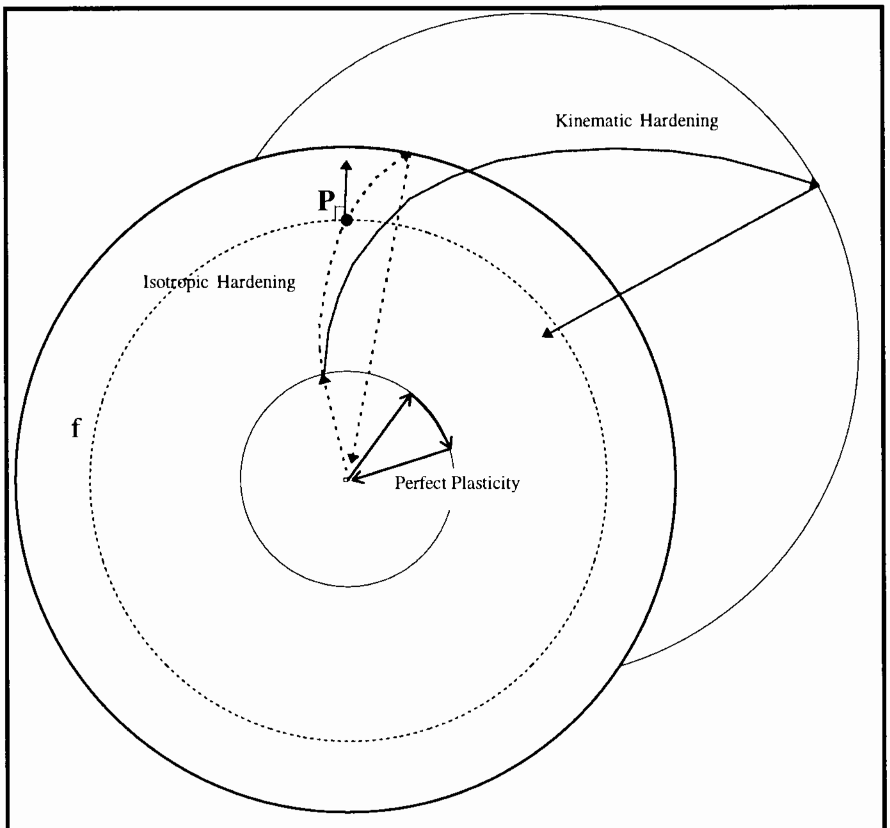

Figure 2.2.4 Variation in radius and position of the yield surface in deviatoric stress space

and the normal to the yield function as:

$$\mu_{ij} = \frac{3}{2}\left(\frac{\dot{S}_{ij}}{\dot{q}}\right) \qquad 2.2.11$$

Both μ_{ij} and η_{ij} are normalised tensors with absolute values of $\sqrt{3/2}$.[15]

Since μ_{ij} and $\frac{\partial f}{\partial S_{ij}}$ have the same direction, Equation 2.1.11 allows us to state:

$$\dot{e}^{pl}_{ij} = \dot{\bar{e}}^{pl}\mu_{ij} \qquad 2.2.12$$

Integrating the above equation will determine the proportions of elastic and plastic strains developed. We need to define the yield function fully before we can set about integrating Equation 2.2.12. We define the yield stress (as in Equation 2.2.6) as H. Thus we can restate Equation 2.2.1 (for the rate independent case):

$$f = q - H \qquad 2.2.13$$

This is the general form of the Von Mises yield surface for rate independent plasticity. To check if yield will occur, we need to see if the stress which arises from assuming an entirely elastic strain increment lies outside the yield surface. This stress is known as the equivalent elastic predictor stress, q^E. Yielding occurs if:

$$q^E > f_{|a} \qquad 2.2.14$$
$$S_{ij}^E = S_{ij|a} + 2G\Delta e_{ij}$$

2.3 Rate Dependence

There are several rate dependent formulae governing the maximum allowable stress due to high strain rates.[11,19,] These generally fall into a class of rate dependence models called power law hardening. One common such type is the Couper Symonds relation, which will be used in subsequent sections. The relation states:[11]

$$\dot{\bar{e}}^{pl} = D\left(\frac{q}{H} - 1\right)^p \qquad 2.3.1$$

Where D and p are predetermined material properties. By assuming that the strain rate remains constant over the increment, we can write:

$$\Delta\bar{e}^{pl} = \Delta t D\left(\frac{q}{H} - 1\right)^p \qquad 2.3.2$$

Inverting leads to:

$$q = D_2 H \qquad 2.3.3$$

where

$$D_2 = \left[\frac{\left(\Delta\bar{e}^{pl}\right)^{\frac{1}{p}} + \left(D\Delta t\right)^{\frac{1}{p}}}{\left(D\Delta t\right)^{\frac{1}{p}}}\right] \qquad 2.3.4$$

or

$$D_2 = \left[\frac{\left(\frac{\Delta\bar{e}^{pl}}{\Delta t}\right)^{\frac{1}{p}} + (D)^{\frac{1}{p}}}{(D)^{\frac{1}{p}}}\right] \qquad 2.3.5$$

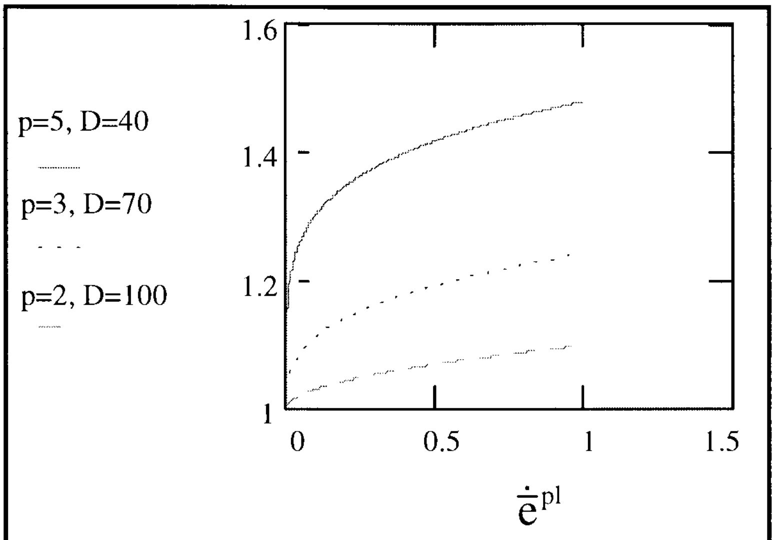

Figure 2.3.1 Overstress factor D_2 with differing governing parameters.

Thus the maximum allowable stress is scaled up by a factor of D_2 depending on the equivalent plastic strain increment and the time increment, thus being called an overstress factor. The law effectively results in a temporary increase in the yield surface radius for the Von Mises model. If the strain rate drops, so does the radius of the yield surface.

The yield function of Equation 2.2.13 may be modified to include rate dependent behaviour by setting:

$$\overline{S} = D_2 H \qquad 2.3.6$$

2.4 Non-Proportional Loading

The following equation defines proportional loading:

$$\Delta e_{ij}^{|(n-1)} = c\Delta e_{ij}^{|(n)} \qquad 2.4.1$$

where c is some scalar factor. The $(n)^{th}$ increment is a direct proportion of the $(n-1)^{th}$ increment. If this is not the case, we have non-proportional loading. This is represented graphically by Figure 2.4.1, in which $(\sim)_{|a}$ denotes a starting vlue for the increment.

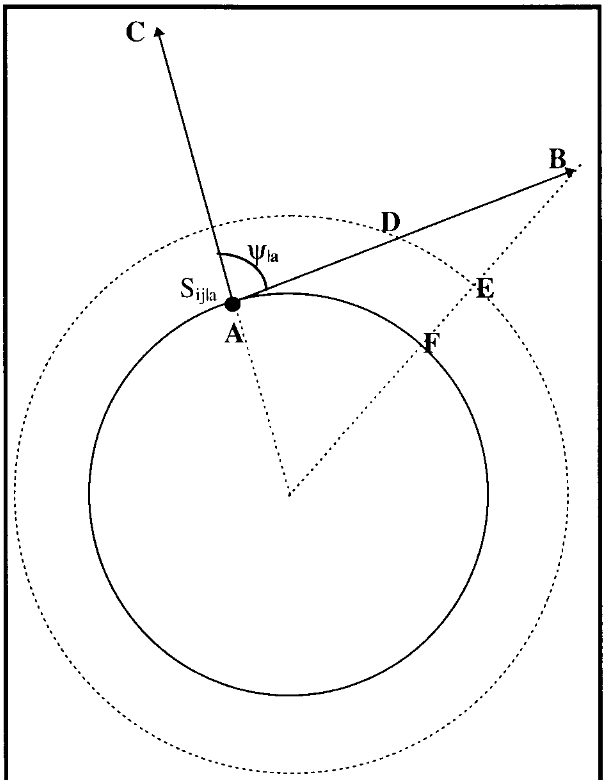

Figure 2.4.1 Proportional loading (**AC**) and non-proportional loading (**AB**)

We can see that non-proportional loading introduces complications. If $(AD) = (DB)$ then $(FE) \neq (EB)$. This means that during a strain increment, the yield surface radius does not increase in direct proportion to the portion of the strain increment completed, thus adding a further non-linearity to the solution algorithm.

2.5 Partially Elastic Increments (Exiting the Yield Surface)

If $S_{ij|a}$ lies within the yield surface, and S^E_{ij} lies outside the yield surface, then the strain increment must be split up into a fully elastic strain increment which will take the stress onto the yield surface, and an elasto-plastic increment for the remainder of the strain increment.

Let us define the portion of the total strain increment which is purely elastic as:

$$\Delta e^{|*}_{ij} = \alpha \Delta e_{ij} \qquad 2.5.1$$

Then:

$$\Delta e^{|**}_{ij} = (1-\alpha)\Delta e_{ij} \qquad 2.5.2$$

Where $\Delta e^{|**}_{ij}$ is the elasto-plastic proportion of the stain increment.

We can make the following definitions:

$$r_1 = H_{|a} - q_{|a}$$
$$r_2 = q^E - q_{|a} \qquad 2.5.3$$

$$\alpha = \frac{r_1}{r_2} \qquad 2.5.4$$

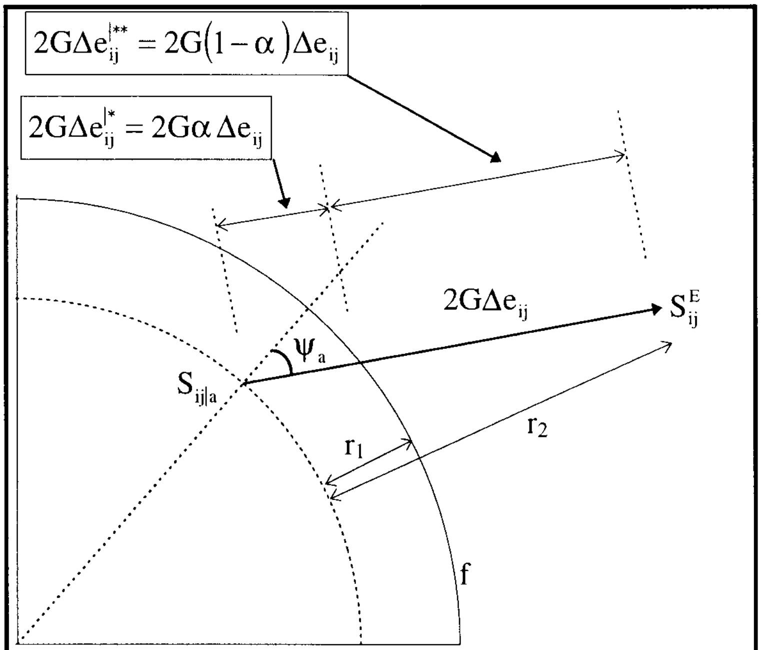

Figure 2.5.1 A partially elasto-plastic non-proportional strain increment

We then start an iterative process, by updating the starting value of the stress $S_{ij|a}$, strain increment Δe_{ij} and time increment Δt with:

$$\begin{bmatrix} S_{ij|a} \\ \Delta e_{ij|a} \\ \Delta t \end{bmatrix}^{|(n+1)} = \begin{bmatrix} S_{ij|a} \\ \Delta e_{ij|a} \\ \Delta t \end{bmatrix}^{|(n)} + \alpha^{|(n)} \begin{bmatrix} 2G\Delta e_{ij} \\ -\Delta e_{ij|a} \\ -\Delta t \end{bmatrix}^{|(n)} \qquad 2.5.5$$

We check a convergence tolerance T such that:

$$|T| \le 0.01(H_a)$$
$$T = q_a - H_a \qquad 2.5.6$$

Once the initial values are sufficiently close to the yield surface to satisfy the convergence criteria, we have the complete set of variables necessary to start the elasto-plastic increment.

If convergence is not achieved, we are required to repeat the process from Equation 2.5.3.

3. LITERATURE REVIEW

3.1 Solution Schemes

By making use of Equation 2.1.1, Equation 2.1.7 we can write the following equation for the stress at the end of an increment:

$$S_{ij|b} = S_{ij|a} + 2G(\Delta e_{ij} - \Delta e_{ij}^{pl}) \qquad 3.1.1$$

$$\Delta e_{ij}^{pl} = \int_{t_{|a}}^{t_{|b}} \dot{\bar{e}}^{pl} \mu_{ij}\, dt \qquad 3.1.2$$

$$t_{|a} \le t \le t_{|b} \qquad 3.1.3$$

where $(\sim)_{|a}$ and $(\sim)_{|b}$ denote initial and final values for the increment, respectively.

We can see that the final stress can be fully defined once the total plastic strain increment is known. Since the solution is dependent on increments of strain, it is essentially strain driven and the total strain rate is assumed to remain constant over the entire increment. In order to calculate the total plastic strain increment, we need to integrate Equation 3.1.2. The methods of accomplishing this and the accuracy of these methods has been the subject of numerous studies over the past years.

We note that the integration procedure of Equation 3.1.2 will require knowledge of the yield surface gradient (μ_{ij}) along the stress path, which depends on the type of hardening and rate dependence laws adopted. Since either the isotropic hardening description, or the rate dependence model, or both, may be highly non-linear, the behaviour of the stress path can only be determined accurately by subincrementation of the increments of total strain and time. We wish to avoid this since it is computationally more expensive, and thus have to make assumptions regarding the plastic strain direction μ_{ij}.

Even if the plastic strain direction were assumed constant (as it will be) Equation 3.1.2 still contains 2 unknowns. We therefore need to establish another equation using at least one of the unknowns of equivalent plastic strain rate or plastic strain rate. In order to do this we make use of an equation which ensures that the stress at any point during the increment does not violate the yield surface. This is known as enforcing consistency of the plastic constitutive equations, and the point at which this is enforced is termed the collation point. Since the resulting equation is potentially highly non-linear, a local Newton-Raphson, secant or similar iterative algorithm is necessary to obtain a solution.

The trend in development of these equations has been to move away from setting up the consistency condition using tensors, to doing so by means of their scalar equivalents. The resulting iterative scheme is computationally quicker when using scalar values rather than tensors. Unfortunately, once the final equivalent values of stress and strain have been fully solved for it is necessary to determine their direction and thus allow determination of the final full tensors. For example, we can establish a consistency condition and solve it to give us the final equivalent stress $q_{|b}$, but:

$$S_{ij|b} = q_{|b}\mu_{ij|b} \tag{3.1.4}$$

Thus leaving us to find $\mu_{ij|b}$.

We will first see where the original full tensor derivations lead to in order to understand the scalar to tensor update directions.

The most popular integration algorithms for the Von Mises yield condition, have been the Backward Euler (backward difference), Trapezoidal and Generalised Midpoint algorithms, all of which fall in the general class of predictor return algorithms. These will be discussed briefly, and a geometrical interpretation of the full tensor derivations presented.

3.1.1 The Backward Euler Algorithm

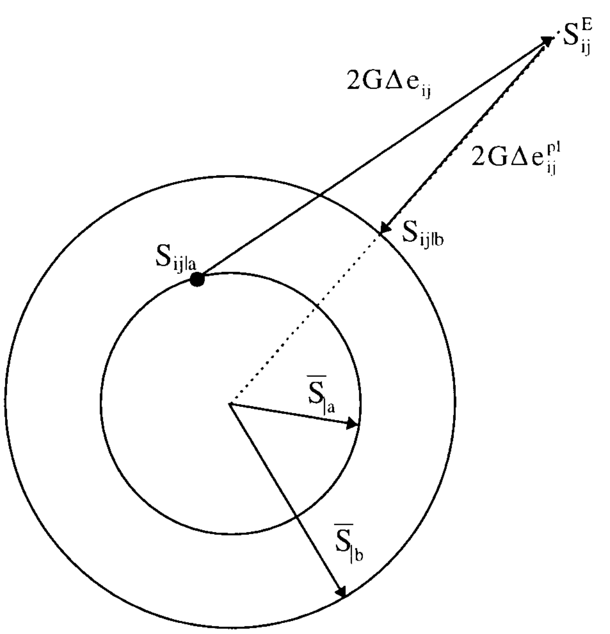

Figure 3.1.1 Isotropic hardening with the backward difference assumption[1]

The Backward Euler method sets the collation point for the consistency condition at the end of increment. This method is extremely popular since it is simple and, like other methods, guarantees convergence and symmetry of the resulting consistent tangent modulus.[15,4] It is however only first order accurate.[13]

We see from Figure 3.1.1 that the final stress values ($S_{ij|b}$) are achieved by returning onto the yield surface from an elastic predictor stress (S^E_{ij}), which is the stress which would have arisen had the entire strain increment been elastic. The distance and direction of return from the elastic predictor is given by the stress effectively 'lost' to plasticity. Thus the direction of plastic strain is assumed to be constant in the direction of the elastic predictor stress, giving the following final solution to the equivalent plastic strain increment:

$$\Delta e^{pl}_{ij} = \Delta t\, \dot{\bar{e}}^{pl}_{|b} \mu_{ij|b} = \Delta \bar{e}^{pl} \mu_{ij|b} \qquad 3.1.5$$

The resulting consistent tangent modulus is also symmetric

3.1.2 The Generalised Midpoint Algorithm

Figure 3.1.2 demonstrates how the generalised midpoint method sets the collation point ($S_{ij|\theta}$) at some intermediate point along the total strain path, with θ representing the fraction of the path to use. The predictor ($S^E_{ij|\theta}$) is the fraction θ of the full elastic predictor, and the stress is returned from there onto the intermediate yield surface. The final stress ($S_{ij|b}$) is then obtained by linear interpolation of the initial and intermediate values of stress.

$$S_{ij|\theta} = (1-\theta) S_{ij|a} + \theta S_{ij|b} \qquad 3.1.6$$

$$0 \le \theta \le 1 \qquad 3.1.7$$

Where $(\sim)_{|\theta}$ denotes an intermediate value.

As can be seen from Figure 3.1.2, since the consistency condition is not enforced at the final stress state ($S_{ij|b}$), there is no guarantee that the final stress will lie on the yield surface, unless we have proportional loading with linear hardening rules. Since proportional loading occurs when the strain increment direction remains constant from increment to increment (only achievable in the simplest analyses), and most materials' have a certain degree of non-linearity during hardening, the midpoint rule is prone to both under and over prediction during large increments.

The plastic flow direction is assumed to be constant in the direction of the intermediate elastic predictor stress:

$$\Delta e^{pl}_{ij} = \Delta t\, \dot{\bar{e}}^{pl}_{|\theta} \mu_{ij|\theta} = \Delta \bar{e}^{pl} \mu_{ij|\theta} \qquad 3.1.8$$

This method gives unconditional stability when the following limit on θ is applied: [1,13]

$$0.5 \leq \theta \leq 1 \quad 3.1.9$$

In order to achieve second order accuracy for <u>small</u> increments: [13]

$$\theta = 0.5 \quad 3.1.10$$

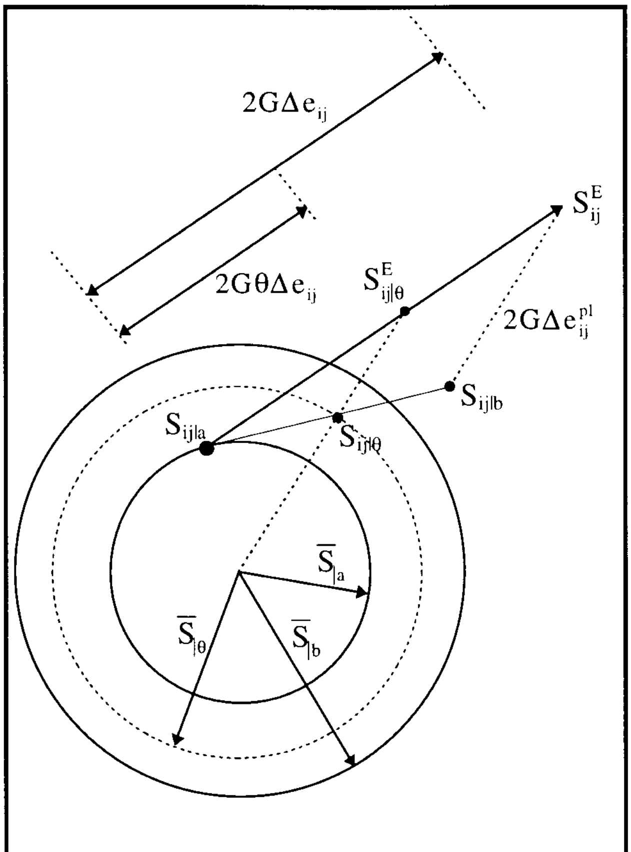

Figure 3.1.2 Isotropic hardening with the midpoint rule[1]

Larger choices of θ may be necessary in order to improve accuracy during larger increments. [13] Setting θ to the limit of 1 will satisfy the consistency condition at the end of the increment, but that in itself will not mean that the radius of yield surface predicted is correct. In fact, a floating θ may be the optimum means of weighting the consistency condition. The further away the point of consistency is from a known solution, the larger the error will be, but the further the point of consistency is from the end of the increment, the greater the susceptibility of the algorithm to non-linear hardening and non-proportional hardening.

A further advantage of the midpoint rule of Figure 3.1.2 is symmetry of the tangent modulus.[4]

We can see from Figure 3.1.2 that setting θ = 1 will result in the backward Euler method.

3.1.3 The Trapezoidal Rule

In the trapezoidal algorithm, the plastic strain increment is subdivided into two parts according to Equation 3.1.11.

$$\Delta e_{ij}^{pl} = \Delta e_{ij}^{pl|*} + \Delta e_{ij}^{pl|**} \qquad 3.1.11$$

$$\Delta e_{ij}^{pl|*} = \frac{1-\theta}{\theta} \Delta e_{ij|a}^{pl|**} \qquad 3.1.12$$

Where $(\Delta\sim)_{|a}$ denotes an incremental value brought over from the previous increment.

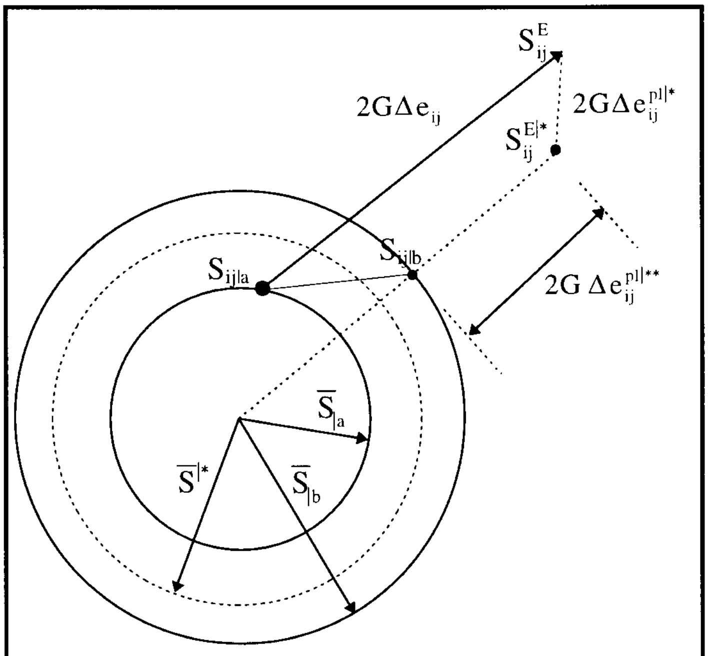

Figure 3.1.3 Isotropic hardening with the trapezoidal rule[1]

The first part of the plastic strain increment ($\Delta e_{ij}^{pl|*}$) is fully determined by forward linear projection of the second part of the plastic strain ($\Delta e_{ij}^{pl|**}$) from the previous increment.[4] The direction of return is from the modified elastic predictor ($S_{ij}^{E|*}$) to the final stress ($S_{ij|b}$).

The point of collation is once again at the end of the increment, thus ensuring consistency and not allowing severe over or under prediction. The forward projection of the second part of the strain increment from the previous increment can lead to errors. For example: The previous increment may have had a large plastic increment, and during unloading in the current increment, the implicit forward projection of the plastic strain could lead to large angular errors.[2] From the study completed by Rencontré and Martin[2] we find 'a restriction on the step size and the feasibility of using a mixed formulation for unloading still needs to be established.' Ortiz and Popov[13] analysed the accuracy and stability of the midpoint and trapezoidal schemes, and concluded that the former rated better. This leads us to search elsewhere for a viable integration scheme.

3.1.4 Separate Return Algorithms

From the previous discussion, we can see that both the enforcement of the consistency condition as well as the choice of return direction is important. All three previous algorithms were derived using full tensor mathematics, and any non-linear equation solution would have to take place on these tensor values. By restating the constitutive equations in a scalar form, one can fully enforce the consistency condition at the desired point during the increment. By using scalar quantities, the choice of return direction can remain open until after this has been completed.

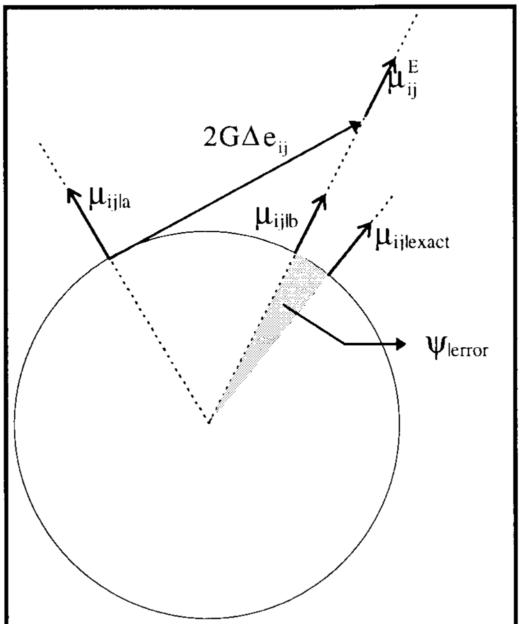

Figure 3.1.4 Directions of plastic flow and stress return for a Von Mises perfectly plastic yield condition

Krieg and Krieg[5] derived an exact solution for the return direction for a Von Mises perfectly plastic, rate independent model, and compared the errors in the return direction which resulted from the standard Backward Euler Radial Return integration scheme. Naturally, without any hardening of any nature, satisfying the consistency condition is simple, and yet the process of returning to the yield surface (of a known correct radius) from the fully elastic predictor (direction $\mu_{ij|b}$) could induce errors of up to $\psi_{|error}=12.7°$.[5]

This type of analysis was carried further by Schreyer, et. al.[14] to compare the angular errors resulting from a linear isotropic hardening Von Mises yield condition. The results showed similar angular errors.

3.1.5 General Solution Requirements

We have thus established the basic requirements for a material update using a Von Mises model:

1. Remove volumetric components from the initial stress and strain increment tensors.
2. Remove back stress components.
3. Calculate predictor stress, and check for yielding.
4. If yielding occurs, calculate the proportion of plastic strain in the increment.
5. Update the stress, and replace the updated volumetric and back stress components.
6. Calculate a consistent tangent modulus.

3.1.6 An Optimal Algorithm

From the analysis in the preceding Sections, we can determine a few basic necessities in order to construct the optimal algorithm.

1. The consistency condition must be satisfied at an optimum point by a scalar equation which is dependent on the underlying hardening rules governing the evolution of stress with respect to strain. This optimum point must be chosen so as to keep the collation point as near to that of second order accuracy as possible, but close enough to the end of the increment to ensure final incremental values are not severely over or under predicted.
2. The return direction (stress update direction) must be evaluated by analysing both the strain increment magnitude relative to the yield surface radius, as well as the direction.
3. The consistent tangent modulus must be available, accurate and symmetric.

We will examine the full formulation of the widely used backward Euler method for a Von Mises, arbitrarily isotropic, rate dependent model in the following section in order to have the groundwork for a full comparison with the suggested algorithm presented later in this work.

3.2 The Standard Backward Euler Radial Return

The integration of the rate equations by means of a backward Euler scheme enjoys popularity as a simple, reasonably accurate scheme. Several FEM packages, including ABAQUS, uses this scheme to solve for metal material models which display isotropic hardening or softening and possible rate dependent behaviour.

3.2.1 Integration Algorithm

We will see the formulation for a stress-strain update procedure, which will include rate dependent behaviour with arbitrarily isotropic hardening. The yield function will be of the Von Mises type with assumed associative flow, thus making it ideal for analysis of common metals. The consistency condition will be enforced using a local Newton-Raphson scheme and the stress update direction assumed to be that of the conventional backward Euler integration procedure[11] (Section 3.1.1).

At the beginning of the increment it is necessary to remove the volumetric strain components from the strain increment, in order to work with the Von Mises criterion in deviatoric space. We do this by using Equation 2.1.8 to Equation 2.1.7.

In order to determine if any plasiticity will occur during the increment, the magnitude of the elastic predictor stress must be known.

$$S_{ij}^{E} = S_{ij|a} + 2G\Delta e_{ij} \tag{3.2.1}$$

Plasticity will occur if:

$$q^{E} > H\left(\bar{e}^{pl}_{|a}\right) \tag{3.2.2}$$

If Equation 3.2.2 is not true, then plasticity will not occur, i.e. the stress will not violate the yield surface, and the final stresses and plastic strain increment is updated as:

$$\begin{aligned} S_{ij|b} &= S_{ij|a} + 2G\Delta e_{ij} \\ \Delta e_{ij}^{pl} &= 0\delta_{ij} \\ D_{ijkl} &= D_{ijkl}^{el} \end{aligned} \tag{3.2.3}$$

If plasticity occurs (i.e. Equation 3.2.2 is true), we use the assumption that the plastic strain direction is the same direction as the elastic predictor stress point and Equation 3.1.5 to integrate for the total plastic strain increment.

$$\Delta e_{ij}^{pl} = \Delta\bar{e}^{pl}\mu_{ij|b} \tag{3.2.4}$$

We use Equation 2.2.12 and Equation 3.1.1 to write the final stress as:

$$S_{ij|b} = 2G(e_{ij|a}^{el} + \Delta e_{ij} - \Delta\bar{e}^{pl}\mu_{ij|b}) \tag{3.2.5}$$

Substituting with Equation 2.2.11 for the stress update direction and rearranging:

$$\left(1 + \frac{3G}{q_{|b}}\Delta\bar{e}^{pl}\right)S_{ij|b} = 2G\left(e_{ij|a}^{el} + \Delta e_{ij}\right) \tag{3.2.6}$$

We can assign:

$$\hat{e}_{ij} = e_{ij|a}^{el} + \Delta e_{ij} \tag{3.2.7}$$

Substituting using Equation 3.2.7 and contracting to bring the resulting equation into a scalar form:

$$\left(1+\frac{3G}{q_{|b}}\Delta\bar{e}^{pl}\right)S_{ij|b} = 2G\hat{e}_{ij} \qquad 3.2.8$$

$$\left(1+\frac{3G}{q_{|b}}\Delta\bar{e}^{pl}\right)^2 S_{ij|b}S_{ij|b} = 4G^2\hat{e}_{ij}\hat{e}_{ij} \qquad 3.2.9$$

Substituting for the equivalent stresses and strains given by Equation 2.2.2 and Equation 2.2.3:

$$q_{|b} = G(2\tilde{e} - 3\Delta\bar{e}^{pl})$$
$$\tilde{e} = \sqrt{\frac{3}{2}\hat{e}_{ij}\hat{e}_{ij}} \qquad 3.2.10$$

Equation 3.2.10 gives the equivalent stress at the end of the increment. We now choose to make the end of the increment the point to enforce consistency of the yield surface. Thus, following Equation 2.2.1, the scalar yield function is:

$$f = G(2\tilde{e} - 3\Delta\bar{e}^{pl}) - \bar{S} \qquad 3.2.11$$

Adopting the Couper Symonds rate dependence law according to Section 2.3:

$$\bar{S} = D_2 H \qquad 3.2.12$$

$$f = G(2\tilde{e} - 3\Delta\bar{e}^{pl}) - D_2 H \qquad 3.2.13$$

$$H \rightarrow H(\bar{e}^{pl}_{|a} + \Delta\bar{e}^{pl}) \qquad 3.2.14$$

H must be fully defined by the user, and for general purposes this means that the user has to input a table of values of yield versus equivalent plastic strain prior to analysis. From this table, the gradient of the yield curve H' can be calculated.

$$H' \rightarrow H'(\bar{e}^{pl}_{|a} + \Delta\bar{e}^{pl}) \qquad 3.2.15$$

Thus we now have a potentially highly non-linear function (Equation 3.2.13) of equivalent plastic strain increment which we need to zero in order to enforce the yield surface conditions of Equation 2.1.12. This is done by means of a Newton-Raphson iterative scheme.

$$\Delta\bar{e}^{pl|(n+1)} = \Delta\bar{e}^{pl|(n)} - \frac{f(\Delta\bar{e}^{pl|(n)})}{f'(\Delta\bar{e}^{pl|(n)})} \qquad 3.2.16$$

We can see that the scheme requires a starting value ($\Delta e^{pl|(n)}$) for the iteration. Since the plastic strain developed during an increment can not be greater than the total strain increment, it follows a reasonable choice for the starting value is:

$$\Delta\bar{e}^{pl|(1)} = \Delta\bar{e} \qquad 3.2.17$$

We are thus starting the iteration by assuming perfect plasticity. The starting value can therefore be called a plastic predictor to the local Newton-Raphson scheme.

We need to define the gradient of the yield function as a function of equivalent plastic strain.

$$f' = \frac{df}{d\Delta\bar{e}^{pl}} = -3G - D_2H' - H\frac{(\Delta\bar{e}^{pl})^{\frac{1-p}{p}}}{p(D\Delta t)^{\frac{1}{p}}} \qquad 3.2.18$$

The iterative procedure is continued until some convergence criteria is achieved. We can assume the solution is converged once the yield function is a small percentage of the allowable flow stress, i.e. when, for example:

$$|f| \leq (0.01)\bar{S} \qquad 3.2.19$$

Thus, based only on initial values of the increment, and with a suitable iteration scheme, we can calculate the scalar equivalent of the equivalent plastic strain increment, allowing us to update the equivalent final stress.

$$f = 0 \quad \Rightarrow \quad q_{|b} = \bar{S} \qquad 3.2.20$$

3.2.2 Stress Update Procedure

In order to obtain the final updated stress, we substitute for the equivalent plastic strain increment in Equation 3.2.8 and Equation 3.2.4:

$$S_{ij|b} = \frac{2G}{\left(1 + \frac{3G}{q_{|b}}\Delta\bar{e}^{pl}\right)}\hat{e}_{ij} \qquad 3.2.21$$

$$\begin{aligned} \Delta e_{ij}^{pl} &= \Delta\bar{e}^{pl}\mu_{ij|b} \\ &= \Delta\bar{e}^{pl}\frac{3}{2}\frac{S_{ij|b}}{q_{|b}} \end{aligned} \qquad 3.2.22$$

The vectors and return update direction are represented graphically in Figure 3.2.1. The choice of the return direction is enforced by asserting that the plastic strain is in the direction of the elastic predictor stress.

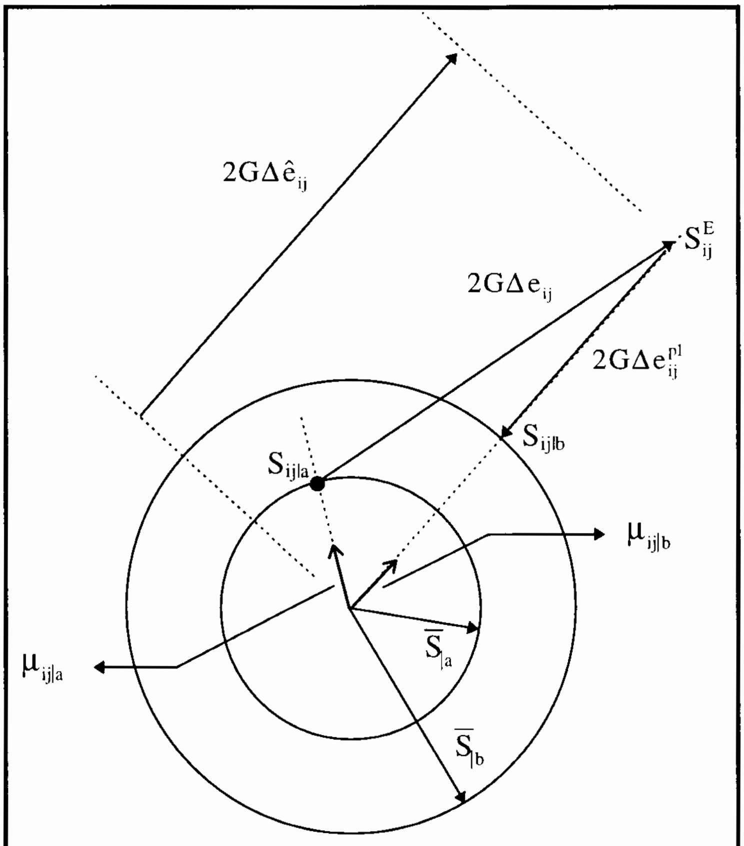

Figure 3.2.1 The backward Euler radial return stress update

With the deviatoric variables fully determined, the linear volumetric components can be reintroduced into the tensors.

3.2.3 The Consistent tangent modulus

The consistent tangent modulus must be defined at the end of the increment:

$$D_{ijkl} = \frac{\partial \sigma_{ij}}{\partial \varepsilon_{kl}} = \frac{\partial \sigma_{ij}}{\partial \Delta \varepsilon_{kl}} \qquad 3.2.23$$

We can define a deviatoric consistent tangent modulus:

$$d_{ijkl} = \frac{\partial S_{ij}}{\partial e_{ij}} = \frac{\partial S_{ij}}{\partial \Delta e_{kl}} = \frac{\partial S_{ij}}{\partial \hat{e}_{kl}} \qquad 3.2.24$$

Which can be converted to the required total stress space tensor with:[15]

$$D_{ijkl} = K\delta_{ij}\delta_{kl} + d_{ijkl}\left(\delta_{ijkl} - \frac{1}{3}\delta_{ij}\delta_{kl}\right) \quad 3.2.25$$

where δ_{ij} is the second order unit tensor (Kronecker delta) and δ_{ijkl} is the fourth order unit tensor:

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \quad 3.2.26$$

$$\delta_{ijkl} = \delta_{ik}\delta_{jl} \quad 3.2.27$$

We will use the final statement of Equation 3.2.24 to define the deviatoric consistent tangent modulus. We take the variation of Equation 3.2.8, Equation 3.2.10 and Equation 3.2.20 with respect to all values at the end of the increment:

$$\left(1 + \frac{3G}{q_{|b}}\Delta\bar{e}^{pl}\right)\partial S_{ij} + S_{ij|b}\frac{3G}{q_{|b}}\left(\partial\bar{e}^{pl} - \frac{\Delta\bar{e}^{pl}}{q_{|b}}\partial q\right) = 2G\partial\hat{e}_{ij} \quad 3.2.28$$

$$\partial q = D_2 H' \partial\bar{e}^{pl} + \frac{H\left(\Delta\bar{e}^{pl}\right)^{\frac{1-p}{p}}}{p\left(D\Delta t\right)^{\frac{1}{p}}}\partial\Delta\bar{e}^{pl} \quad 3.2.29$$

$$\partial q + 3G\partial\bar{e}^{pl} = 2G\partial\tilde{e} \quad 3.2.30$$

Combining Equation 3.2.29 and Equation 3.2.30 and noting that $\partial\Delta\bar{e}^{pl} \equiv \partial\bar{e}^{pl}$:

$$\partial\bar{e}^{pl}\left[D_2 H' + \frac{H\left(\Delta\bar{e}^{pl}\right)^{\frac{1-p}{p}}}{p\left(D\Delta t\right)^{\frac{1}{p}}} + 3G\right] = 2G\partial\tilde{e} \quad 3.2.31$$

We can define:

$$H_2' = D_2 H'$$

$$D_3 = \frac{H\left(\Delta\bar{e}^{pl}\right)^{\frac{1-p}{p}}}{p\left(D\Delta t\right)^{\frac{1}{p}}} \quad 3.2.32$$

$$B = \frac{H_2' + D_3}{3G}$$

From Equation 3.2.10:

$$\partial\tilde{e} = \frac{3}{2\tilde{e}}\hat{e}_{ij}\partial\hat{e}_{ij} \qquad 3.2.33$$

Substituting Equation 3.2.33 into Equation 3.2.31:

$$\partial\bar{e}^{pl} = \frac{1}{\tilde{e}(1+B)}\hat{e}_{ij}\partial\hat{e}_{ij} \qquad 3.2.34$$

From Equation 3.2.29 and Equation 3.2.34:

$$\partial q = \frac{(H_2' + D_3)}{\tilde{e}(1+B)}\hat{e}_{ij}\partial\hat{e}_{ij} \qquad 3.2.35$$

Making the necessary substitutions into Equation 3.2.28:

$$\partial S_{ij} = \left(\frac{q_{|b}}{q + 3G\Delta\bar{e}^{pl}}\right)\left(\begin{array}{l}\dfrac{\Delta\bar{e}^{pl}}{q_{|b}}\left(\dfrac{H_2' + D_3}{\tilde{e}(1+B)}\right)\hat{e}_{kl}\partial\hat{e}_{kl} \\ -\dfrac{1}{\tilde{e}(1+B)}\hat{e}_{kl}\partial\hat{e}_{kl}\end{array}\right)S_{ij|b}\frac{3G}{q_{|b}} + \frac{2G\partial\hat{e}_{ij}}{\left(1 + \dfrac{3G}{q_{|b}}\Delta\bar{e}^{pl}\right)} \qquad 3.2.36$$

We can now substitute for $S_{ij|b}$ and $\tilde{e}$ from the first part of Equation 3.2.10 and Equation 3.2.21 and make the following substitutions:

$$R = \frac{3}{2q_{|b}\tilde{e}}\frac{\left(1 - \Delta\bar{e}^{pl}\dfrac{H_2' + D_3}{q_{|b}}\right)}{(1+B)} \qquad 3.2.37$$

$$Q = \frac{q_{|b}}{\tilde{e}}$$

and convert the indices to the necessary forms with:

$$\partial\hat{e}_{ij} = \delta_{ijkl}\partial\hat{e}_{kl} \qquad 3.2.38$$

to give us the final deviatoric consistent tangent modulus:

$$\begin{aligned}\partial S_{ij} &= \left(Q\delta_{ijkl} - RS_{ij|b}S_{kl|b}\right)\partial\hat{e}_{kl} \\ \Rightarrow d_{ijkl} &= \left(Q\delta_{ijkl} - RS_{ij|b}S_{kl|b}\right)\end{aligned} \qquad 3.2.39$$

3.2.4 Summary of Backward Euler with Radial Return

1. Calculate the deviatoric equivalents of all variables.
2. Assume zero plastic strain and calculate an equivalent elastic predictor stress.
3. If the predictor stress lies within the yield surface, update all stresses elastically. Goto 6.
4. Use local a Newton-Raphson scheme to zero yield function and obtain equivalent plastic strain increment.
5. Calculate the final equivalent stress.
6. Return the final equivalent stress and equivalent plastic strain to tensor values by multiplying them by the direction from the centre of the yield surface to the elastic predictor stress.
7. Calculate the consistent tangent modulus.

4. OPTIMISED MIDPOINT METHOD WITH VARIABLE RETURN

After the work presented by P. A. Fotiu[15] we will investigate the implementation of an optimised midpoint method with a variable stress return direction. The material model under study is based on a Von Mises yield criterion, the overstress power law of Couper Symonds with an isotropic/kinematic hardening material.

4.1 Integration procedure

We start by extracting the deviatoric and back stress components from the initial stress:

$$S_{ij|a} = \sigma_{ij|a} - \frac{1}{3}\sigma_{kk|a}\delta_{ij} - B_{ij|a} \qquad 4.1.1$$

where B_{ij} is the backstress tensor and is assumed to develop as a function of the equivalent plastic strain:

$$\dot{B}_{ij} = H'\left(\bar{e}^{pl}_{|a} + \Delta\bar{e}^{pl}\right)\dot{e}^{pl}_{ij} \qquad 4.1.2$$

During the increment, there are 3 points in time which are of interest to us for the generalised midpoint rule. These are $t_{|a} \le t_{|\theta} \le t_{|b}$. Some important relations to recall:

$$\mu_{ij} = \frac{3}{2}\frac{S_{ij}}{q} \qquad 4.1.3$$

$$\mu_{ij}\mu_{ij} = \frac{3}{2} \qquad 4.1.4$$

$$\dot{e}^{pl}_{ij} = \dot{\bar{e}}^{pl}_{ij}\mu_{ij} \qquad 4.1.5$$

$$\eta_{ij} = \frac{\dot{e}_{ij}}{\dot{\bar{e}}} \qquad 4.1.6$$

$$\eta_{ij}\eta_{ij} = \frac{3}{2} \qquad 4.1.7$$

$$\dot{S}_{ij} = 2G\left(\dot{e}_{ij} - \dot{e}^{pl}_{ij}\right) \qquad 4.1.8$$

$$\Delta\bar{e}^{pl} = \sqrt{\frac{2}{3}\Delta e^{pl}_{ij}\Delta e^{pl}_{ij}} \qquad 4.1.9$$

In order to integrate Equation 4.1.5, we use the mean value theorem from Equation 3.1.8:

$$\Delta e^{pl}_{ij} = \dot{\bar{e}}_{|\theta} \Delta t \mu_{ij|\theta} = \frac{3}{2} \Delta \bar{e}^{pl} \frac{S_{ij|\theta}}{\bar{S}_{|\theta}} \qquad 4.1.10$$

where we use the substitution $q_{|\theta} = \bar{S}_{|\theta}$. This must be true during plastic loading, since the equivalent stress must not exceed the maximum allowable stress at the point of collation.

The values of $S_{ij|\theta}$ and $\bar{e}^{pl}_{|\theta}$ are approximated by linear interpolation:

$$S_{ij|\theta} = (1-\theta) S_{ij|a} + \theta S_{ij|b} = S_{ij|a} + \theta \Delta S_{ij} \qquad 4.1.11$$

$$\bar{e}^{pl}_{|\theta} = (1-\theta) \bar{e}^{pl}_{|a} + \theta \bar{e}^{pl}_{|b} = \bar{e}^{pl}_{|a} + \theta \Delta \bar{e}^{pl} \qquad 4.1.12$$

We integrate Equation 4.1.2 to find the backstress at the collation point:

$$B_{ij|\theta} = B_{ij|a} + \mu_{ij|\theta} \int_{\bar{e}^{pl}_{|a}}^{\bar{e}^{pl}_{|\theta}} H' d\bar{e}^{pl} = B_{ij|a} + \left(H_{|\theta} - H_{|a} \right) \mu_{ij|\theta} \qquad 4.1.13$$

$$B_{ij|\theta} = B_{ij|a} + \frac{2}{3} H^{*}_{|\theta} \Delta e^{pl}_{ij} \qquad 4.1.14$$

$$H^{*}_{|\theta} = \frac{3}{2} \frac{H_{|\theta} - H_{|a}}{\Delta \bar{e}^{pl}} \qquad 4.1.15$$

Substituting Equation 4.1.8, Equation 4.1.11 and Equation 4.1.14 into Equation 4.1.10:

$$\Delta e^{pl}_{ij} = \frac{3}{2} \frac{\Delta \bar{e}^{pl}}{\bar{S}_{|\theta}} \left(S_{ij|a} + \theta \Delta S_{ij} - B_{ij|a} - \frac{2}{3} H^{*}_{|\theta} \Delta e^{pl}_{ij} \right) \qquad 4.1.16$$

Using Equation 4.1.3 and Equation 4.1.6, and rearranging:

$$\Delta e^{pl}_{ij} \left[\bar{S}_{|\theta} + \Delta \bar{e}^{pl} \left(3G\theta + H^{*}_{|\theta} \right) \right] = q_{|a} \Delta \bar{e}^{pl} \left[\mu_{ij|a} + \frac{3G\theta \, \Delta \bar{e}}{q_{|a}} \eta_{ij} \right] \qquad 4.1.17$$

We can define a relative measure of the increment size:

$$\Gamma = \frac{3G\Delta \bar{e}}{q_{|a}} \qquad 4.1.18$$

giving:

$$\Delta e_{ij}^{pl} = \frac{q_{|a}\Delta\bar{e}^{pl}}{\left[\bar{S}_{|\theta} + \Delta\bar{e}^{pl}\left(3G\theta + H_{|\theta}^{*}\right)\right]}\left[\mu_{ij|a} + \theta\Gamma\eta_{ij}\right] \qquad 4.1.19$$

Substituting this into Equation 4.1.9 in order to obtain a scalar equation to satisfy consistency:

$$\Delta e_{ij}^{pl} = \frac{q_{|a}\Delta\bar{e}^{pl}}{\left[\bar{S}_{|\theta} + \Delta\bar{e}^{pl}\left(3G\theta + H_{|\theta}^{*}\right)\right]}\sqrt{\frac{2}{3}\left(\mu_{ij|a} + \theta\Gamma\eta_{ij}\right)\left(\mu_{ij|a} + \theta\Gamma\eta_{ij}\right)} \qquad 4.1.20$$

By using Equation 4.1.4 and Equation 4.1.7, we get:

$$\bar{S}_{|\theta} + \Delta\bar{e}^{pl}\left(3G\theta + H_{|\theta}^{*}\right) = q_{|a}y(\theta) \qquad 4.1.21$$

where

$$\begin{aligned} y(\theta) &= \sqrt{\frac{2}{3}\left(\mu_{ij|a} + \theta\Gamma\eta_{ij}\right)\left(\mu_{ij|a} + \theta\Gamma\eta_{ij}\right)} \\ &= \sqrt{(\theta\Gamma)^2 + 2\theta\Gamma\cos\psi_{|a} + 1} \end{aligned} \qquad 4.1.22$$

and

$$\cos\psi_{|a} = \frac{2}{3}\mu_{ij|a}\eta_{ij} \qquad 4.1.23$$

We can see geometrically from Figure 4.1.1 that $y(\theta)$ is the sum of the weighted direction tensors of the initial stress and the strain increment, with the initial stress direction having a weighting of 1 and the strain increment direction having a weighting of $\theta\Gamma$.

By restating the consistency condition of 4.1.21:

$$0 = q_{|a}y(\theta) - \Delta\bar{e}^{pl}\left(3G\theta + H_{|\theta}^{*}\right) - \bar{S}_{|\theta} \qquad 4.1.24$$

we can define the yield surface f as:

$$f = q_{|a}y(\theta) - \Delta\bar{e}^{pl}\left(3G\theta + H_{|\theta}^{*}\right) - \bar{S}_{|\theta} \qquad 4.1.25$$

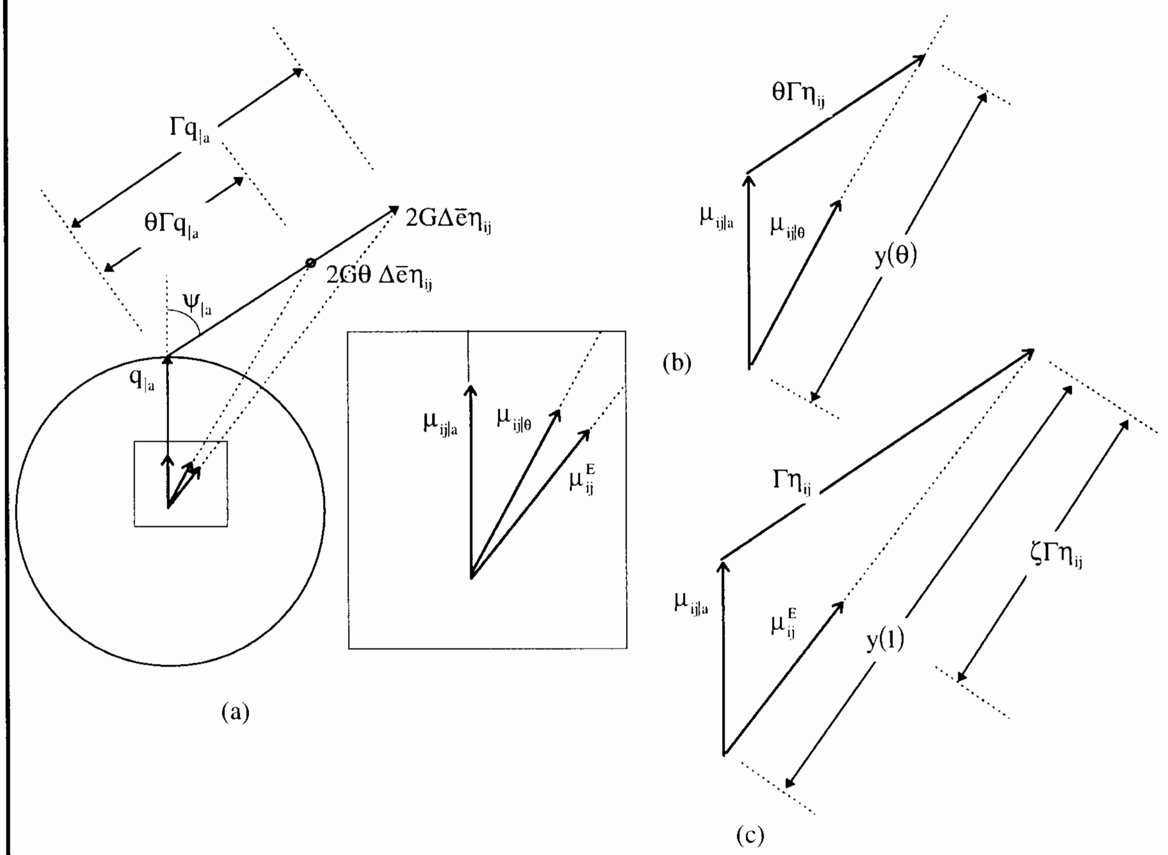

Figure 4.1.1 Diagrammatic representations of relative variables $y(\theta)$ and Γ

To solve for $\Delta\bar{e}^{pl}$ we use a local Newton-Raphson scheme

$$\Delta\bar{e}^{pl|(n+1)} = \Delta\bar{e}^{pl|(n)} - \frac{f(\Delta\bar{e}^{pl|(n)})}{f'(\Delta\bar{e}^{pl|(n)})} \qquad 4.1.26$$

$$f' = \frac{df}{d\Delta\bar{e}^{pl}}$$

$$= -\left[3G\theta + \frac{3}{2}H'_{|\theta}\right] - H_{|\theta}\frac{(\Delta\bar{e}^{pl})^{\frac{1-p}{p}}}{p(D\Delta t)^{\frac{1}{p}}} - D_{2|\theta}H'_{|\theta} \qquad 4.1.27$$

The format of the above equations are essentially the same as those derived for the backward Euler scheme of Section 3.2 except that the values are evaluated at a proportion of θ of the total strain increment. Equation 4.1.27 also contains consistency conditions for the Bauschinger effect induced by kinematic hardening.

Once again, the iterative scheme of Equation 4.1.26 requires a starting value. For larger time steps, $\Delta\bar{e}^{pl}$ approaches $\Delta\bar{e}$. The equivalent plastic strain developed is less, however, for non-proportional loading, $\psi_{|a} > 0$. To take this into account, we assign:

$$\Delta \bar{e}^{pl|(1)} = \zeta \Delta \bar{e} \qquad 4.1.28$$

where

$$\zeta = \frac{y(1) - 1}{\Gamma} \qquad 4.1.29$$

The geometrical interpretation of ζ can be seen in Figure 4.1.1 (c). The use of Equation 4.1.28 may be considered as using a plastic predictor, which comes close to the true solution for large time steps.[15]

4.2 Stress Update Procedure

Once the yield function f is zeroed, Equation 4.1.24 is satisfied, and we therefore have:

$$\bar{S}_{|\theta} = q_{|a} y(\theta) - \Delta \bar{e}^{pl} \left(3G\theta + H^{*}_{|\theta} \right) \qquad 4.2.1$$

Since the consistency condition requires:

$$q_{|\theta} = \bar{S}_{|\theta} \qquad 4.2.2$$

we have:

$$q_{|\theta} = q_{|a} y(\theta) - \Delta \bar{e}^{pl} \left(3G\theta + H^{*}_{|\theta} \right) \qquad 4.2.3$$

In order to obtain the final equivalent stress, we must linearly interpolate Equation 4.2.3 to $\theta = 1$:

$$q_{|b} = \bar{S}_{|b} = q_{|a} y(1) - \left(3G + H^{*}_{|b} \right) \Delta \bar{e}^{pl} \qquad 4.2.4$$

The solution at $t = t_{|b}$ is:

$$\mu_{ij|\theta_\psi} = \frac{\mu_{ij|a} + \theta_\psi \Gamma \eta_{ij}}{y(\theta_\psi)} \qquad 4.2.5$$

$$e^{pl}_{ij|b} = e^{pl}_{ij|a} + \Delta \bar{e}^{pl} \mu_{ij|\theta_\psi} \qquad 4.2.6$$

$$S_{ij|b} = \frac{2}{3} q_{|b} \mu_{ij|b} \qquad 4.2.7$$

$$B_{ij|b} = B_{ij|a} + \left(H_{|b} - H_{|a}\right)\mu_{ij|\theta_\psi} \qquad 4.2.8$$

$$\sigma_{ij|b} = S_{ij|b} + B_{ij|b} + \delta_{ij}\left(\sigma_{kk|a} + K\Delta e_{kk}\right) \qquad 4.2.9$$

Where the parameter θ_ψ in Equation 4.2.5 will be discussed in Section 4.4, and has the same limits as θ.

It is important to note the distinction between the generalised midpoint method adopted here, and that presented in Section 3.1.2. In the original definition, the full stress tensor at θ is calculated and linearly projected to the endpoint. In the above derivation, the equivalent stress at θ, $q_{|\theta}$, is used to linearly project to obtain the final equivalent stress $q_{|b}$. This method allows us to choose freely the direction of the final stress after linear interpolation of the equivalent values.

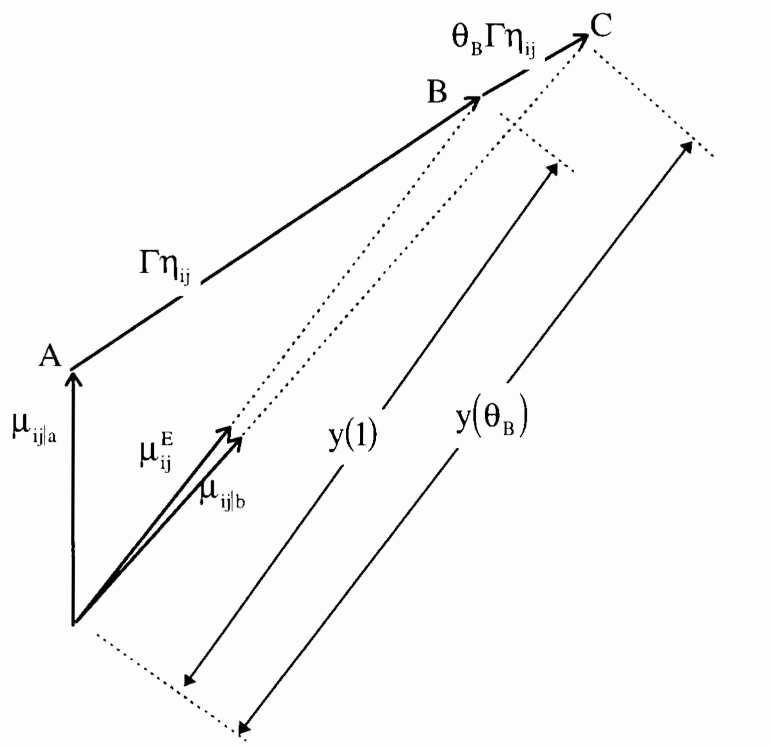

Figure 4.2.1 The variable return direction

In the original radial return algorithm, the final stress direction $\mu_{ij|b}$ is chosen to match the direction to the elastic predictor stress S^E_{ij}, i.e.:

$$\mu_{ij|b} = \mu^E_{ij} = \frac{3}{2}\frac{S^E_{ij}}{q^E} \qquad 4.2.10$$

In order to remain flexible in our choice of return direction, we will allow $\mu_{ij|b}$ to be chosen according to a single parameter θ_B:

$$\mu_{ij|b} = \frac{\theta_B \Gamma \eta_{ij} + \mu_{ij|a}}{y(\theta_B)} \qquad 4.2.11$$

Figure 4.2.1 shows geometrically that θ_B represents the ratio $\frac{AC}{AB}$. Setting $\theta_B = 1$ recovers the standard radial return method. We will examine the determination of θ_B in Section 4.4.

4.3 Calculating θ (Optimising the Collation Point)

Following the work of Fotiu[15] we will see how to determine a value for θ without incurring much computational expense, whilst compensating for the under performance of the backward Euler integration scheme in the realm of increasing rate dependency where elastic and plastic strains are of the same order of magnitude.

In essence, we need to be able to use some sort of estimate of the equivalent plastic strain increment and the non-linear functions which depend on it, and work backwards to obtain a value for θ.

We will begin by assuming that the function governing the permissible stresses can be split into a rate dependent and rate independent part. This is true when using the Couper Symonds overstress factor.

$$\overline{S} = D_2 H \equiv X(\dot{\overline{e}}^{pl}) k(\overline{e}^{pl}) \qquad 4.3.1$$

The functions $X(\dot{\overline{e}}^{pl})$ and $k(\overline{e}^{pl})$ will be used only for the determination of θ, with:

$$\begin{aligned} X(\dot{\overline{e}}^{pl}) &= D_2(\dot{\overline{e}}^{pl}) \\ k(\overline{e}^{pl}) &= H(\overline{e}^{pl}) \end{aligned} \qquad 4.3.2$$

We will need the inverse and derivative of the latter with respect to its argument:

$$X^{-1}(X(\dot{\overline{e}}^{pl})) = \left| D\left[X(\dot{\overline{e}}^{pl}) - 1\right]^p \right| \qquad 4.3.3$$

$$X'(\dot{\overline{e}}^{pl}) = \frac{(\dot{\overline{e}}^{pl})^{\frac{1-p}{p}}}{pD^{\frac{1}{p}}} \qquad 4.3.4$$

The result of the inverse function is a plastic strain rate. Since plastic strain is irreversible, we enforce the absolute value constraint in Equation 4.3.3.

We now approximate the non-linear functions (evaluated at θ) with:

$$\overline{S}_{|\theta} \rightarrow X\left(\frac{\Delta\overline{e}^{pl}}{\Delta t}\right)k^{|*} \quad 4.3.5$$

$$H^{*}_{|\theta} \rightarrow \theta H^{|*} \quad 4.3.6$$

$$y(\theta) = \theta\Gamma + \cos\psi_{|a} \quad 4.3.7$$

where:

$$k^{|*} = k\left(\overline{e}^{pl|*}\right) \quad 4.3.8$$

$$H^{|*} = \frac{3}{2}\frac{H\left(\overline{e}^{pl}_{|a} + \zeta\Delta\overline{e}\right) - H\left(\overline{e}^{pl}_{|a}\right)}{\zeta\Delta\overline{e}} \quad 4.3.9$$

$$\overline{e}^{pl|*} = \overline{e}^{pl}_{|a} + \frac{\zeta\Delta\overline{e}}{2} \quad 4.3.10$$

We now substitute these into the consistency condition (Equation 4.2.1) to get:

$$\theta = \frac{X\left(\frac{\Delta\overline{e}^{pl}}{\Delta t}\right) - \overline{q}}{3G\Delta\overline{e} - \left(3\overline{G} + \overline{H}^{|*}\right)\Delta\overline{e}^{pl}} \quad 4.3.11$$

where

$$\overline{G} = \frac{G}{k^{|*}} \qquad \overline{H}^{|*} = \frac{H^{|*}}{k^{|*}} \qquad \overline{q} = \frac{q_{|a}\cos\psi_{|a}}{k^{|*}} \quad 4.3.12$$

Equation 4.3.7 is an asymptotic approximation derived from $\Gamma >> 1$. Thus we can expect Equation 4.3.11 to be accurate only for large increments. For small increments the value of θ only has a small impact on the accuracy of results.[15]

We need an analytical expression for $\Delta\overline{e}^{pl}$ to substitute into Equation 4.3.7. We start with:

$$\dot{S}_{ij} = 2G\left(\dot{e}_{ij} - \dot{e}^{pl}_{ij}\right) - H'\dot{e}^{pl}_{ij} \quad 4.3.13$$

where:

$$\dot{e}_{ij} = \dot{\bar{e}}\eta_{ij} \qquad 4.3.14$$

We contract on μ_{ij} and note that:

$$\dot{q} = \dot{S}_{ij}\mu_{ij} \qquad 4.3.15$$

By using Equation 4.3.14 and Equation 4.3.15, we can integrate Equation 4.3.13 across the limits $t_{|a}$ to $t > t_{|a}$ to obtain:

$$\bar{S}_{|t} - \bar{S}_{|a} = 3G\dot{\bar{e}}\zeta\left(t - t_{|a}\right) - \left(3G + H^{*}_{|t}\right)\left(\bar{e}^{pl}_{|t} - \bar{e}^{pl}_{|a}\right) \qquad 4.3.16$$

Using Equation 4.3.5 and:

$$\left.\begin{array}{l} \Delta\bar{e}^{pl}(t) = \bar{e}^{pl}_{|t} - \bar{e}^{pl}_{|a} \\ \Delta t(t) = t - t_{|a} \end{array}\right\} \qquad 4.3.17$$

$$\varepsilon = \frac{\Delta\bar{e}}{\Delta t}\frac{1}{\dot{e}_0} \qquad 4.3.18$$

$$x = \bar{e}^{pl}_{|t} - \bar{e}^{pl}_{|a} \qquad 4.3.19$$

$$T = \left(t - t_{|a}\right)\dot{e}_0 \qquad 4.3.20$$

where $\dot{e}_0$ is a reference strain rate, given as a material parameter, we obtain:

$$X(\dot{x}) + \left(3\bar{G} + \bar{H}^{|*}\right)x = 3\bar{G}\varepsilon T + \bar{q} \qquad 4.3.21$$

The solution to the differential equation is given as:

$$x^{|*}(T) = c\varepsilon T - \frac{X\left(c\varepsilon\right) - \bar{q}}{3\bar{G} + \bar{H}^{|*}}\left(1 - \exp(-\kappa T)\right) \qquad 4.3.22$$

where:

$$c = \frac{3\bar{G}}{3\bar{G} + \bar{H}^{|*}} \qquad \kappa = \left(3\bar{G} + \bar{H}^{|*}\right)\frac{c\varepsilon - X^{-1}\left(\bar{q}\right)}{X\left(c\varepsilon\right) - \bar{q}} \qquad 4.3.23$$

Now:

$$\Delta\bar{e}^{pl} = x^{|*}_{|b} = x^{|*}\left(T_{|b} = \dot{e}_0\Delta t\right) \qquad 4.3.24$$

We can introduce the above equations into Equation 4.3.11 to get:

$$\theta = \frac{X\left(\frac{x_{|b}^{|*}}{T_{|b}}\right) - \bar{q}}{\left[X(c\varepsilon) - \bar{q}\right]\left[1 - \exp\left(-\omega\kappa T_{|b}\right)\right]} \qquad 4.3.25$$

$$\omega = X'\left(X^{-1}(\bar{q})\right)\frac{c\varepsilon - X^{-1}(\bar{q})}{X(c\varepsilon) - \bar{q}} \qquad 4.3.26$$

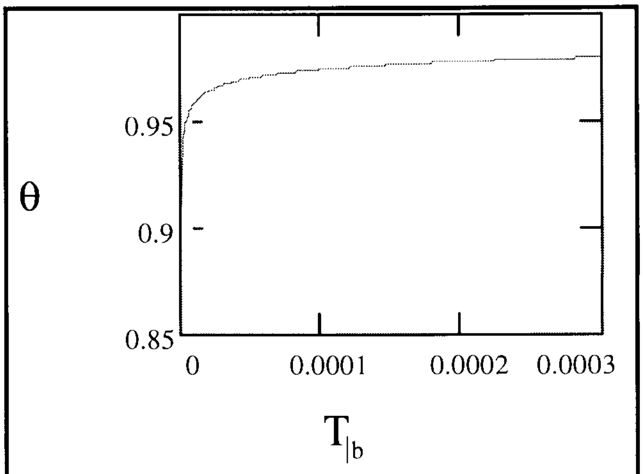

Figure 4.3.1 Example curve of θ plotted against the dimensionless variable $T_{|b}$.

The factor ω is introduced in order to assert the limit $\mathrm{Lim}_{T_{|b}\to 0}\theta = 0.5$, corresponding to the second order accuracy limit for θ. Since Equation 4.3.25 is only a good approximation for large $T_{|b}$, the required limit my not be achieved exactly for small $T_{|b}$. It is then necessary to enforce the condition $\theta \geq 0.5$.

Figure 4.3.1 shows how $\mathrm{Lim}_{T_{|b}\to\infty}\theta = 1.0$.

Unfortunately the above method, although numerically inexpensive, still requires the input of an external parameter. The choice of θ depends strongly on $\dot{e}_0$, as we see by the definition of $T_{|b}$ in Equation 4.3.24. The choice of this parameter will be examined in the section dealing with the numerical results.

4.4 Calculating θ_B (Optimising the Return Direction)

We have seen from previous sections and Figure 4.2.1, that the return direction for the generalised midpoint rule $\mu_{ij|b}$ is not the standard radial return direction μ_{ij}^{E} from the full elastic predictor. We also saw that the plastic strain direction $\mu_{ij|\theta}$ is assumed to be constant in the direction of the partial elastic predictor $\left(S_{ij|a} + 2G\theta\,\Delta e_{ij}\right)$. The following relation was laid down in Section 4.2 for the final stress update direction:

$$\mu_{ij|b} = \frac{\theta_B \Gamma \eta_{ij} + \mu_{ij|a}}{y(\theta_B)} \qquad 4.4.1$$

Since the equivalent values of stress and strain have been determined from the integration procedure, we can now choose a preferred partial elastic predictor $\left(S_{ij|a} + 2G\theta_\psi \Delta e_{ij}\right)$ from which to take the plastic strain direction $\mu_{ij|\theta_\psi}$, which is assumed constant over the increment. The choice of optimum return direction thus comes down to choosing the optimum plastic strain direction for the increment. We must then ensure that the final stress update direction remains consistent with our choice of plastic strain direction.

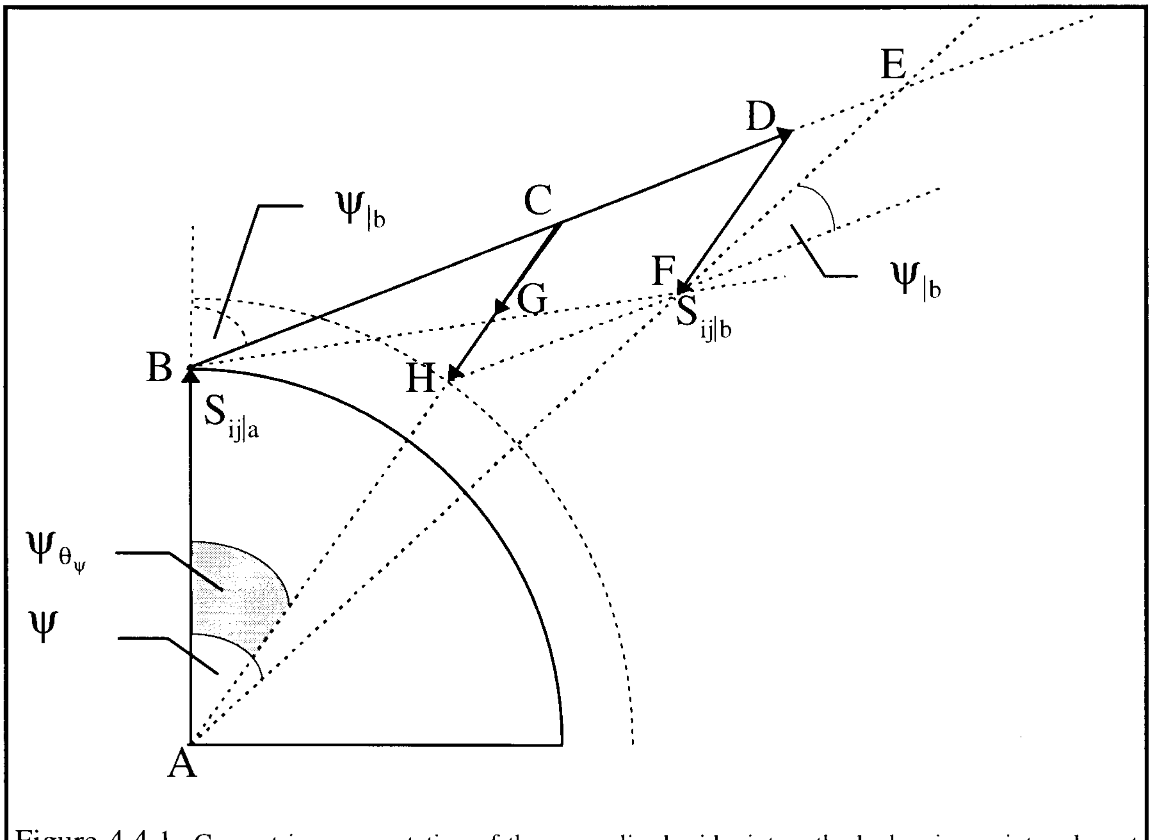

Figure 4.4.1 Geometric representation of the generalised midpoint method, showing points relevant to calculating θ_B

θ_B represents the factor necessary to correct the elastic predictor stress so that the stress update direction is consistent with the direction AF in Figure 4.4.1. We can thus define the relationship between θ_B and θ_ψ .

$$\frac{BC}{BD} = \theta_\psi \quad ; \quad \frac{BE}{BD} = \theta_B \qquad 4.4.2$$

We can extend CG to H such that:

$$DF = CH \qquad 4.4.3$$

where

$$\left.\begin{array}{l} DF = 3G\Delta\bar{e}^{pl} \\ CG = 3G\theta_\psi \Delta\bar{e}^{pl} \\ \Rightarrow DF \| CH \\ \Rightarrow BE \| HF \end{array}\right\} \qquad 4.4.4$$

By making use of the above relations and:

$$\frac{AF}{FE} = \frac{CD}{DE} = \frac{AH}{HC} \qquad 4.4.5$$

We can prove that:

$$\theta_B = \frac{AG}{AH} \qquad 4.4.6$$

Equation 4.4.6 is fully defined since we know:

$$AG = q_{|a} y(\theta_\psi) - \theta_\psi \left(3G + H^*_{|b}\right)\Delta\bar{e}^{pl} \qquad 4.4.7$$

$$AH = q_{|a} y(\theta_\psi) - \left(3G + H^*_{|b}\right)\Delta\bar{e}^{pl} \qquad 4.4.8$$

We will assume that the stress direction vector remains in the plane common to $\mu_{ij|a}$ and η_{ij} throughout the increment, and we can thus define an angle of rotation of the stress as (Figure 4.4.1):

$$\psi = \psi_{|a} - \psi_{|b} \qquad 4.4.9$$

After subincrementation to obtain an exact stress rotation, we can write an angular error:

$$\psi_{|error} = \psi_{|exact} - \psi \qquad 4.4.10$$

Krieg and Krieg[5] established an exact solution for ψ for rate independent perfect plasticity. Fotiu[15] proposed that the same solution be applied to the case of isotropic hardening with rate dependence, and set about relating the solution presented by Krieg and Krieg to θ_ψ.

After Krieg and Krieg:

$$S_{ij|b} = z_1 S_{ij|a} + z_2 2G\Delta\bar{e}\eta_{ij} \qquad 4.4.11$$

where

$$z_1 = \frac{2C}{1 + C^2 + (1 - C^2)\cos\psi_{|a}} \qquad 4.4.12$$

$$z_2 = \frac{1}{\Gamma_2}\frac{1 - C^2 + (1 - C)^2\cos\psi_{|a}}{1 + C^2 + (1 - C^2)\cos\psi_{|a}}$$

$$C = \exp(-\Gamma_2) \qquad 4.4.13$$

and Γ_2 is evaluated using the final values for the increment, i.e.:

$$\Gamma_2 = \frac{3G\Delta\bar{e}}{q_{|b}} \qquad 4.4.14$$

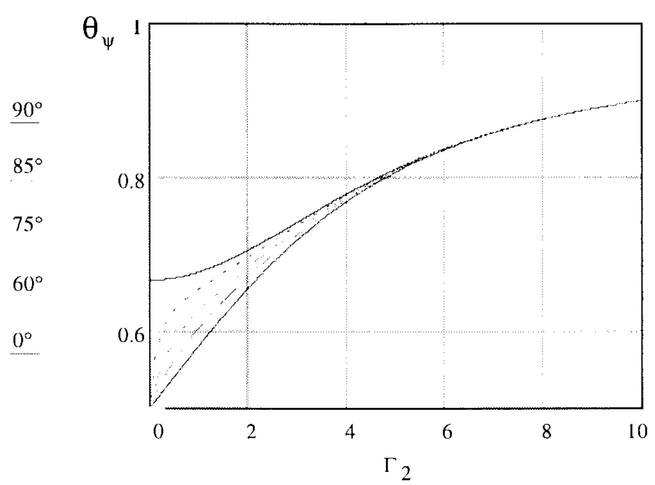

Figure 4.4.2 θ_ψ varying according to various load sizes and directions

Fotiu, asserts colinearity between the left and right of:

$$z_1 S_{ij|a} + z_2 2G\Delta\bar{e}\eta_{ij} = S_{ij|a} + 2G\Delta\bar{e}\eta_{ij} - \left(2G + \frac{2}{3}H^*_{|b}\right)\Delta\bar{e}^{pl}\mu_{ij|\theta_\psi} \qquad 4.4.15$$

$$(1 - z_1)\mu_{ij|a} + (1 - z_2)\Gamma_2\eta_{ij} \cong \mu_{ij|a} + \theta_\psi\Gamma_2\eta_{ij} \qquad 4.4.16$$

The result being:

$$\theta_\psi = \frac{1-z_2}{1-z_1} = \frac{\Gamma_2\left[1+C^2+\left(1-C^2\right)\cos\psi_{|a}\right]-\left(1-C^2\right)-(1-C)^2\cos\psi_{|a}}{\Gamma_2\left[(1-C)^2+\left(1-C^2\right)\cos\psi_{|a}\right]} \qquad 4.4.17$$

The behaviour of θ_ψ is demonstrated in Figure 4.4.2. We can see that $\theta_\psi \to 1$ as $\Gamma_2 \to \infty$. This is in accordance with the conclusions of Ortiz and Popov.[13]

4.5 The Consistent Tangent Modulus

With the integration and stress update fully defined, we must now derive a consistent tangent modulus for the end of the increment.

$$d_{ijkl} = \frac{\partial \Delta S_{ij}}{\partial \Delta e_{kl}} \qquad 4.5.1$$

$$\Delta S_{ij} = 2G\left(\Delta e_{ij} - \Delta e_{ij}^{pl}\right) = 2G\left(\Delta e_{ij} - \Delta\bar{e}^{pl}\mu_{ij|\theta_\psi}\right) \qquad 4.5.2$$

$$\partial\Delta S_{ij} = 2G\left(\partial\Delta e_{ij} - \partial\Delta\bar{e}^{pl}\mu_{ij|\theta_\psi} - \Delta\bar{e}^{pl}\partial\mu_{ij|\theta_\psi}\right) \qquad 4.5.3$$

By differentiating Equation 4.2.5:

$$\partial\mu_{ij|\theta_\psi} = \partial\left[\frac{\theta_\psi\Gamma\eta_{ij} + \mu_{ij|a}}{y(\theta_\psi)}\right] \qquad 4.5.4$$

As mentioned, the values of θ_ψ and θ only have a significant effect on the result for large increments. We also saw that the solutions for θ_ψ and θ are relatively constant for larger increments. Since the two quantities are calculated anew in each step, they should strictly be differentiated too, but for simplicity, they will be assumed constant due to the aforementioned reasons.

$$\partial\left[\Gamma\eta_{ij}\right] = \frac{3G}{q_{|a}}\partial\Delta e_{ij} \qquad 4.5.5$$

and by taking the partial derivative of the first part of Equation 4.1.24:

$$\partial y(\theta) = \frac{2G\theta}{q_{|a}}\partial\Delta e_{ij} \qquad 4.5.6$$

to get:

$$\partial\mu_{ij|\theta_\psi} = \frac{\theta_\psi 3G}{q_{|a} y(\theta_\psi)}\left[\delta_{ijkl} - \frac{2}{3}\mu_{ij|\theta_\psi}\mu_{ij|\theta_\psi}\right]\partial\Delta e_{kl} \qquad 4.5.7$$

Now we differentiate Equation 4.2.1 with respect to the equivalent plastic strain increment to obtain:

$$\partial\Delta\bar{e}^{pl} = \frac{\theta 2G}{\zeta_{\bar{S}} + \theta 3G + \frac{3}{2}\zeta_H}\mu_{ij|\theta}\partial e_{ij} \qquad 4.5.8$$

where:

$$\zeta_{\bar{S}} = \frac{d\bar{S}_{|\theta}}{d\Delta\bar{e}^{pl}} = \theta D_{2|\theta}\left[\frac{\partial H}{\partial\bar{e}^{pl}}\right]_{|\theta} + \left[\frac{\partial D_2}{\partial\Delta\bar{e}^{pl}}\right]_{|\theta} H_{|\theta} \qquad 4.5.9$$

$$\zeta_H = \frac{dH_{|\theta}}{d\Delta\bar{e}^{pl}} = \theta\left[\frac{\partial H}{\partial\bar{e}^{pl}}\right]_{|\theta} \qquad 4.5.10$$

If we make use of:

$$\lambda = \frac{\theta\Gamma + \cos\psi_{|a}}{\theta_\psi\Gamma + \cos\psi_{|a}}\cdot\frac{y(\theta_\psi)}{y(\theta)} \qquad 4.5.11$$

Equation 4.5.8 becomes:

$$\partial\Delta\bar{e}^{pl} = \frac{\theta 2G\lambda}{\zeta_{\bar{S}} + \theta 3G + \frac{3}{2}\zeta_H}\mu_{ij|\theta_\psi}\partial e_{ij} \qquad 4.5.12$$

We now substitute back into Equation 4.5.3:

$$\partial\Delta S_{ij} = 2G\begin{bmatrix}\delta_{ijkl}\left(1 - \Delta\bar{e}^{pl}\frac{\theta 3G}{q_{|a}y(\theta_\psi)}\right) + \\ \left(\mu_{ij}\mu_{ij}\right)_{|\theta_\psi}\left[\frac{2}{3}\Delta\bar{e}^{pl}\frac{\theta 3G}{q_{|a}y(\theta_\psi)} - \frac{\theta 2G\lambda}{\zeta_{\bar{S}} + \theta 3G + \frac{3}{2}\zeta_H}\right]\end{bmatrix}\partial\Delta e_{kl} \qquad 4.5.13$$

4.6 Summary of Optimised Midpoint with Variable Return

1. Calculate the deviatoric equivalents of all variables.
2. Assume zero plastic strain and calculate an equivalent elastic predictor stress (Equation 2.2.14).
3. If the predictor stress lies within the yield surface, update all stresses elastically. Go to step 8.
4. Calculate the optimised midpoint using Equation 4.3.25.
5. Use local Newton-Raphson scheme to zero the yield function and obtain equivalent plastic strain increment.
6. Calculate final equivalent stress using Equation 4.2.4.
7. If $\psi|_a > 0$ then calculate the return directions using the method described in Section 4.4, else the loading is proportional and $\mu_{ij|a} = \mu_{ij|\theta_\psi} = \mu_{ij|b}$. Return the final equivalent stress and equivalent plastic strain to tensor values.
8. Calculate the consistent tangent modulus.

5. TEST PROCEDURES

The stress update procedures for the Backward Euler Method with Radial Return, and the Optimised Midpoint Method with Variable Return were implemented using FORTRAN (see Appendix C). The code was also written in such a way that it could be incorporated into the Finite Element Analysis package ABAQUS. In addition to this, simple input parameters can restrict the dynamic determination of θ in the OMM in order to simulate the Generalised Midpoint Method.

5.1 Error Measures

The analysis of errors in a procedure using the Von Mises yield criterion is simplified due to the fact that the conversion to and from deviatoric space is error free, and thus any errors which occur can be represented in a two dimensional system. The yield surface remains circular in this plane, thus allowing us to define any point in the Von Mises deviatoric space by an angle and a radius.

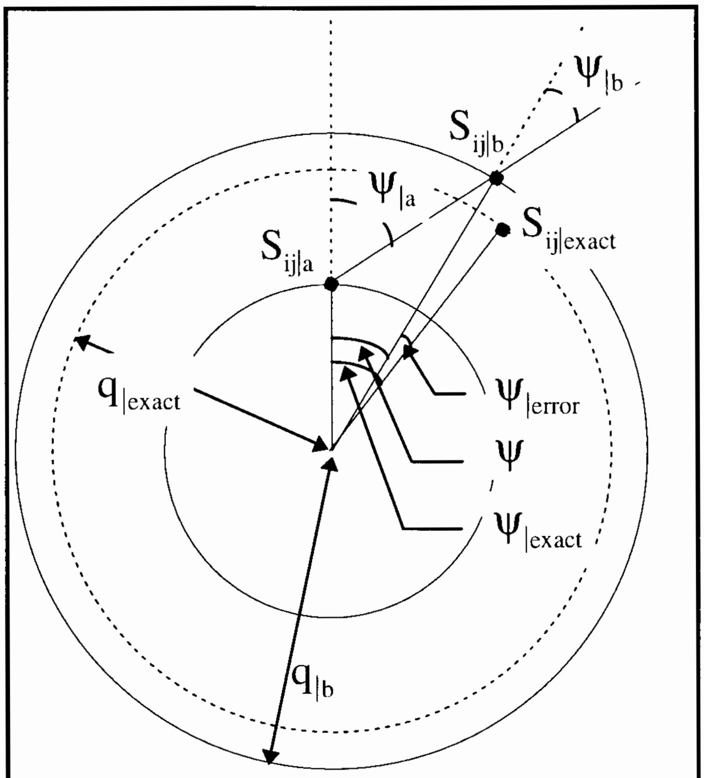

Figure 5.1.1 Definition of predicted and exact radial and angular components of a solution

Firstly, before we can gauge any errors, we need an exact solution. Except for rate independent perfect plasticity (as pointed out by Krieg and Krieg[5]) we cannot establish an explicit solution for the radius of the yield surface and the angle of rotation from the initial stress direction. The standard method of obtaining an 'exact' solution in the FEM is by dividing the increments into as many subincrements (n) as is necessary so that any further subdivision does not result in a substantially different solution. The number of subincrements used to obtain the exact results in the tests procedures which follow will be given.

Using Figure 5.1.1 we define the errors associated with the radial and angular components of the final stress states predicted.

The angular error (in degrees or radians) is given by:

$$\psi_{|error} = \psi - \psi_{|exact} \tag{5.1.1}$$

where:

$$\psi = \psi_{|a} - \psi_{|b} \tag{5.1.2}$$

and the radial error (as a percentage) is given by:

$$q_{|error} = 100\frac{q_{|b} - q_{|exact}}{q_{|exact}} \tag{5.1.3}$$

5.2 Material Description

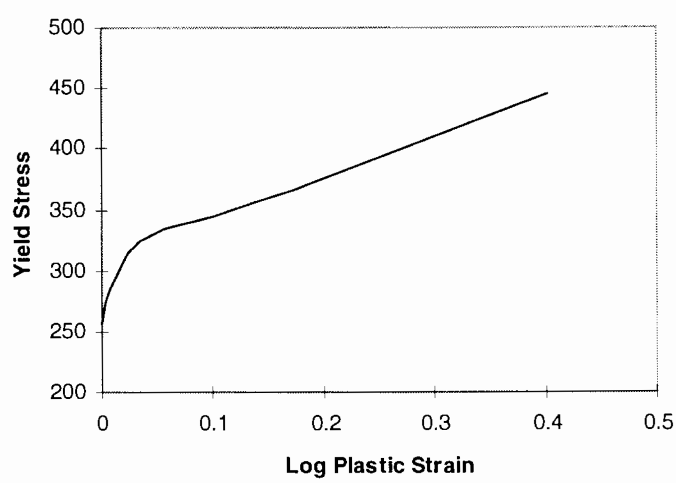

Figure 5.2.1 Plastic yield curve for material used to conduct tests.

Since there is such a wide variety of materials, along with their material curves, we will only use one material which displays the essential characteristics of most materials applicable to this study. The material chosen is a ductile metal which conforms well to the Von Mises condition with rate dependent behaviour. The yield curve is of a general shape and magnitude similar to those of a wide range of ductile metals, which are the primary users of the algorithms to be tested. We see from

Figure 5.2.1, the material displays isotropic hardening, with non-linear isotropic hardening in the initial regions of plastic strain. The Young's modulus of the material is E=197 MPa and Poisson's ratio is υ =0.3.

Performance of the algorithms for the perfectly plastic cases will be tested using a fixed yield stress corresponding to the initial yield stress of the above material. Kinematic hardening will not be tested.

5.3 Representation of Results

There are two common means of displaying results. The first is by using a simple graph of error, stress, a stress component or some other quantity versus either time, or equivalent plastic strain, etc. This is however limiting in that we can only see one pass of the test procedure per data set plotted on the graph.

A preferred method used for analysing the angular and radial errors to be used is the contour plot, which can show the variation of the error quantities with respect to two axes. These are commonly referred to as isoerror maps, since they represent contours of constant error over the domain and range of the two axes chosen.

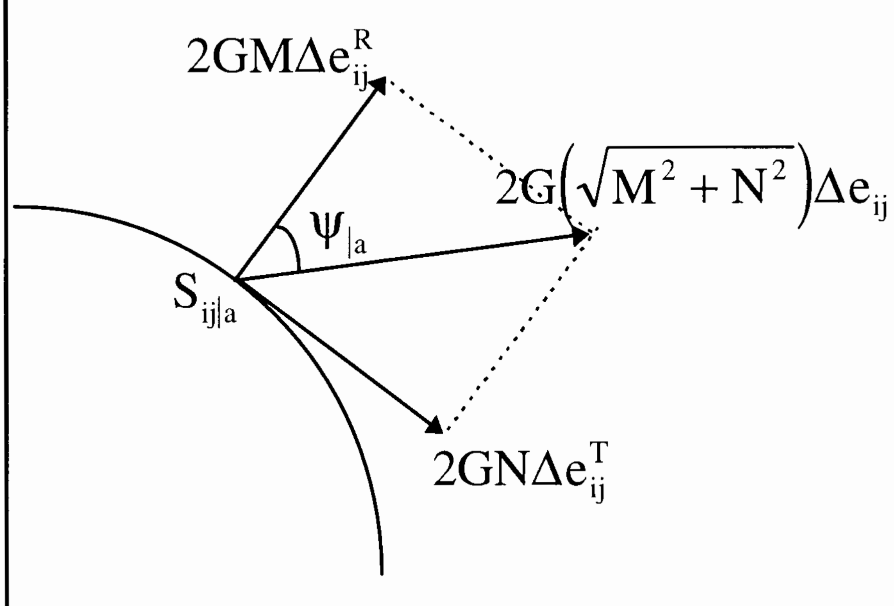

Figure 5.3.1 Tangential and radial strain increments

Since we are primarily interested in determining the performance of a new algorithm against the traditional and widely used backward Euler radial return scheme, and we are particularly concerned with the accuracy of the yield surface radius and return direction, we need to devise tests which will introduce load steps of varying sizes and direction. From a given (converged) stress point, we can give the algorithms a strain increment which has a known size and direction. This strain increment can be broken up into tangential and radial components. The size of the tangential and radial increments can be expressed as a ratio of the initial strain magnitude.

I.e. if we set:

$$\Delta\bar{e}^{R} = \Delta\bar{e}^{T} = \bar{e}_{|a} \qquad 5.3.1$$

Then:

$$\Gamma = \sqrt{M^{2} + N^{2}} \qquad 5.3.2$$

and we can plot the errors each on a set of axes defined by M and N.

6. TEST RESULTS

Results for tests are contained in Appendix A and Appendix B and will be referred to by Figure number, e.g. Figure A.23.

6.1 Error Contours

6.1.1 Test Parameters

In order to gauge the performance of the algorithms during the initial, non-linear hardening phase, the initial stress used for the tangential-radial tests was the static yield stress for the material, 256 MPa. The strain at this point was chosen as (in vector form):

$$\varepsilon_{|a} = e_{|a} = \begin{bmatrix} 0 & 0 & 0 & 0.1951 \cdot 10^{-2} & 0 & 0 \end{bmatrix}^T \quad 6.1.1$$

This was convenient for dealing with the volumetric components of the strain, since the volumetric strains of Equation 6.1.1 are zero, and the radial $(\sim)^R$ and tangential $(\sim)^T$ strain increments are easily definable:

$$\begin{aligned} \Delta e^R_{ij} &= e_{ij|a} \\ \Delta e^T &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0.1951 \cdot 10^{-2} & 0 \end{bmatrix}^T \end{aligned} \quad 6.1.2$$

The number of subincrements to obtain the correct solution was set at $n = 50\Gamma$. The difference in exact solution between $n = 50\Gamma$ and $n = 40\Gamma$ was a fraction of a percentage and considered to be sufficiently small for confidence.

We will plot the results for the ranges:

$$\left.\begin{array}{l} 0 \le M \le 5 \\ 0 \le N \le 5 \end{array}\right\} \text{Steps of } 0.5 \quad 6.1.3$$

Since the magnitude of the time increment is important when considering rate dependent materials, in order to maintain a constant strain rate regardless of the increment size, we will define a relative time increment:

$$\Delta \bar{t} = \frac{\Delta t}{\Gamma} \quad 6.1.4$$

With a given relative time increment, the total time increment to be used for each pass of the solution scheme can be simply calculated from Equation 6.1.4.

6.1.2 Discussion of Results

6.1.2.1 Effects of various reference strain rates on the OMM

In order to compare the performance of the Optimised Midpoint Method (OMM{ $\dot{e}_0$ }) with other common integration and update procedures, such as the Backward Euler Method with Radial Return (BE) and Generalised Midpoint algorithms (GMM{ θ }), we need to obtain a value for the material constant $\dot{e}_0$. The larger the value for $\dot{e}_0$ chosen, the higher will be the values chosen for θ, and vice versa. I.e. OMM{ $\dot{e}_0 >> 0$ } $\rightarrow$ BE except for very small increments where the OMM{ $\dot{e}_0 \in R$ } $\rightarrow$ Midpoint Method by design.

The series of tests conducted to determine this parameter were conducted for a highly rate dependent isotropic material (p=5, D=40) with a high strain rate ($\Delta\bar{t}$ =0.001).

Figure A.1(b) to Figure A.4(b) show how the choice of θ increases in general with increasing $\dot{e}_0$. Figure A.1(a) to Figure A.4(a) show that the tendency to over predict $q_{|b}$ for small θ becomes a tendency to under predict $q_{|b}$ as θ increases. The radial isoerrors are at a minimum percentage in Figure A.4(a) for $\dot{e}_0$ =0.1. Larger values of $\dot{e}_0$ were not tested since this caused numerical difficulties in the calculation of Equation 4.3.25.

With a reasonable choice of the material constant to work with, we now need to compare the OMM to the GMM and BE.

6.1.2.2 Rate independent perfect plasticity

We now compare the various methods' performance for the simplest case of rate independent (p=1, D=100) perfect plasticity, in Figure A.5 to Figure A.8.

As can be expected, the GMM{0.5} of Figure A.5 and GMM{0.75} of Figure A.6 correctly predict the equivalent stress and return direction for pure and near pure proportional loading. However, the forward linear projection of intermediate values at θ to final values leads to large over prediction of the yield surface for tangential loading. The angular errors of the GMM{0.5} and GMM{0.75} vary considerably over the test range.

Since the BE in Figure A.7 satisfies the consistency condition at $\theta = 1$, the error in yield surface radius is a fraction of a percent, and attributable primarily to the range of the convergence criteria used in the Newton-Raphson algorithm. The only two drawbacks of the BE in this situation are the assumption that the return direction is the same as the direction to the elastic predictor stress, and the fact that the second order accuracy of the GMM{0.5} for small increments is not utilised. The return direction results in a maximum angular error of 12.7°, which agrees with the findings of Krieg and Krieg.[5]

Figure A.8(c) shows that the OMM{0.1} utilises the accuracy of the fully explicit backward Euler integration for all but the most proportional cases and enforces the GMM{0.5} for small increments. The accuracy of the return direction used by the OMM{0.1} (Figure A.8(b)) leads to a maximum error of 0.6°. The variation of θ_{ψ} over the test range is shown by Figure A.8(d). This reduction in angular error, combined with the ability to dynamically predict a θ, leads to excellent results for rate independent perfectly plasticity.

6.1.2.3 Rate dependent perfect plasticity

Figure A.9 to Figure A.12 show that the results for the rate dependent (p=2, D=60, $\Delta\bar{t}$=0.01) perfectly plastic case are very similar to those of the rate independent case.

The BE (Figure A.11) and OMM{0.1} (Figure A.12) outperform the GMM{0.5} and GMM{0.75} in both the prediction of the yield surface radius and the return direction. It is clear from the results of this section and Section 6.1.2.2 that the choice of a constant $\theta = \theta_{\psi}$ leads to excessive errors when used in an algorithm which has to update stresses resulting from arbitrary strain increments. If an algorithm cannot predict the yield surface radius accurately for perfect plasticity, it can be discarded as inaccurate.

Once again, the OMM and BE are similar in the resulting radial isoerrors. The maximum angular isoerror of 0.8° for the OMM{0.1} demonstrates it power over the BE radial return assumption, which has a maximum error of over 12°.

6.1.2.4 Rate independent isotropic hardening

We will now begin to examine the performance of the BE and OMM{0.1} under similar conditions to those of Section 6.1.2.2 and 6.1.2.3, except we will use the isotropic hardening case. We look first at rate independent (p=1, D=100) isotropic hardening. The results for the GMM{0.5} and GMM{0.75} are not given since, as expected, the errors for these two cases remain vastly larger than those of the OMM{0.1} and BE.

Both the BE (Figure A.13) and the OMM{0.1} (Figure A.14) show errors of less than 1% in predicting the yield surface radius. Since the tests are not rate dependent, the choice of θ (Figure A.14(c)) by the OMM{0.1} is very close to a fully backward difference assumption with the exception of large proportional increments. Once again, second order accuracy for small increments is enforced.

6.1.2.5 Rate dependent isotropic hardening

The BE (Figure A.15(a)) and the OMM{0.1} (Figure A.16(a)) show that for rate dependent (p=2, D=60, $\Delta\bar{t}$=0.01) isotropic hardening, the percentage errors in the yield surface radii are less than 1% in both cases. Once again, since the tests were lowly rate dependent, the OMM{0.1} does not differ significantly from the BE in the integration of the scalar rate equations. The angular accuracy of the OMM{0.1} (Figure A.16(c)) is once again far superior to that of the BE (Figure A.15(b)).

6.1.2.6 Rate dependent isotropic hardening with varying strain rates

A series of tests were conducted for more highly rate dependent (p=5, D=40) isotropic hardening. The results for the BE appear in Figure A.17 to Figure A.21, with the strain rate varying such $\Delta\bar{t}$ decreases from $\Delta\bar{t}=1.0$ to $\Delta\bar{t}=0.0001$. The corresponding results for the OMM{0.1} appear in Figure A.22 to Figure A.26.

Examining the results for the BE shows that the maximum radial isoerrors increase steadily as the strain rate increases. We can see that the OMM{0.1} uses backward Euler integration for the perfectly or near perfectly tangential strain increments of N>1. The highest radial isoerrors occur for values of N slightly more than this, resulting in maximum errors for the OMM{0.1} and BE being the same for all the test cases. The average values for the radial isoerrors increase steadily for both methods, with the BE performing slightly better on average than the OMM{0.1} except for the final test with $\Delta\bar{t}=0.0001$. It is also important to note that for the final tests (Figure A.21 and Figure A.24) the performance of the OMM{0.1} for proportional and near proportional loading is better than that of the BE, the maximum errors being 2% and 6% respectively.

Although the angular isoerrors of the BE decrease as the strain rate increases and those of the OMM{0.1} increase, a glance at the maximum errors and averages show that in general the OMM{0.1} outperforms the BE. The angular errors of the OMM{0.1} are also more consistent during the full range of the tests.

6.2 Stress Curves

6.2.1 Test Parameters

The next set of tests record pertinent stresses during a multistage loading history. The initial stress and strain values are all zero, and the following strain increments are imposed on the material:

$$\Delta\varepsilon^{|(1)} = \begin{bmatrix}4\cdot10^{-2} & -2\cdot10^{-2} & -2\cdot10^{-2} & 0 & 0 & 0\end{bmatrix}^T \qquad 6.2.1$$

$$\Delta\varepsilon^{|(2)} = \begin{bmatrix}0 & 0 & 0 & 4\cdot10^{-2} & 0 & 0\end{bmatrix}^T \qquad 6.2.2$$

$$\Delta\varepsilon^{|(3)} = \begin{bmatrix}0 & 0 & 0 & 1\cdot10^{-2} & 5\cdot10^{-2} & 5\cdot10^{-2}\end{bmatrix}^T \qquad 6.2.3$$

The second and third increments correspond to changes in load direction of 90° and 82° respectively.

The time increment per step used was $\Delta t = 0.0002$ seconds.

In a similar manner to that of Fotiu[15] the stress components $\bar{\sigma}$, σ_{xx} and σ_{xy} are examined with respect to $\bar{e}^{pl}$ or total time. One significant difference with the results of Fotiu is that those results were plotted with respect to T. This value includes the reference strain rate $\dot{e}_0$. Practically, we need to know the stresses irrespective of the choice of the reference strain rate. Thus, for the following tests, the value $\dot{e}_0=0.1$ was again chosen.

The exact result was obtained by dividing each strain increment into n=100 subincrements, while the tests on the OMM and BE were performed using n=3 subincrements.

6.2.2 Discussion of Results

6.2.2.1 Constant θ with high rate dependence test

Figure B.1 shows how small values of θ lead to severe under and over prediction around the exact curves with large highly rate dependent (p=5, D=40) increments. Although the BE is reasonably accurate, it is not the most accurate choice of integration schemes throughout the test, with optimal θ s varying from 1.0 down to 0.7. This demonstrates, once again, the need for a scheme which can optimally vary θ.

6.2.2.2 Rate dependent isotropic tests

The rate dependent test, (p=2, D=50) is shown in Figure B.2 and Figure B.4. The rate dependent test (p=5, D=40) is shown in Figure B.3 and Figure B.5.

We can see from the $\bar{\sigma}$ vs. $\bar{e}^{pl}$ graphs that the equivalent plastic strain and equivalent stress predicted by the OMM{0.1} and BE are accurate, with the OMM{0.1} being slightly more accurate. The σ_{xx} vs. t and σ_{xy} vs. t graphs demonstrate the major improvement of the OMM{0.1} over the BE.

7. CONCLUSIONS

Based on the work completed in this thesis, the following conclusions may be drawn:

1. The Standard Backward Euler Radial Return (BE) and the Optimised Midpoint Method with Variable Return (OMM) provide simple and understandable theoretical bases from which to practically implement numerical stress update procedures for a Von Mises yield model.
2. We can see from the discussions of Section 3, that in theory the OMM and BE are more accurate and stable than other applicable stress update procedures.
3. The stress update procedure developed for the OMM fills a gap in the field of update algorithms by providing a computationally inexpensive means of dynamically choosing the collation point. The separation of the collation point and the plastic strain direction provides an additional means of improving accuracy. This is a distinct difference from previous update procedures, including the BE.
4. The inclusion of a reference strain rate as a material parameter into the formulation of the OMM is a limiting factor in the use of the method, making it necessary to determine the best reference strain rate for each application prior to obtaining test results. Further study into the elimination of this parameter from the derivation would facilitate the use of the method.
5. Practically, isoerror maps show that the BE and OMM outperform the Generalised Midpoint Method (GMM) enough to allow one to discard the latter for all but the smallest increments. The OMM makes use of the second order accuracy of the GMM at small increments in any case, making the GMM obsolete for our purposes.
6. The adaptation of the formulation presented by Krieg and Krieg to determine exactly the plastic strain direction for rate independent perfect plasticity, and its implementation into the OMM leads to an improvement in angular accuracy (in the π plane) over other methods by an order of magnitude. This part of the formulation is short and simple and can be used to slightly modify other methods, such as the BE, with significant results.
7. The isoerror maps show that the improvement of the OMM over the BE in predicting the radius of the yield surface is not as significant as the improvement in predicting the plastic strain direction. The OMM does, however, perform slightly better on average than the BE, particularly at higher strain rates.
8. The stress curves which were extracted from tests show that the OMM outperforms the BE significantly and remains very close to the exact values for all load cases. The equivalent stress values predicted by the two methods are both sufficiently and almost equally accurate, indicating that the variation in the collation point from $\theta=1$ does not have a significant effect on accuracy.
9. There is extra computation time involved in determining the collation point for the OMM. This increases the complexity of the consistent tangent modulus, while resulting in only minor improvements in accuracy of the yield surface radius. The OMM can however exploit the second order accuracy limit for small increments, which most other fixed collation point methods cannot do.
10. The OMM satisfies the first two of the three criterion for an optimal stress update algorithm (Section 3.1.6). It chooses the collation point such that average errors in predicting the yield surface radius seldom exceed 1%. The final stress direction

results in average angular errors of typically less than 1%. Both these errors are more consistent for varying degrees of non-proportionality and rate dependence than those of the BE.

8. REFERENCES

[1] S. Caddemi, "The Mid-Point Rule and the Trapezoidal Rule for the Integration of the Von Mises Linear Hardening Constitutive Laws", *CERECAM*, report 160

[2] L. J. Rencontré and J. B. Martin, "Generalised Trapezoidal Rule for Plasticity", *CERECAM*, report 176

[3] S. Caddemi, "Computational Aspects of the Integration of the Von Mises Linear Hardening Constitutive Laws", *CERECAM*, report 236

[4] L. J. Rencontré, S. Caddemi and J. B. Martin, "The Relationship Between the Generalised Mid-Point and Trapezoidal Rules in Incremental Elasto-Plasticity", *Computer Methods in Applied Mechanics and Engineering*, vol. 96 (1992), pp. 201-212

[5] R. D. Krieg and D. B. Krieg, "Accuracies of Numerical Solution Methods for the Elastic-Perfectly Plastic Model", *Transactions of the ASME Journal of Pressure Vessel Technology*, vol. 99 (1977), pp. 510-515

[6] M. Ortiz and J. C. Simo, "An Analysis of a New Class of Integration Algorithms for Elastoplastic Constitutive Relations", *International Journal for Numerical Methods in Engineering*, vol. 23 (1986), pp. 353-366

[7] J. C. Simo and R. L. Taylor, "A Return Mapping Agorithm for Plane Stress Elastoplasticity", ", *International Journal for Numerical Methods in Engineering*, vol. 22 (1986), pp. 649-670

[8] J. C. Simo and M. Ortiz, "A Unified Approach to the Finite Deformation Elastoplastic Analysis Based on the Use of Hyperelastic Constitutive Equations", *Computer Methods in Applied Mechanics and Engineering*, vol. 49 (1985), pp. 221-245

[9] J. C. Simo and R. L. Taylor, "Consistent Tangent Operators for Rate-Independent Elastoplasticity", *Computer Methods in Applied Mechanics and Engineering*, vol. 48 (1985), pp. 101-118

[10] G. P. Mitchell, "A Solution Algorithm for Elastic-Viscoplastic Solids"

[11] Hibbit Karlsson and Sorensen, Inc., "ABAQUS Theory Manual",

[12] W. F. Hosford and R. M. Caddell, "Metal Forming: Mechanics and Metallurgy", Prentice-Hall, Inc. 1991

[13] M. Ortiz and E. P. Popov, "Accuracy and Stability of Integration Algorithms for Elastoplastic Constitutive Relations", *International Journal for Numerical Methods in Engineering*, vol. 21, 1985, pp 1561-1576

[14] H. L.Schryer, et. al., "Accurate Numerical Solutions for Elastic-Plastic Models", *Transactions of the ASME. Journal of Pressure Vessel Technology*, vol. 101, August 1979, pp. 226-234

[15] P. A. Fotiu, "A modified Generalized Midpoint Rule for the Integration of Rate-Dependent Thermo-Elastic-Plastic Constitutive Equations", *Computer Methods in Applied Mechanics and Engineering*, vol 122, pp. 105-129

[16] J. Lubliner, "Plasticity Theory", Macmillan Publishing Company, 1990

[17] R. Hill, "The Mathematical Theory of Plasticity", Oxford at the Clarendon Press, 1950

[18] R. D. Cook, et. Al., "Concepts and Applications of Finite Element Analysis", John Wiley and Sons, 1981

[19] S. Nemat-Nasser and D. T. Chung, "An Explicit Constitutive Algorithm for Large-Strain, Large-Strain-Rate Elastic-Viscoplasticity", *Computer Methods in Applied Mechanics and Engineering*, vol. 95 (1992), pp. 205-219

[20] O. C. Zienkiewicz and R. L. Taylor, "The Finite Element Method", McGraw-Hill, 1994

APPENDIX A - Isoerror Maps

OMM	Optimised midpoint method with variable return
ref=xxx	Reference strain $\dot{e}_0$ rate for OMM
BE	Backward Euler integration with radial return
GMM	Generalised midpoint rule
0.5	θ for GMM
Isotropic/Perfect	Isotropic hardening/Perfect plasticity
p=x	p rate dependence constant
D=x	D rate dependence constant
dt=xxx	$\Delta\bar{t}$: Time increment per Γ
AveRI	Average absolute Radial Isoerror
xxx	
AveAI	Average absolute Angular Isoerror
xxx	

Table 1: Legend used for Figures in Appendix A

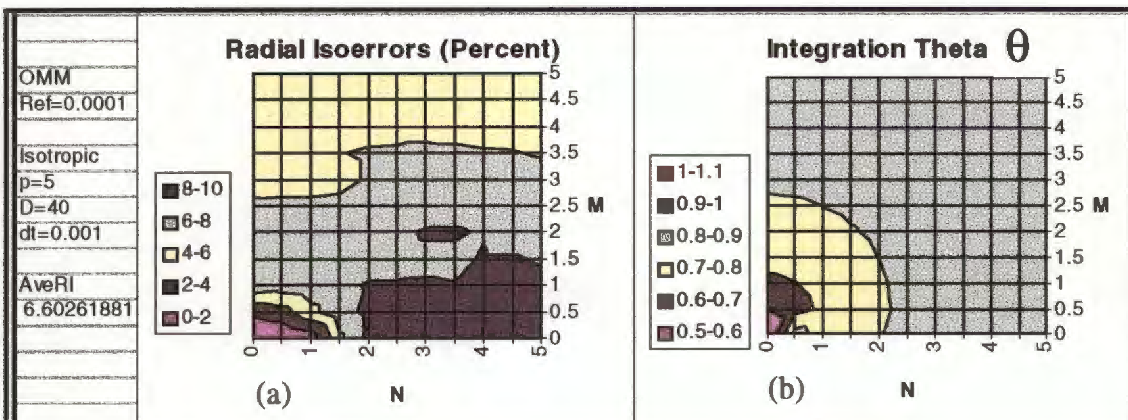

Figure A.1 Effects of various reference strain rates on the OMM

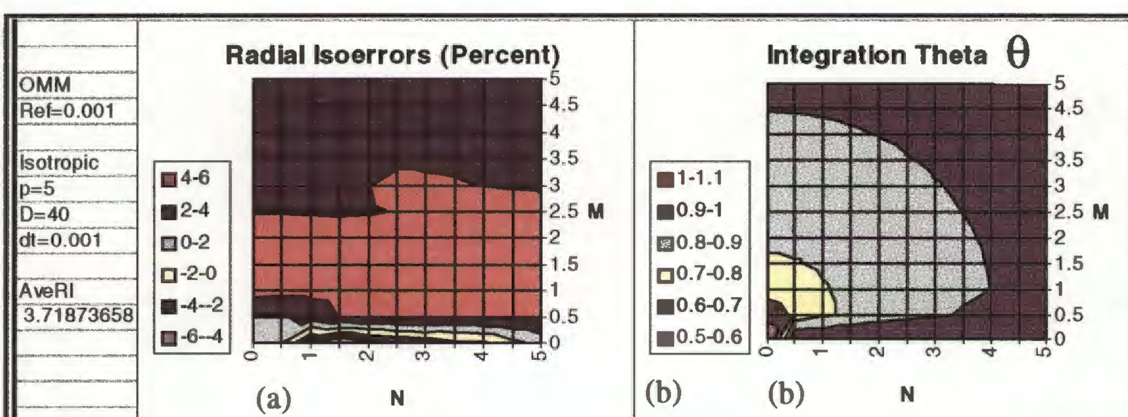

Figure A.2 Effects of various reference strain rates on the OMM

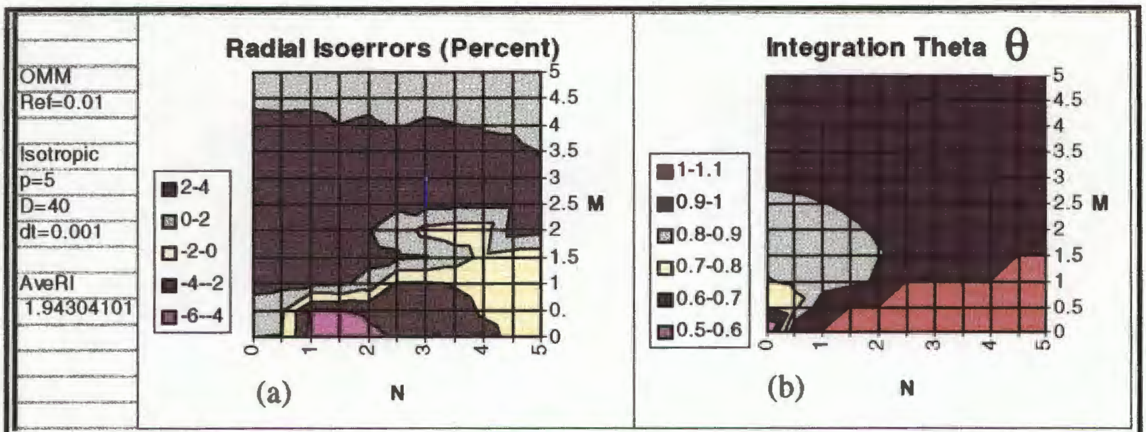

Figure A.3 Effects of various reference strain rates on the OMM

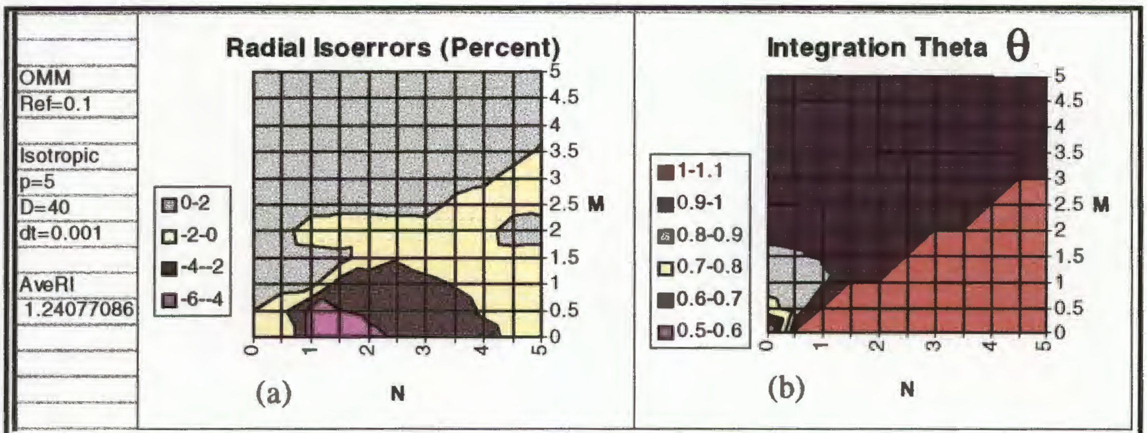

Figure A.4 Effects of various reference strain rates on the OMM

GMM
0.5

Perfect
p=1
D=100
dt=1

AveRI
20.1469034
AveAI
5.25968968

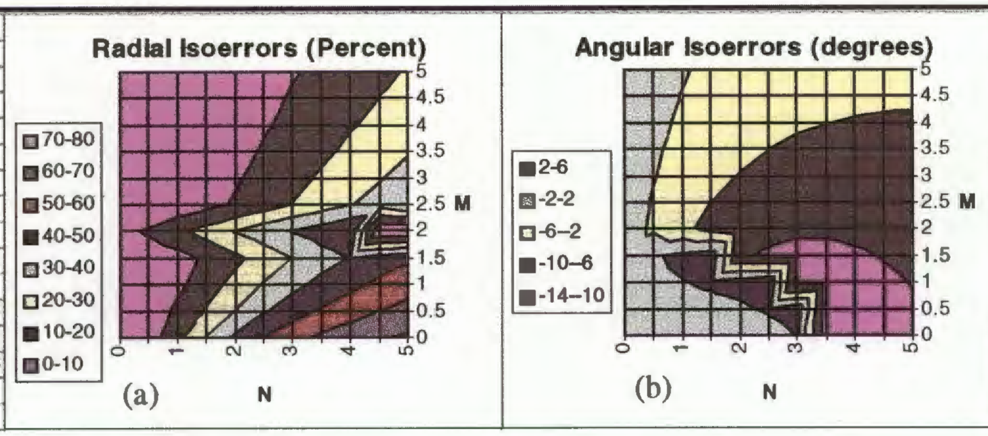

Figure A.5 Midpoint method with rate independent perfect plasticity

GMM
0.75

Perfect
p=1
D=100
dt=1

AveRI
7.72510637
AveAI
2.89385695

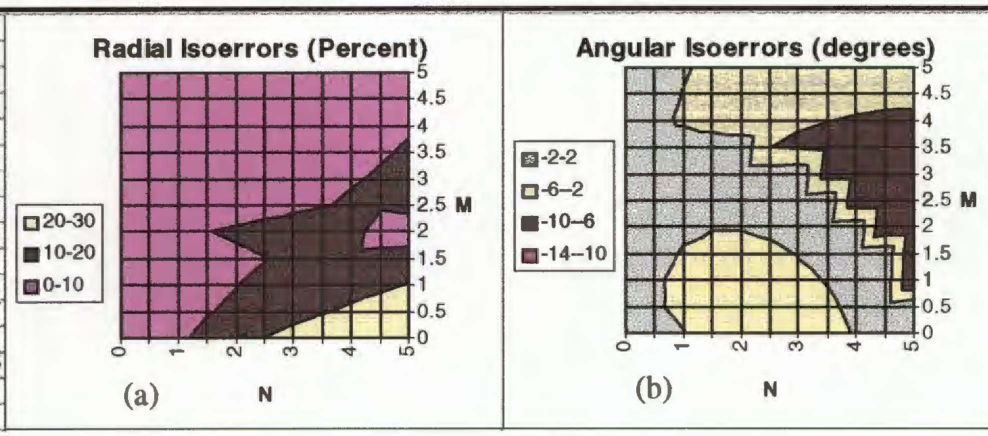

Figure A.6 Generalised midpoint method with rate independent perfect plasticity

BE

Perfect
p=1
D=100
dt=1

AveRI
0.27309903
AveAI
6.62975388

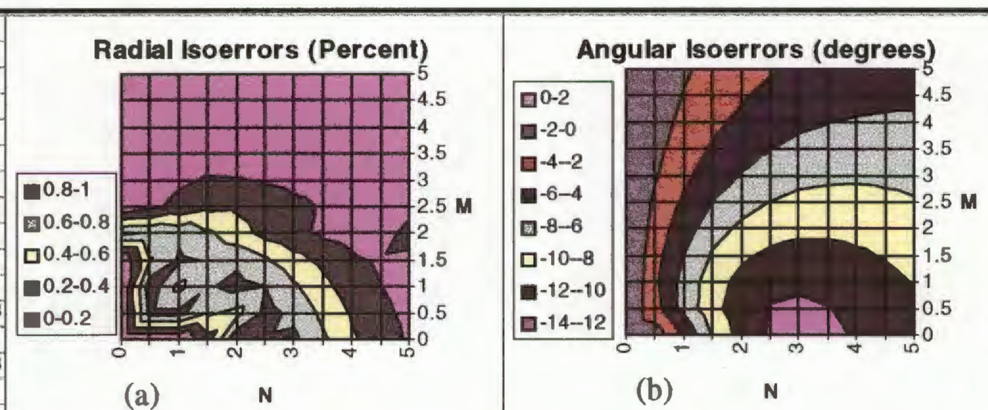

Figure A.7 Backward Euler method with rate independent perfect plasticity

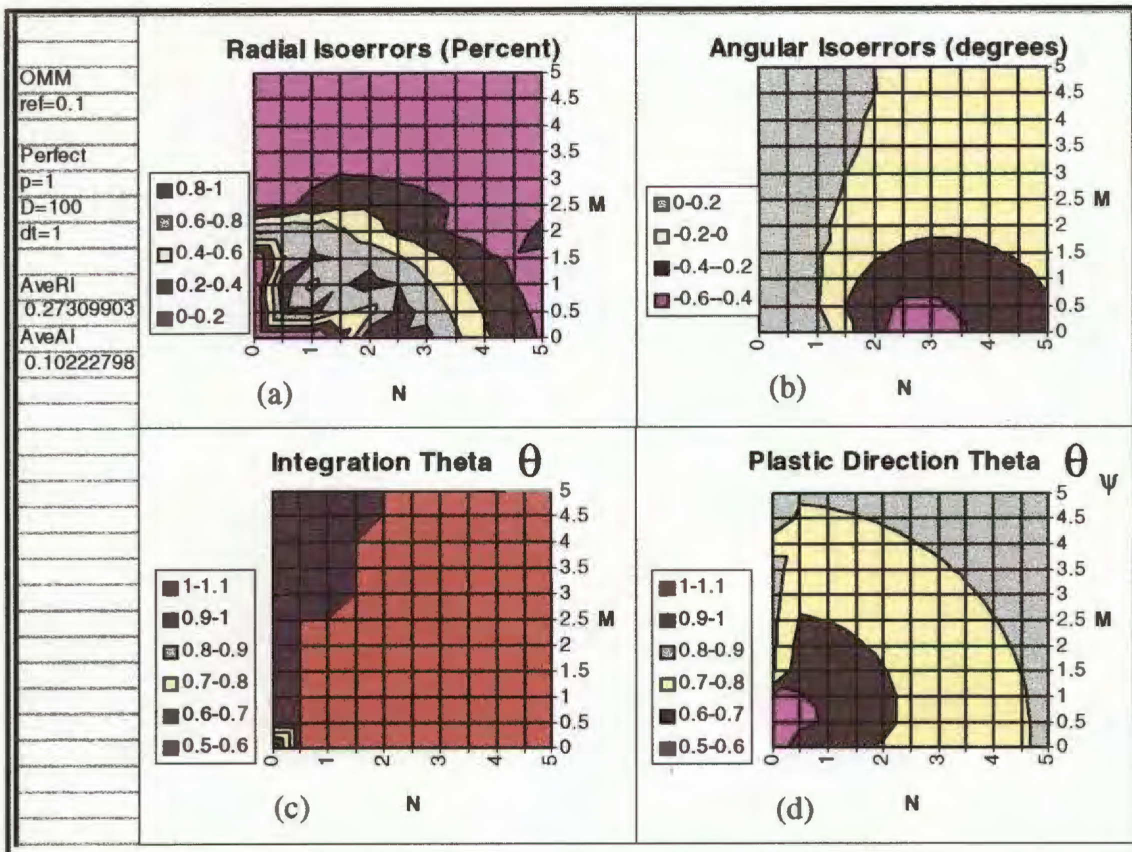

Figure A.8 Optimised midpoint method with rate independent perfect plasticty

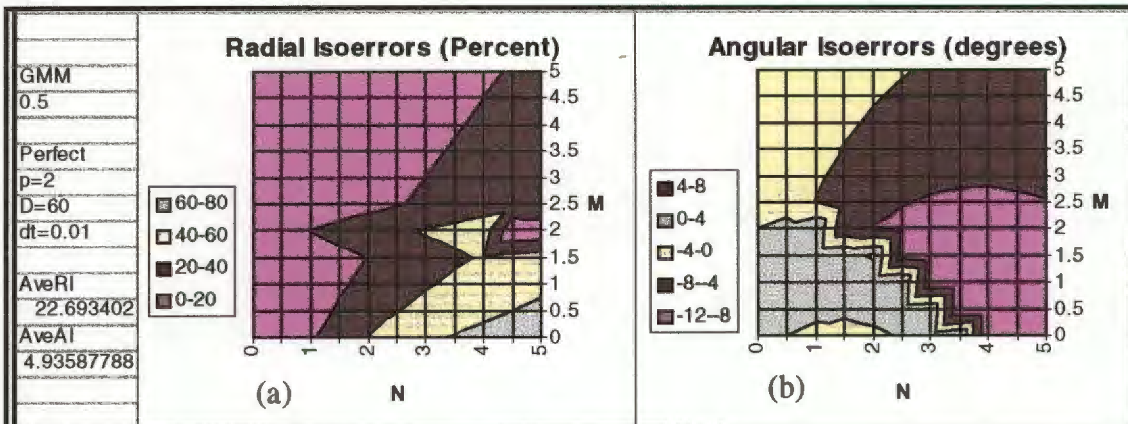

Figure A.9 Midpoint method with rate dependent perfect plasticity

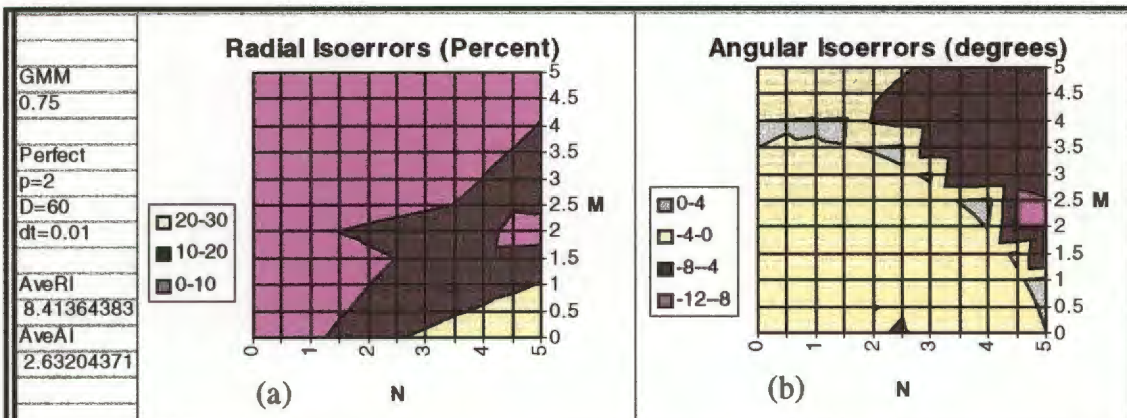

Figure A.10 Generalised midpoint method with rate dependent perfect plasticity

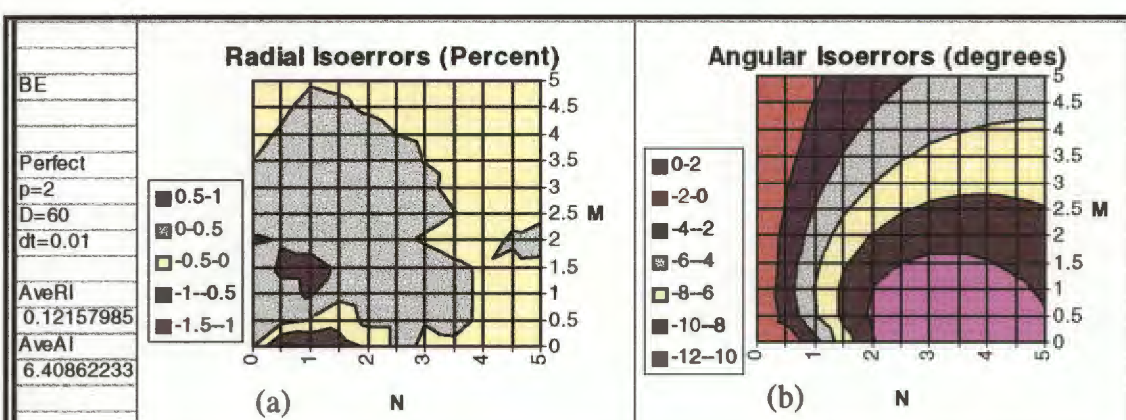

Figure A.11 Backward Euler method with rate dependent perfect plasticity

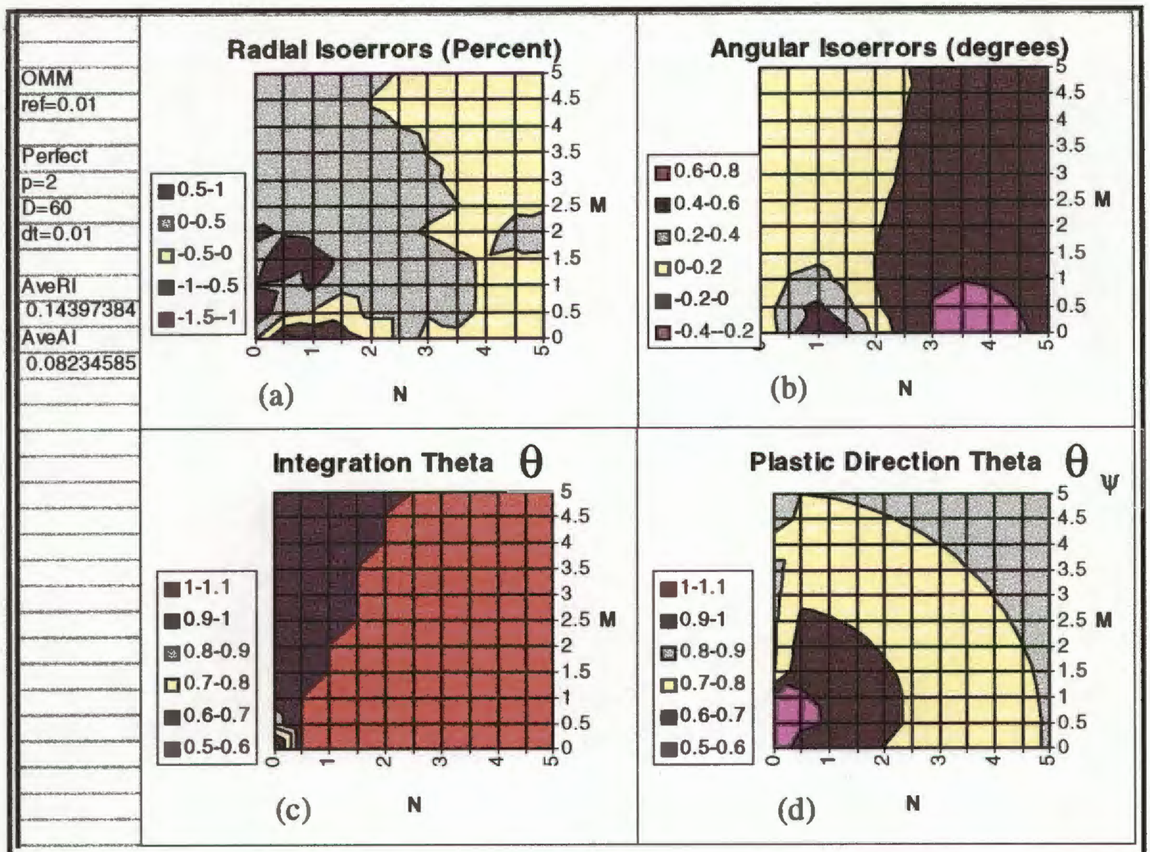

Figure A.12 Optimised midpoint method with rate independent perfect plasticity

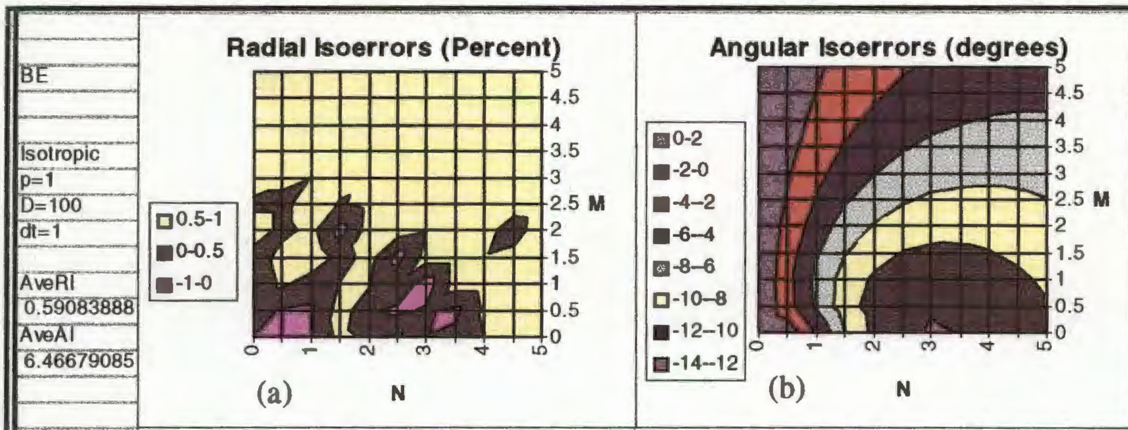

Figure A.13 Backward Euler method with rate independent isotropic hardening

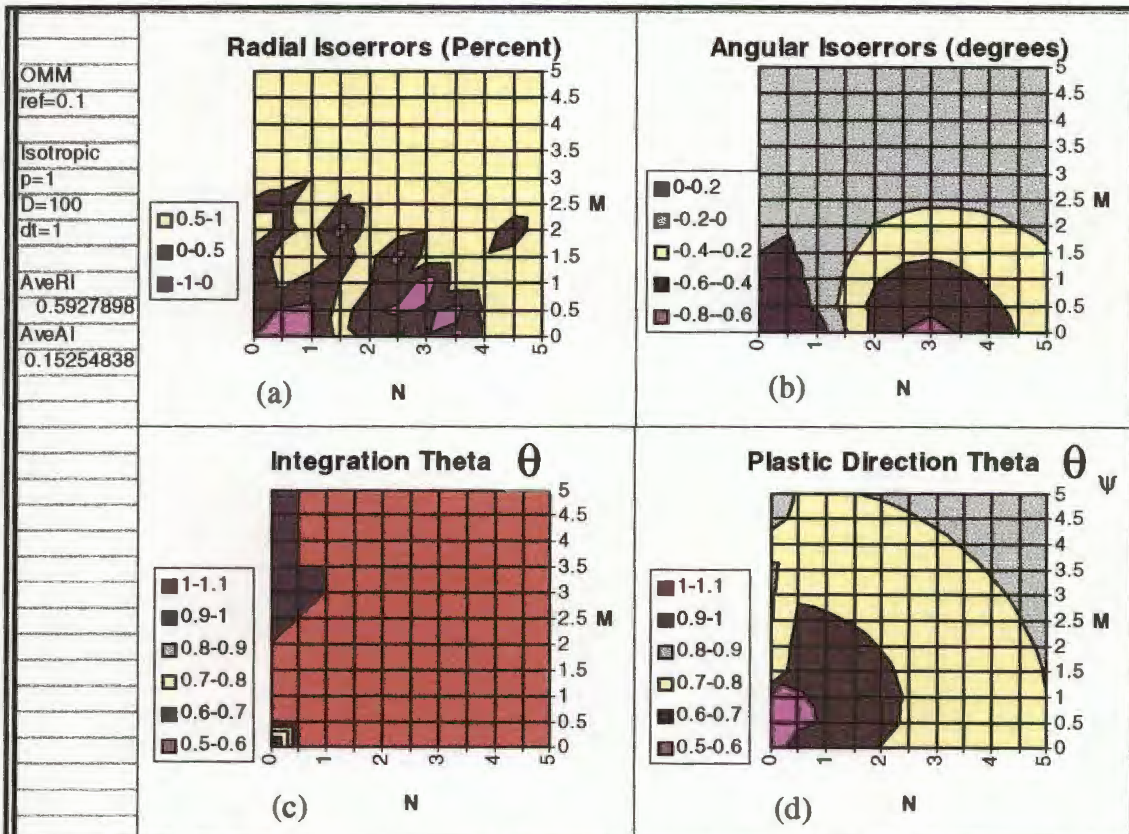

Figure A.14 Optimised midpoint method with rate independent isotropic hardening

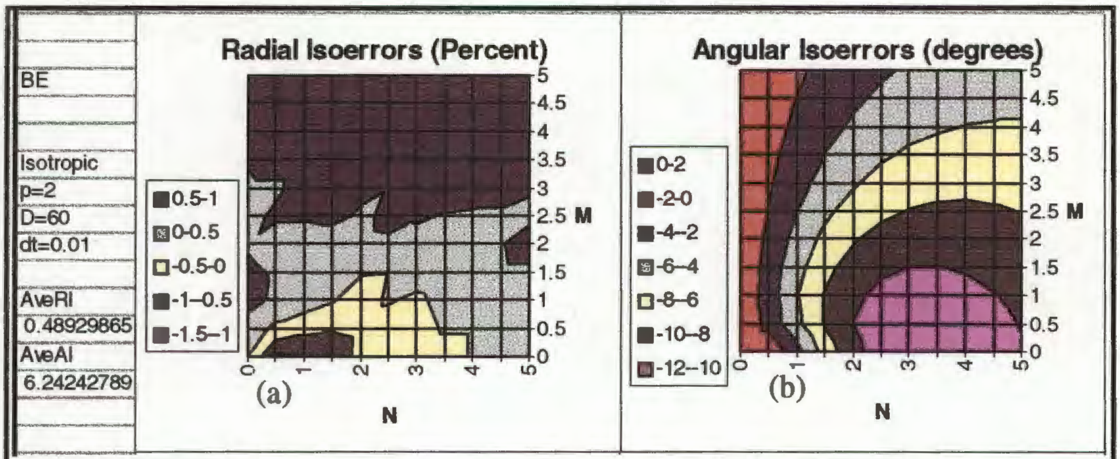

Figure A.15 Backward Euler method with rate dependent isotropic hardening

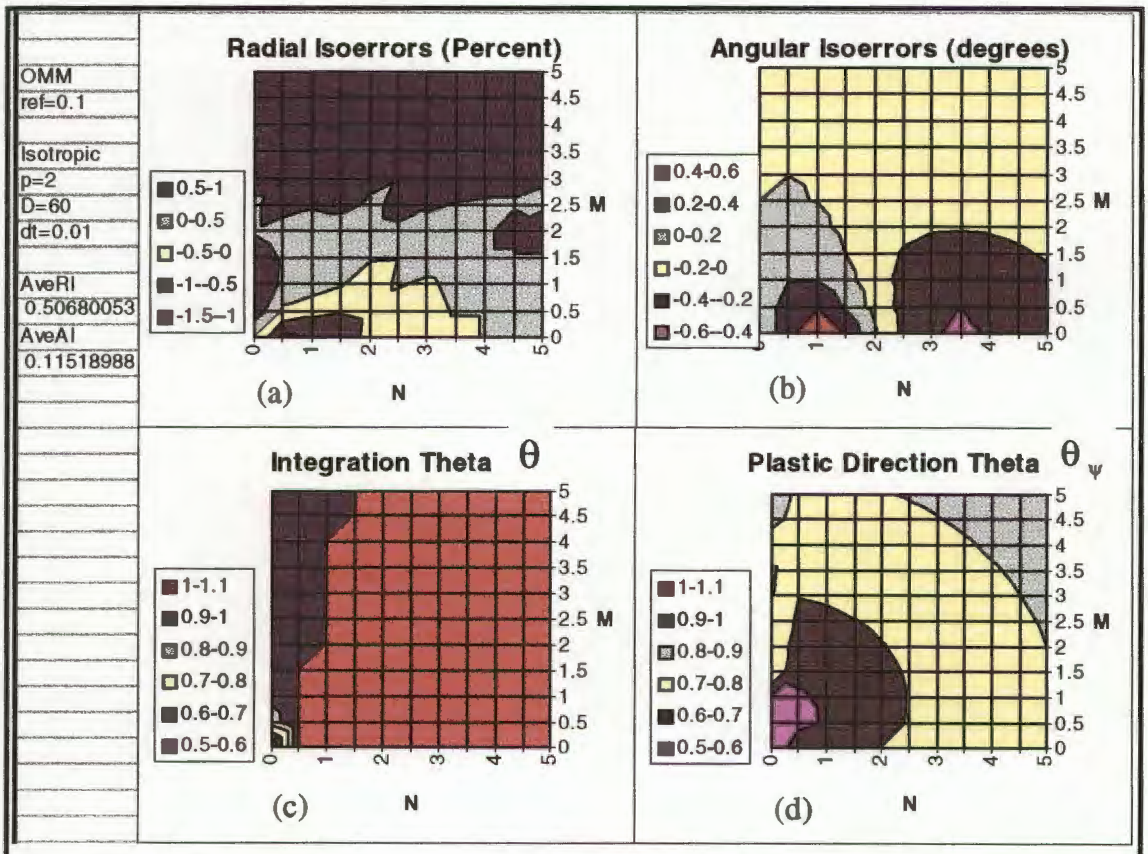

Figure A.16 Optimised midpoint method with rate dependent isotropic hardening

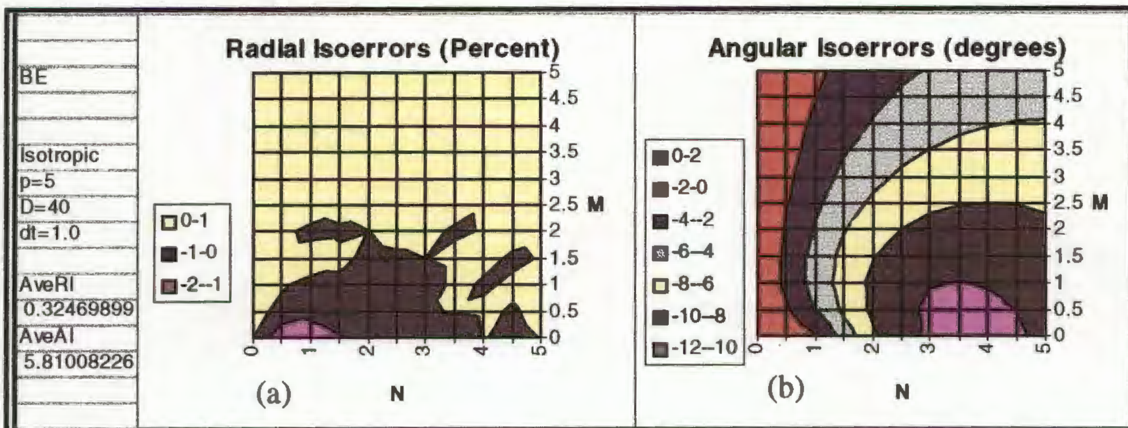

Figure A.17 Rate dependent backward Euler method with varying strain rates

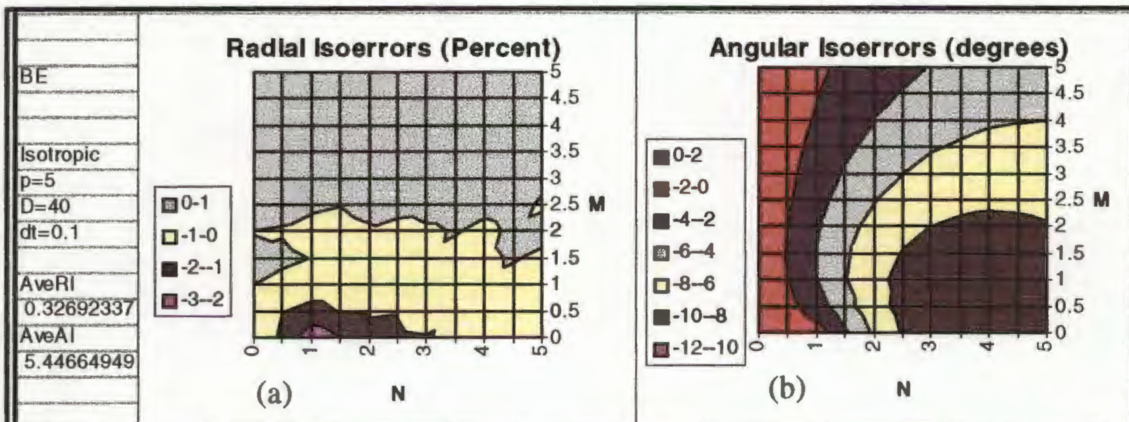

Figure A.18 Rate dependent backward Euler method with varying strain rates

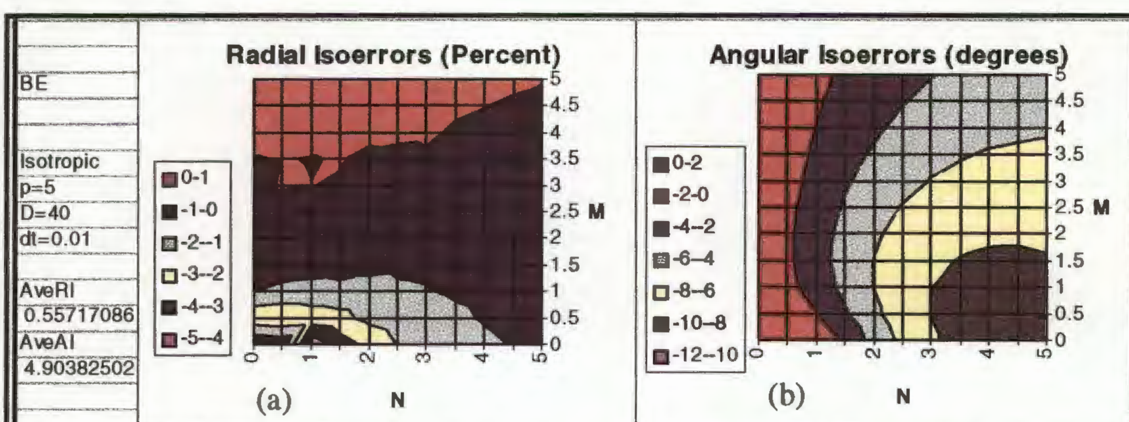

Figure A.19 Rate dependent backward Euler method with varying strain rates

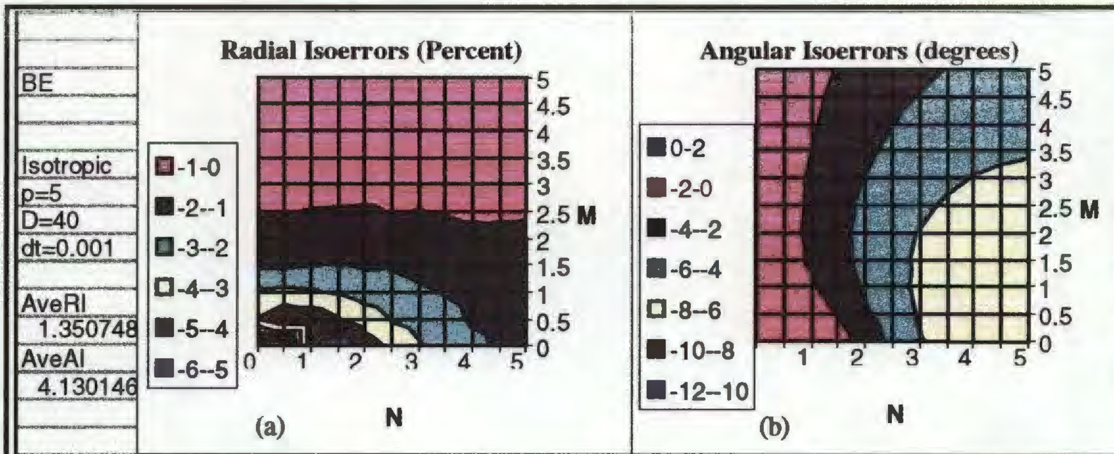

Figure A.20 Rate dependent backward Euler method with varying strain rates

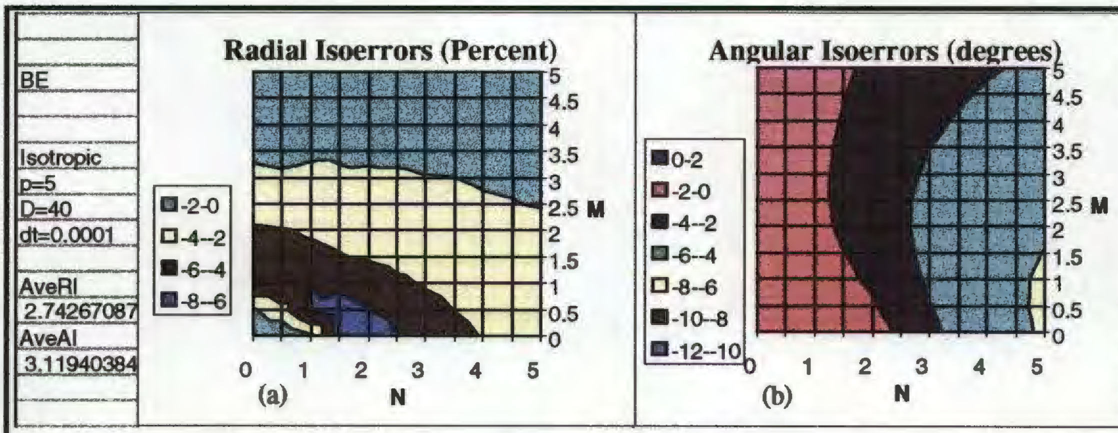

Figure A.21 Rate dependent backward Euler method with varying strain rates

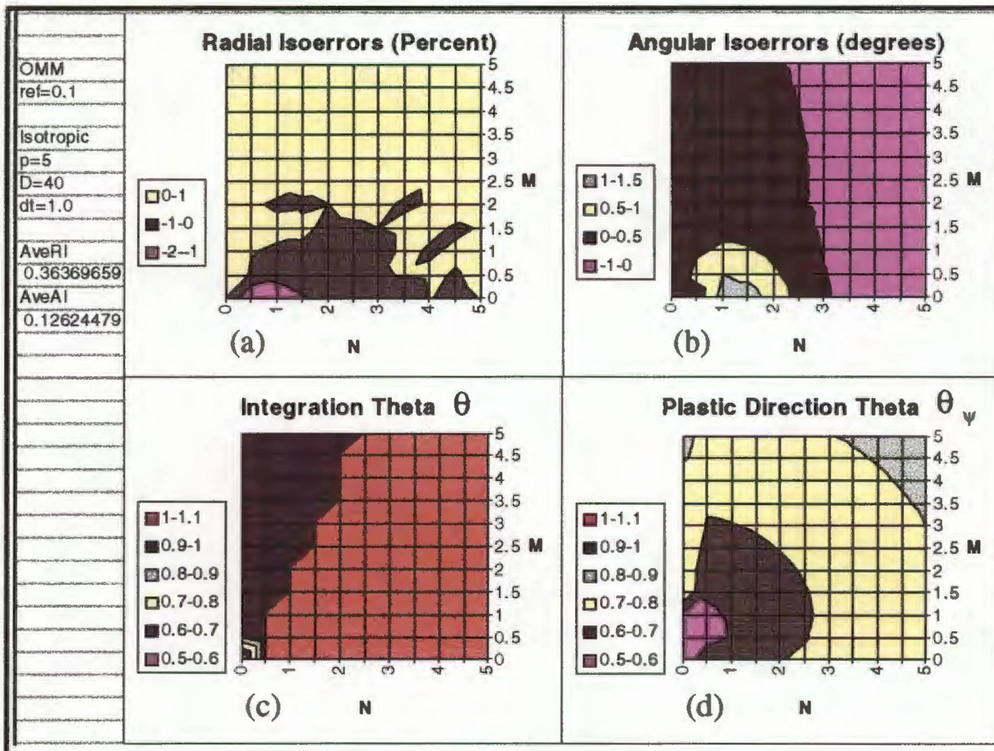

Figure A.22 Rate dependent optimised midpoint method with varying strain rates

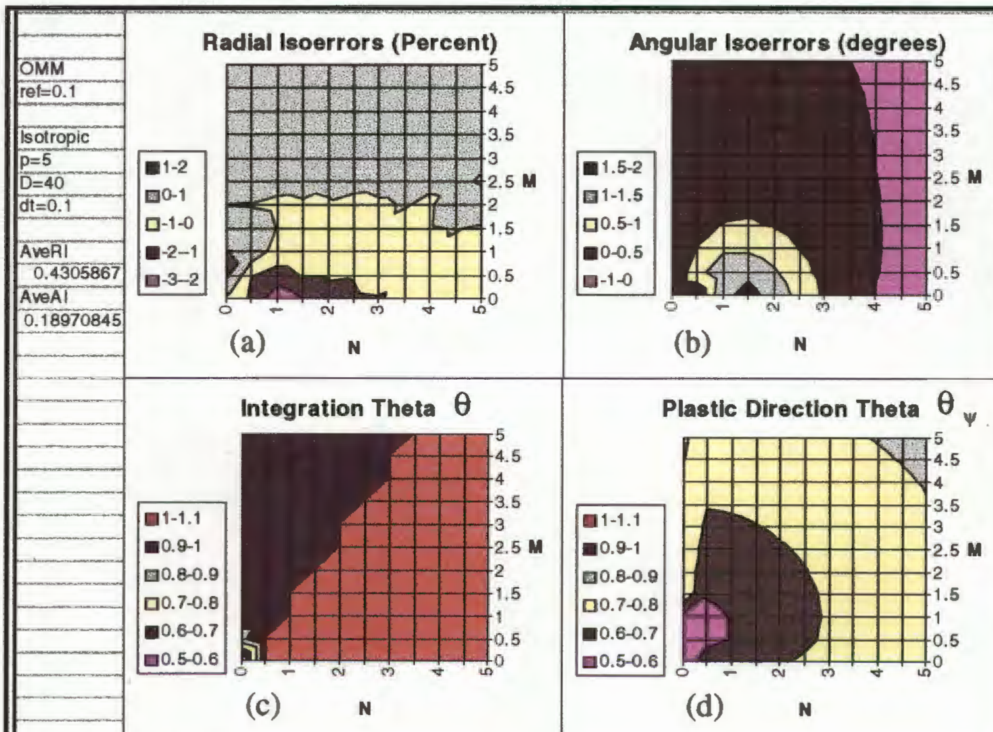

Figure A.23 Rate dependent optimised midpoint method with varying strain rates

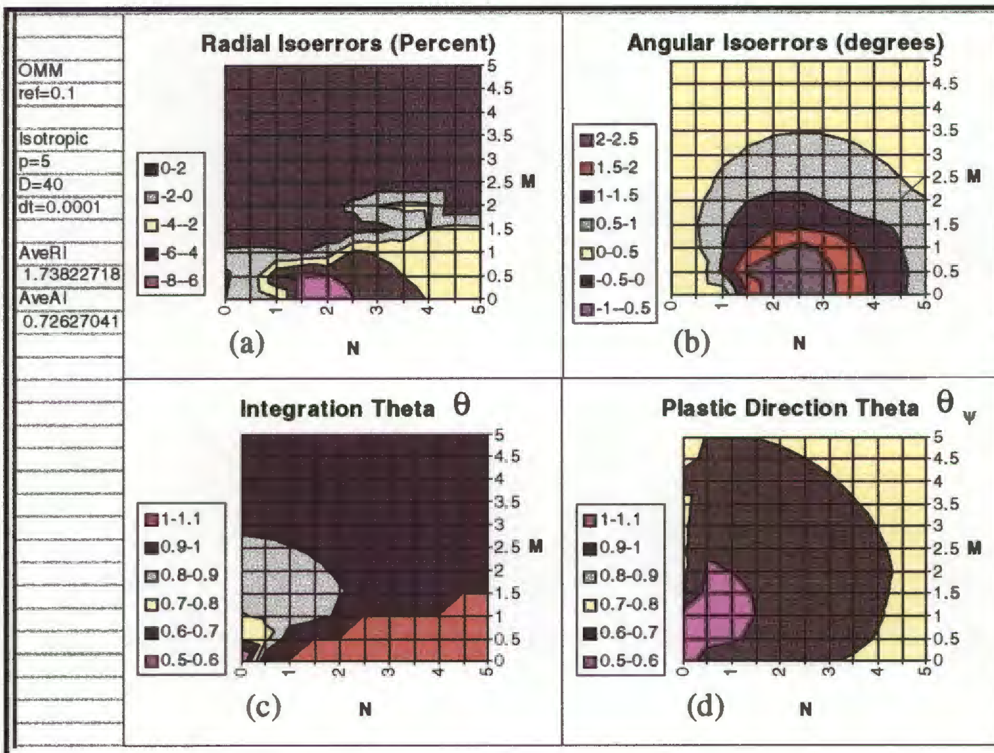

Figure A.24 Rate dependent optimised midpoint method with varying strain rates

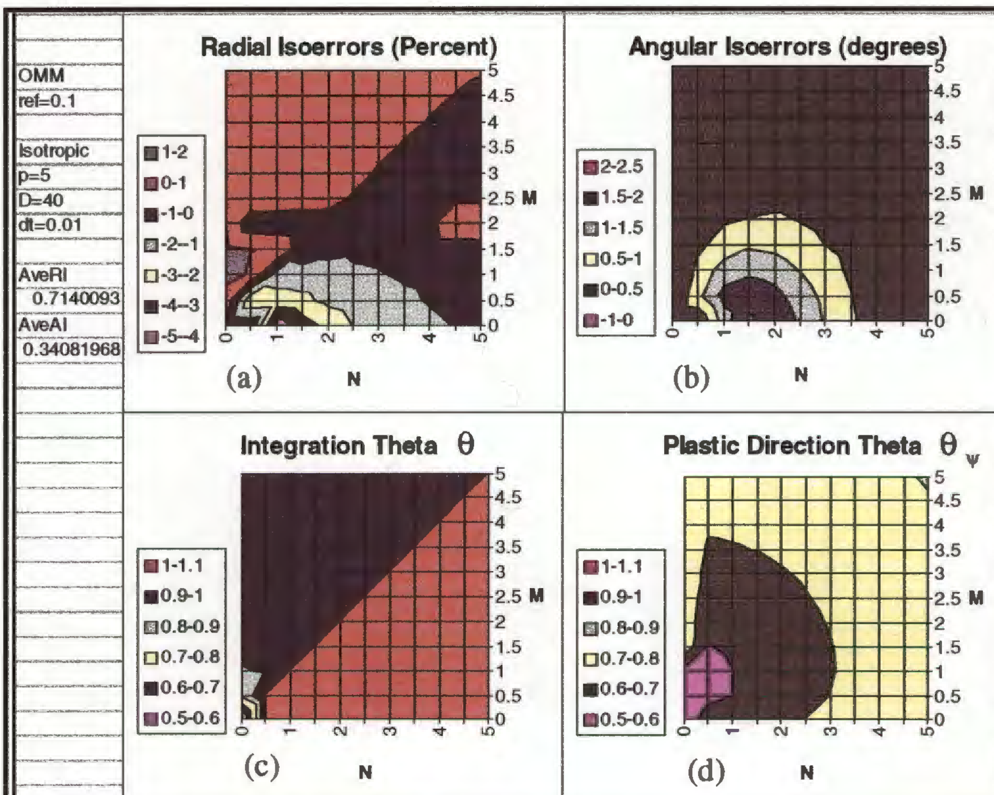

Figure A.25 Rate dependent optimised midpoint method with varying strain rates

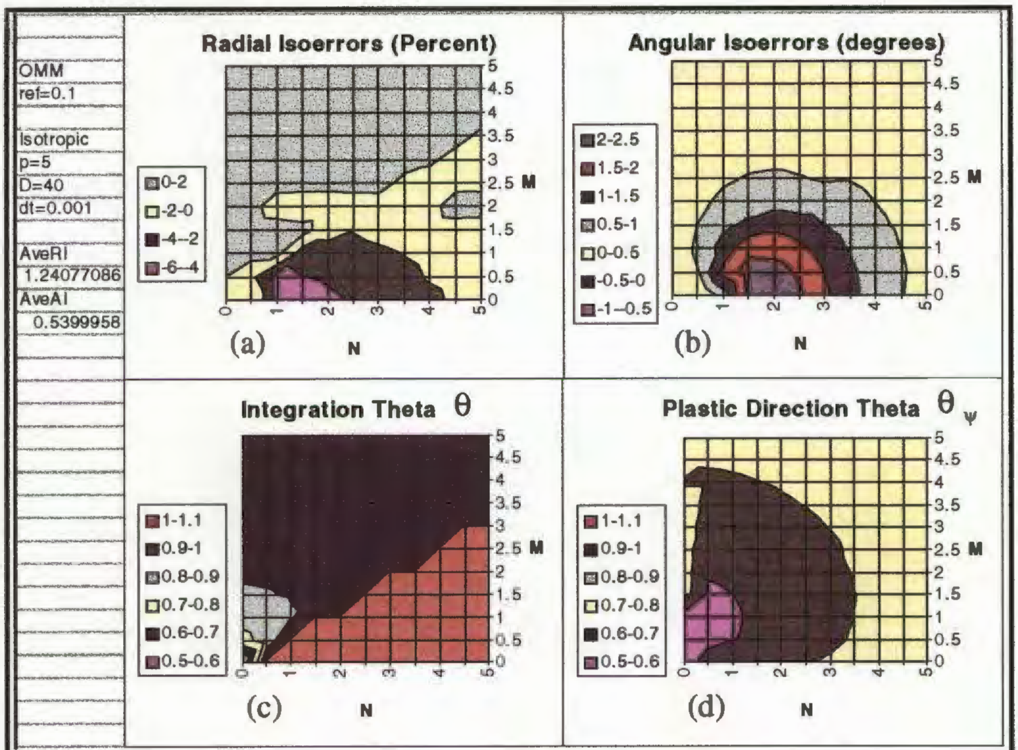

Figure A.26 Rate dependent optimised midpoint method with varying strain rates

APPENDIX B - Stress vs Strain Curves

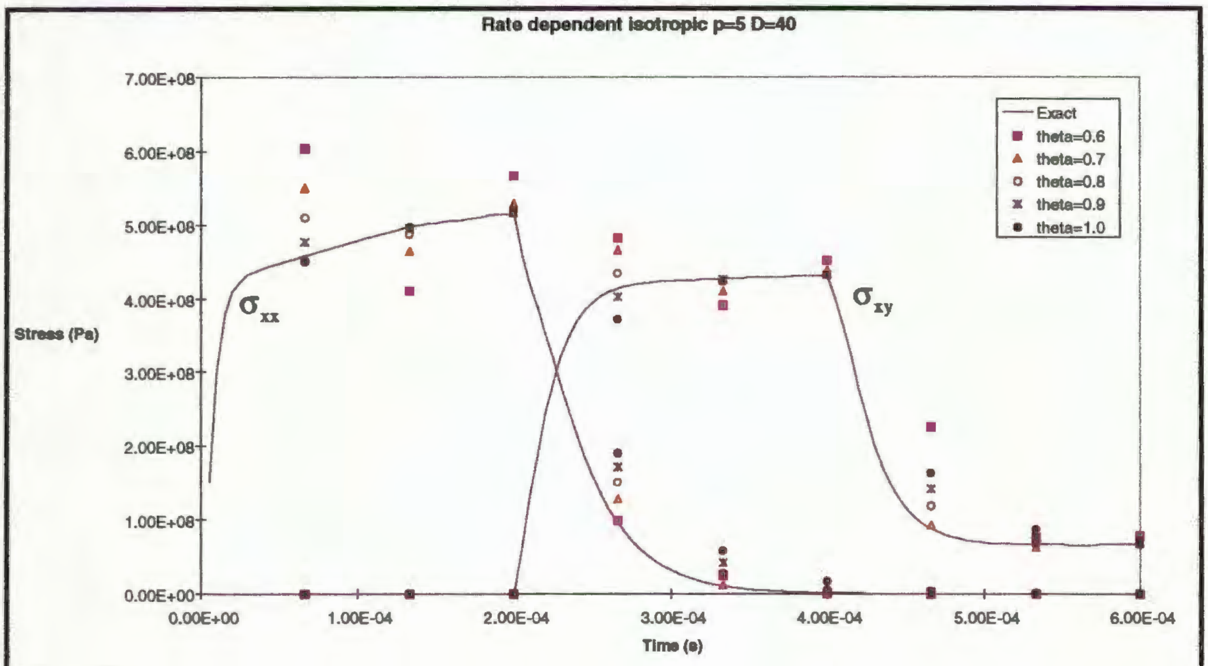

Figure B.1 Rate dependent (p=5, D=40) isotropic test demonstrating the effect of using constant θ

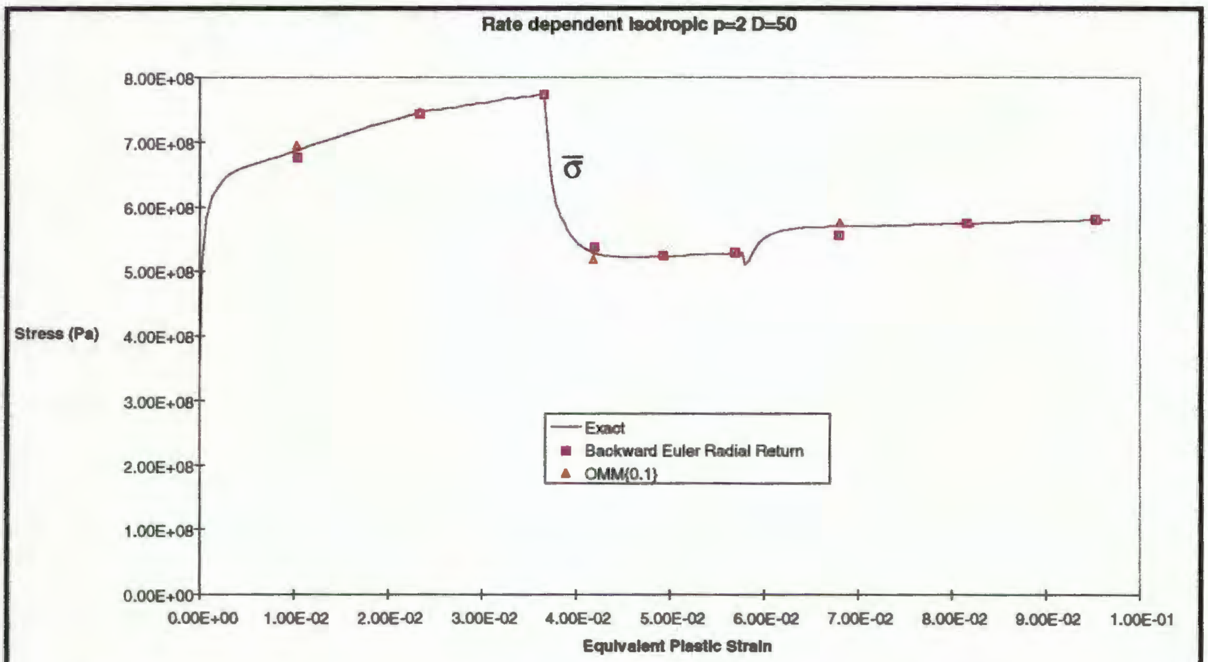

Figure B.2 Equivalent stress versus $\bar{e}^{pl}$ for rate dependent (p=2, D=50) isotropic hardening

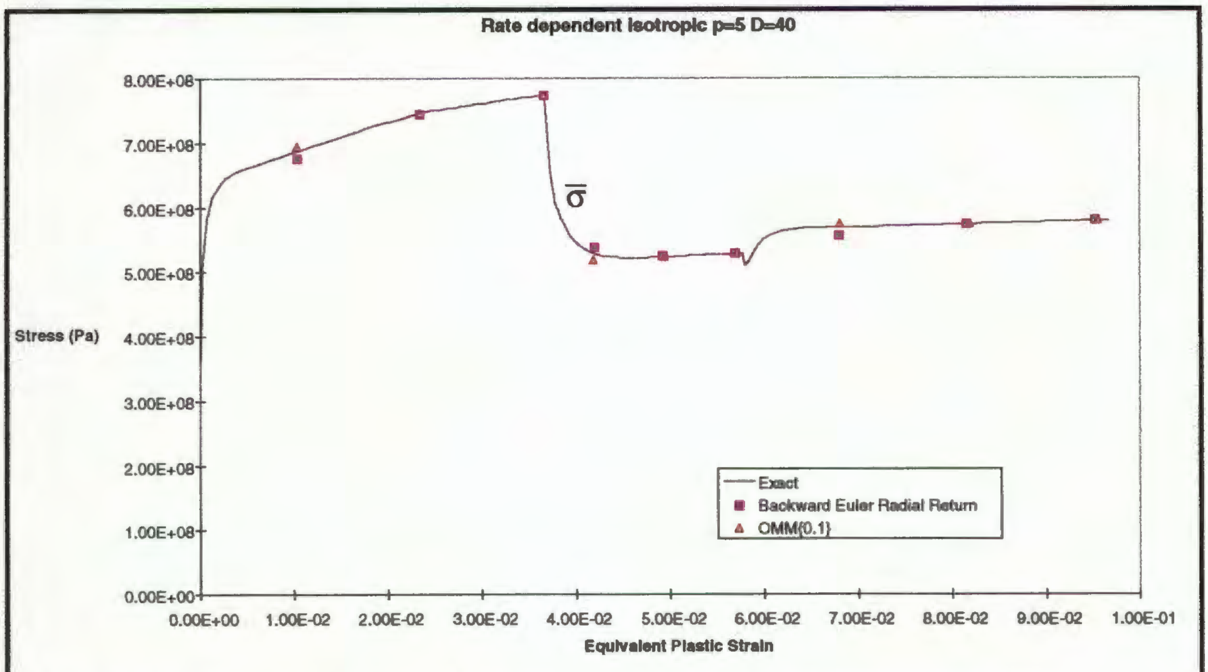

Figure B.3 Equivalent stress versus $\overline{e}^{pl}$ for rate dependent (p=5, D=40) isotropic hardening

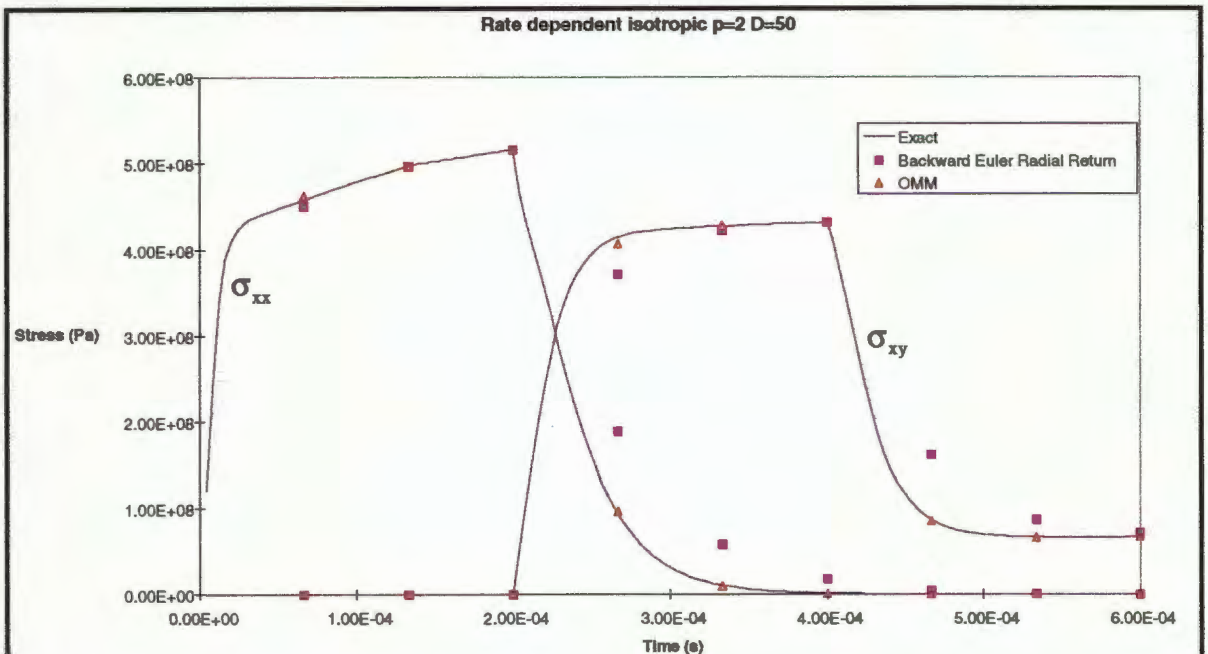

Figure B.4 Stress components versus time for rate dependent (p=2, D=50) isotropic hardening

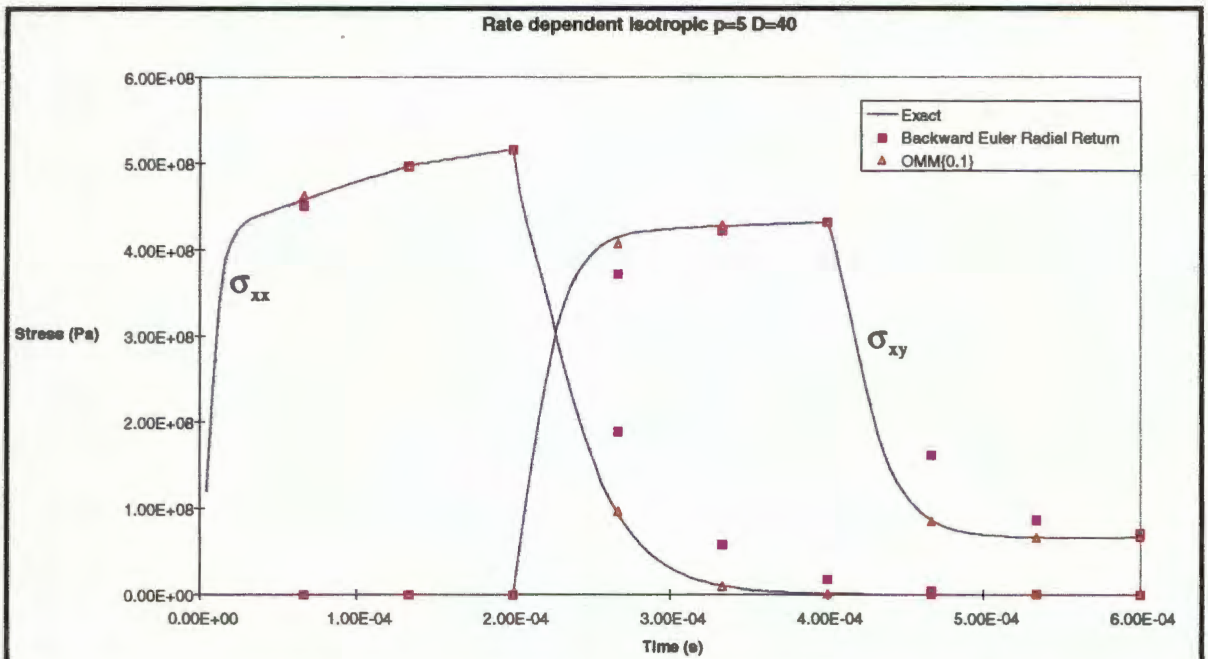

Figure B.5 Stress components versus time for rate dependent (p=5, D=40) isotropic hardening

APPENDIX C - A Summary of the Finite Element Equations

For displacement based Finite Element Analysis, the position of a point in a body can be represented by position vector {x,y,z} in Cartesian coordinates, and the displacement by a vector {u,v,w} where u, v and w are displacements in the x, y, and z directions respectively. A set of shape functions is used to approximate the displacement of a point anywhere within the body as a function of the displacement of the nodes which define the elements within the body.

If we define the vector of nodal displacements within a body having n nodes as:

$$\mathbf{a} = \begin{pmatrix} u_1 & v_1 & w_1 & \dots & u_n & v_n & w_n \end{pmatrix}^T \qquad \text{C 1}$$

and the matrix containing the shape functions as:

$$\mathbf{N} = \begin{bmatrix} N_1 & & & \dots & N_n & & \\ & N_2 & & \dots & & N_n & \\ & & N_3 & \dots & & & N_n \end{bmatrix} \qquad \text{C 2}$$

Then we can calculate the displacement of a point within the body $\mathbf{u} = (u, v, w)^T$ with:

$$\mathbf{u} \approx \mathbf{Na} \qquad \text{C 3}$$

For small strain analysis, we can define the strain at a point in the body using:

$$\varepsilon = \mathbf{Su} \qquad \text{C 4}$$

where **S** is a linear operator which transforms the displacement field to a strain field. Equation C 4 can be written out in full as:

$$\begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{xz} \end{bmatrix} = \left[\begin{bmatrix} \frac{\partial}{\partial x} & & \\ & \frac{\partial}{\partial y} & \\ & & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & \\ & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} & & \frac{\partial}{\partial x} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \right] \qquad \text{C 5}$$

Thus:

$$\varepsilon = \mathbf{Ba} \qquad \text{C 6}$$

Where:

$$\mathbf{B} = \mathbf{SN} \qquad \text{C 7}$$

The stress in the body is related to the strain in the following manner:

$$\sigma = \mathbf{D}\varepsilon \qquad \text{C 8}$$

$$\sigma = \mathbf{DBa} \qquad \text{C 9}$$

where **D** is a matrix containing the material properties of the body, termed the consistent tangent modulus.

$$\mathbf{D} = \frac{\partial\sigma}{\partial\varepsilon} \qquad \text{C 10}$$

Equilibrium of a body can be expressed through the Principle of Virtual Work, whereby an arbitrary (virtual) displacement field $\delta\mathbf{a}$ is applied to the body, allowing us to write the resulting displacement and strain fields as:

$$\delta\mathbf{u} = \mathbf{N}\delta\mathbf{a} \qquad \text{C 11}$$

$$\delta\varepsilon = \mathbf{B}\delta\mathbf{a} \qquad \text{C 12}$$

By defining a vector **r** of equivalent nodal forces which are equivalent to the boundary stresses and the distributed loads on the elements[20],as well as a vector of distributed loads **b**, we can express the work done by the load forces as:

$$W_L = \delta\mathbf{a}^T\mathbf{r} \qquad \text{C 13}$$

and the internal work per unit volume done by the stresses and distributed forces as:

$$W_i = \delta\varepsilon^T\sigma - \delta\mathbf{u}^T\mathbf{b} \qquad \text{C 14}$$

The work done per unit surface by pressure is:

$$W_s = \delta\mathbf{u}^T\bar{\mathbf{t}} \qquad \text{C 15}$$

If we equate the internal and external work on the body:

$$W_i = W_L + W_s \qquad \text{C 16}$$

then:

$$-\delta \mathbf{a}^{T}\mathbf{r} = \int_{V} \delta\mathbf{u}^{T}\mathbf{b}dV + \int_{A} \delta\mathbf{u}^{T}\bar{\mathbf{t}}\,dA - \int_{V} \delta\varepsilon^{T}\sigma\,dV \qquad \text{C 17}$$

Substituting Equation C 1, Equation C 6 and Equation C 9 into Equation C 17, and making use of the arbitrary displacements, yields:

$$\mathbf{K}\mathbf{a} + \mathbf{f} = 0 \qquad \text{C 18}$$

where:

$$\mathbf{K} = \int_{V} \mathbf{B}^{T}\mathbf{D}\mathbf{B}dV \qquad \text{C 19}$$

and **K** is known as the stiffness matrix in structural problems. The load vector **f** is defined by:

$$\mathbf{f} = -\int_{V} \mathbf{N}^{T}\mathbf{b}\,dV - \int_{A} \mathbf{N}^{T}\bar{\mathbf{t}}\,dA - \int_{V} \mathbf{B}^{T}\mathbf{D}\varepsilon_{0}\,dV + \int_{V} \mathbf{B}^{T}\sigma_{0}\,dV - \mathbf{r} \qquad \text{C 20}$$

Thus far we have dealt with the equilibrium across the entire body, but the FEM allows us to treat each element within the body separately and to sum the element contributions to satisfy Equation C 19 on a global level, i.e.:

$$\int_{V}(\ ) = \sum_{\text{Elements}} \int_{V_e}(\ ) \qquad \text{C 21}$$

We thus need only consider the problem of integration across individual elements. A common means of integrating the terms is by using Gauss Quadrature whereby an integral with arbitrary limits, can be transformed so that its limits are -1 and +1 by multiplying the function by a Jacobean. For the 3D case, the integrals transform[18]

$$\int_{Ve} \mathbf{f}(x,y,z)dV = \int_{-1}^{+1}\int_{-1}^{+1}\int_{-1}^{+1} \mathbf{J}\,\mathbf{f}(\xi,\eta,\zeta)d\xi\,d\eta\,d\zeta \qquad \text{C 22}$$

$$\int_{-1}^{+1}\int_{-1}^{+1}\int_{-1}^{+1} \mathbf{J}\,\mathbf{f}(\xi,\eta,\zeta)d\xi\,d\eta\,d\zeta = \sum_{i}\sum_{j}\sum_{k} W_{i}W_{j}W_{k}\mathbf{J}\,\mathbf{f}(\xi_{i},\eta_{j},\zeta_{k}) \qquad \text{C 23}$$

Thus the function is integrated by summing weighted sampling points of the transformed function $\mathbf{Jf}(\xi,\eta,\zeta)$. These sampling points, known as integration points, are the points at which the stresses and strains will actually be calculated, and the results extrapolated to any other parts of the body.

For non-linear problems, the solution of Equation C 18 must be broken up into several increments, such that the path of load versus displacement can be followed with reasonable accuracy. We thus split the displacement and load vectors with:

$$\begin{aligned} \mathbf{a} &= \Delta\mathbf{a}_1 + \Delta\mathbf{a}_2 + \Delta\mathbf{a}_3 + \ldots \\ \mathbf{f} &= \Delta\mathbf{f}_1 + \Delta\mathbf{f}_2 + \Delta\mathbf{f}_3 + \ldots \end{aligned} \qquad \text{C 24}$$

Thus:

$$\mathbf{K}\mathbf{a}_n + \mathbf{f}_n = 0 \qquad \text{C 25}$$

Since we do not expect to obtain an exact solution to Equation C 25, we can use an iterative scheme to solve for the nodal displacements. One such scheme is the Newton-Raphson scheme. We define a residual $\psi_{(n+1)}$ such that:

$$\psi_{(n+1)} = \mathbf{P}_{(n+1)} + \mathbf{f}_{(n+1)} \qquad \text{C 26}$$

where:

$$\mathbf{P} = \int_V \mathbf{B}^T \sigma \mathrm{d}V \qquad \text{C 27}$$

and is a vector of internal forces.

By splitting each displacement increment into successive approximations:

$$\Delta\mathbf{a}_n = \sum_{i=1}^{k} \delta\mathbf{a}_n^i \qquad \text{C 28}$$

we get:

$$\psi_{n+1}^{i+1} \approx \psi_{n+1}^{i} + \left(\frac{\partial \psi}{\partial \mathbf{a}}\right)_{n+1}^{i} \delta\mathbf{a}_n^i \qquad \text{C 29}$$

where:

$$\frac{\partial \psi}{\partial \mathbf{a}} = \frac{\partial \mathbf{P}}{\partial \mathbf{a}} = \frac{\partial}{\partial \mathbf{a}}\left(\int_V \mathbf{B}^T \sigma \mathrm{d}V\right) = \int_V \mathbf{B}^T \frac{\partial \sigma}{\partial \varepsilon} \mathrm{B}\mathrm{d}V = \int_V \mathbf{B}^T \mathbf{D}\mathbf{B} \mathrm{d}V = \mathrm{K} \qquad \text{C 30}$$

Thus **K** corresponds to the tangent direction of the load-displacement curve.

Equation C 29 is repeatedly applied until the residual is sufficiently small. Equation C 28 can then be applied to determine the total displacement vector at the end of the increment, before the next increment is applied.

We use the following basic solution procedure in displacement based FEM:

1. We solve Equation C 18 in order to solve for the nodal displacements **a**. This is only possible once the consistent tangent modulus **D** relating the stresses and strains by Equation C 8 is known, allowing Equation C 19 to be put entirely in terms of nodal variables and boundary conditions, with **D** taking care of material considerations.
2. Once **a** is known, the next step is to calculate the resulting strain field and the state of stress σ which results from these.

Since the tensor **D** is used extensively, the global solution scheme can be speeded up significantly were **D** to be symmetric too. This condition is a major consideration during the formulation of a material model.

APPENDIX D - FORTRAN Code Listings

In order to have a basis from which to execute rapid tests on, amongst others, the Backward Euler Radial Return Method and the Optimised Midpoint Method with Variable Return, a subincrementation program (TEST) was written. This controlled the basic input parameters into the subroutines governing the material update procedures, and essentially simulated the input parameters which would be received from the well known FEM package ABAQUS. The subroutines governing the two methods were UMAT1 (Backward Euler) and UMAT2 (Optimised Midpoint). UMAT2 also has input parameters which allow it to behave simply as a Generalised Midpoint Algorithm (GMM), i.e. an input parameter (described in the code itself) which disables the dynamic choosing of θ and allows one to specify a constant instead.

The subroutine which TEST calls (UMAT) is passed a parameter which determines which of the two material update procedures is to be used, and controls the external output for data capture. The code is written such that UMAT can be directly implemented into ABAQUS, while TEST enables stand alone testing of the two methods.

Both UMAT1 and UMAT2 receive as input the initial values of stress, strain and material constants. Wherever possible, vector quantities are passed in, with the strain being engineering strain.

The routines perform the full stress update, and calculate a consistent tangent modulus. The tangent modulus is of little interest in stand alone testing, but is vital for implementation into FEM code.

```
c***********************************************
c file INC.FOR
c***********************************************

c***********************************************
      program TEST
c-----------------------------------------------
c Test program for user material
c Uses subincrementation outputs results to file
c-----------------------------------------------

      doubleprecision stress,stran,dstran,ddsdde,statev,props,
     + sse,spd,scd,rpl,ddsddt,drplde,drpldt,pnewdt,time,dtime,
     + temp,dtemp,predef,coords,dfgrd1,dfgrd0,celent,drot,dpred

      doubleprecision tote,tott,tan,rad
      integer n,i,step,steps

      integer ntens,nstatv,nprops,kinc,kstep,kspt,ndi,npt,noel,nshr,
```

```
  + layer

    character*8 cmname

    parameter (ntens=6,nstatv=7,nprops=35,steps=2)

    dimension stress(ntens),stran(ntens),dstran(ntens),
  + ddsdde(ntens,ntens),statev(nstatv),props(nprops),
  + ddsddt(ntens),drplde(ntens),time(2),predef(1),dpred(1),
  + coords(3),drot(3,3),dfgrd0(3,3),dfgrd1(3,3)

    dimension tote(ntens,steps),tott(steps)

    data n/1/
    data props/2.0d0,197.0d9,0.3d0,1.0d0,100.0d0,0.0d0,0.0d0,
  +        256d6,0.000,
  +        275d6,0.003,
  +        285d6,0.008,
  +        295d6,0.013,
  +        305d6,0.018,
  +        315d6,0.024,
  +        325d6,0.035,
  +        335d6,0.056,
  +        345d6,0.1,
  +        355d6,0.137,
  +        365d6,0.171,
  +        375d6,0.2,
  +        445d6,0.403,
  +        600d6,10.0/

c-loop over tangential and radial increment
    do 100 rad=0.0, 2.0, 0.5
      do 200 tan=0.0, 2.0, 0.5
        print *,'********************',tan,rad

    call MZERO (statev,7,1)
    call MZERO (stress,6,1)
    call MZERO (stran,6,1)
    call MZERO (dstran,6,1)

    stress(1)=0.0d0
    stress(2)=0.0d0
    stress(3)=0.0d0
    stress(4)=0.0d0
    stress(5)=0.0d0
    stress(6)=0.0d0

    stran(1)=0.0d0
    stran(2)=0.0d0
```

```
      stran(3)=0.0d0
      stran(4)=0.0d0
      stran(5)=0.0d0
      stran(6)=0.0d0

c-this marks the total strain increment desired
      tote(1,1)=0.0D0
      tote(2,1)=0.0d0
      tote(3,1)=0.0d0
      tote(4,1)=1.951D-3
      tote(5,1)=0.0D0
      tote(6,1)=0.0D0

      tote(1,2)=0.0d0
      tote(2,2)=0.0d0
      tote(3,2)=0.0d0
      tote(4,2)=1.951D-3+1.951d-3*rad
      tote(5,2)=1.951d-3*tan
      tote(6,2)=0.0d0

c-set this to the total time which the analysis is to take
      tott(1)=0.01d0
      tott(2)=0.01d0
      tott(3)=0.01d0
      tott(4)=0.01d0
      tott(5)=0.01d0

      kstep=1
      kinc=1
      noel=1
      npt=1
      time=0.0d0

      statev(1)=0.0d0

c-start subincrementation, set n=number of subincrements per step
      do 20 step=1,steps
        print *,'Here starts step:',step

        call MSUB (tote(1,step),1,6,stran)
        call MEQU (tote(1,step),dstran,ntens,1)
        call MCONST (dstran,ntens,1,1.0d0/dble(n))
        do 10 i=1,n

        dtime=tott(step)/dble(n)

c-update the stress and calculate tangent matrix
        call UMAT (stress,statev,ddsdde,sse,spd,scd,rpl,ddsddt,
     +  drplde,drpldt,stran,dstran,time,dtime,temp,dtemp,predef,
```

```
   +  dpred,cmname,ndi,nshr,ntens,nstatv,props,nprops,coords,drot,
   +  pnewdt,celent,dfgrd0,dfgrd1,noel,npt,layer,kspt,kstep,kinc)

         call MADD (stran,ntens,1,dstran)

c-update the increment number
         kinc=kinc+1
         time=time+tott(step)/dble(n)

10     continue
       kstep=step+1
       kinc=1

20   continue

c-complete loop over tangential and radial components
200     continue
100   continue


     end
c**********************************************

c**********************************************
c  file UMAT.FOR
c----------------------------------------------
c  Programmed for MSc(eng)
c  by Derek Hulley
c
c  Description of subroutine requirements:
c    Property 1: Type of element: 1=ABAQUS rate dependent 3D
c                                 2=Modified midpoint method
c    Property 2: Young's Modulus
c    Property 3: Poisson's ratio
c    Property 4: p value for power law
c    Property 5: D value for power law
c    Property 6: Reference strain rate (for UMAT2 only).  Set to zero to default
c                to backward Euler integration.  Set to 100*theta to choose theta
c                as a constant.
c    Property 7: Bauschinger effect modelled 0=NO (for UMAT2 only)
c    Properties 8,9-: Yield stress1, Equivalent plastic strain1, Yield stress2, Equ.....
c
c**********************************************

c**********************************************
     subroutine UMAT(stress,statev,ddsdde,sse,spd,scd,rpl,ddsddt,
   +  drplde,drpldt,stran,dstran,time,dtime,temp,dtemp,predef,
   +  dpred,cmname,ndi,nshr,ntens,nstatv,props,nprops,coords,drot,
   +  pnewdt,celent,dfgrd0,dfgrd1,noel,npt,layer,kspt,kstep,kinc)
```

```
c-----------------------------------------------

      doubleprecision stress,stran,dstran,ddsdde,statev,props,
     +  sse,spd,scd,rpl,ddsddt,drplde,drpldt,pnewdt,time,dtime,
     +  temp,dtemp,predef,coords,dfgrd1,dfgrd0,celent,drot,dpred

      integer ntens,nstatv,nprops,kinc,kstep,kspt,ndi,npt,noel,nshr,
     +  layer
      character*8 cmname

      dimension stress(ntens),stran(ntens),dstran(ntens),
     +  ddsdde(ntens,ntens),statev(nstatv),props(nprops),
     +  ddsddt(ntens),drplde(ntens),time(2),predef(1),dpred(1),
     +  coords(3),drot(3,3),dfgrd0(3,3),dfgrd1(3,3)

      integer count,lunout,prnlst,stsstn

      character*80 filout

      doubleprecision temp2,totstn,xcoord,ycoord
      doubleprecision theta,thetap,psa,psb,gamma,qb

      dimension prnlst(9),stsstn(2,10),temp2(2,10),totstn(15)

c-prnlst (print list) is (element no*10+integration pt number) to print
c-stsstn (stress-strain) is the list of curves to generate (max 10):
c-ie. 1 1 4 1 will generate S11 vs E11, and S12 vs E11
c-1==S11,E11; 2==S22,E22; 3==S33,E33; 4==S12,E12; etc up to S13,E13
c-The use of 7 in the stress part of prnlst will generate the list of
c-equivalent plastic strain values after the step.
c-The use of 8 in the stress part of prnlst will generate the list of
c-total equivalent stress vs whatever strain value chosen
c-Use 9 in the stress part will generate the PI plane projection
c-Use 10 in the stress part to get theta vs some strain/time
c-Use 11 in the stress part to get thetap vs some strain/time
c-Use 12 in the stress part to get psa vs some strain/time
c-Use 13 in the stress part to get psb vs some strain/time
c-Use 14 in the stress part to get qb vs some strain/time
c-Use 15 in the stress part to get gamma vs some strain/time
c-Totstn 7-13 are the 7 state variables
c-Totstn 14 is the total equivalent strain
c-Totstn 15 is the total time
      data prnlst/11,0,0,0,0,0,0,0,0/
      data stsstn/12,15,13,15,14,15,10,15,11,15,0,0,
     +        0,0,0,0,0,0,0,0/

      lunout=0
      call MEQU (stran,totstn,1,ntens)
```

```
c-open the output file to write to
c-set filout to not more than 5 letters
c-The file naming convention is to put the directory letter first,
c-the name of the input deck follows, less the last number which should
c-indicate the umat number to use, up to a maximum of 5 letters.
c-The umat number is appended first, followed by _and the int.pt.no
c-given by prnlst
      filout="d:\work\thesis\tanrad\tr"
      count=0
      do 20 i=1,80
        if (filout(i:i).ne." ") then
          count=count+1
        end if
20    continue

      do 10 i=1,9
        if ((noel*10+npt).eq.prnlst(i)) then
          lunout=60+prnlst(i)
          filout(count+3:count+3)=char(i+48)
          filout(count+4:count+7)='.dat'
        end if
10    continue

      if ((count.eq.0).and.(lunout.ne.0)) then
        print *,'You need to specify an output file'
        stop
      end if

c-no output
c     lunout=0

      if (lunout.ne.0) then
        if (props(1).eq.1.0d0) then
          filout(count+1:count+2)='1_'
        else if (props(1).eq.2.0d0) then
          filout(count+1:count+2)='2_'
        else
          print *,'The umat number requested has not been made.'
          stop
        end if

        open (unit=lunout,file=filout,status='unknown',
     +       access='sequential',form='formatted')

      end if

      call MADD (totstn,1,ntens,dstran)
      call MINNER (totstn,totstn,1,ntens,totstn(14))
      totstn(14)=sqrt(2.0d0/3.0d0*totstn(14))
```

```
      totstn(15)=dtime+time(2)

      if (props(1).eq.1.0d0) then
        call UMAT1(stress,statev,ddsdde,sse,spd,scd,rpl,ddsddt,
     +  drplde,drpldt,stran,dstran,time,dtime,temp,dtemp,predef,
     +  dpred,cmname,ndi,nshr,ntens,nstatv,props,nprops,coords,drot,
     +  pnewdt,celent,dfgrd0,dfgrd1,noel,npt,layer,kspt,kstep,kinc,
c-some output variables
     +  psa,psb,qb)
      end if

      if (props(1).eq.2.0d0) then
        call UMAT2(stress,statev,ddsdde,sse,spd,scd,rpl,ddsddt,
     +  drplde,drpldt,stran,dstran,time,dtime,temp,dtemp,predef,
     +  dpred,cmname,ndi,nshr,ntens,nstatv,props,nprops,coords,drot,
     +  pnewdt,celent,dfgrd0,dfgrd1,noel,npt,layer,kspt,kstep,kinc,
c-some output variables
     +  theta,thetap,psa,psb,qb,gamma)
      end if

c-update total stress
      call MEQU (statev,totstn(7),1,nstatv)

      if (lunout.ne.0) then
        count=0
        do 30 i=1,10
          if ((stsstn(1,i).ne.0).and.(stsstn(1,i).lt.7)) then
            temp2(1,i)=stress(stsstn(1,i))
            temp2(2,i)=totstn(stsstn(2,i))
          else if (stsstn(1,i).eq.7) then
            temp2(1,i)=statev(1)
            temp2(2,i)=totstn(stsstn(2,i))
          else if (stsstn(1,i).eq.8) then
            call MINNER (stress,stress,1,6,temp2(1,i))
            temp2(1,i)=sqrt(3.0d0/2.0d0*temp2(1,i))
            temp2(2,i)=totstn(stsstn(2,i))
          else if (stsstn(1,i).eq.9) then
            call DEVSPC (xcoord,ycoord,stress)
            temp2(1,i)=ycoord
            temp2(2,i)=xcoord
          else if ((stsstn(1,i).eq.10).and.(props(1).eq.2.0d0))
     +       then
            temp2(1,i)=theta
            temp2(2,i)=totstn(stsstn(2,i))
          else if ((stsstn(1,i).eq.11).and.(props(1).eq.2.0d0))
     +       then
            temp2(1,i)=thetap
            temp2(2,i)=totstn(stsstn(2,i))
          else if (stsstn(1,i).eq.12)
```

```
   +      then
          temp2(1,i)=psa
          temp2(2,i)=totstn(stsstn(2,i))
        else if (stsstn(1,i).eq.13) then
          temp2(1,i)=psb
          temp2(2,i)=totstn(stsstn(2,i))
        else if (stsstn(1,i).eq.14) then
          temp2(1,i)=qb
          temp2(2,i)=totstn(stsstn(2,i))
        else if ((stsstn(1,i).eq.15).and.(props(1).eq.2.0d0))
   +      then
          temp2(1,i)=gamma
          temp2(2,i)=totstn(stsstn(2,i))
        end if
30    continue

      write (lunout,1000) kstep,kinc,temp2(2,1),
   +    temp2(1,1),temp2(2,2),temp2(1,2),temp2(2,3),
   +    temp2(1,3),temp2(2,4),temp2(1,4),temp2(2,5),
   +    temp2(1,5),temp2(2,6),temp2(1,6),temp2(2,7),
   +    temp2(1,7),temp2(2,8),temp2(1,8),temp2(2,9),
   +    temp2(1,9),temp2(2,10),temp2(1,10)
1000  format (2(i5),20(E15.5E2))

    end if

    return
    end
c*****************************************************

c*******************************************
c file UMAT2.FOR
c---------------------------------------------
c Programmed for MSc(eng)
c by Derek Hulley
c*******************************************

c*****************************************************
    subroutine UMAT2(stress,statev,ddsdde,sse,spd,scd,rpl,ddsddt,
   + drplde,drpldt,stran,dstran,time,dtime,temp,dtemp,predef,
   + dpred,cmname,ndi,nshr,ntens,nstatv,props,nprops,coords,drot,
   + pnewdt,celent,dfgrd0,dfgrd1,noel,npt,layer,kspt,kstep,kinc,
c-some output variables
   + theta,thetap,psa,psb,qb,gamma)
c-------------------------------------------------------
c   Type of Umat selected was 1: Modified midpoint method
c
c   Number of state variables = 7
c-------------------------------------------------------
```

```
      doubleprecision stress,stran,dstran,ddsdde,statev,props,
     +  sse,spd,scd,rpl,ddsddt,drplde,drpldt,pnewdt,time,dtime,
     +  temp,dtemp,predef,coords,dfgrd1,dfgrd0,celent,drot,dpred

      integer ntens,nstatv,nprops,kinc,kstep,kspt,ndi,npt,noel,nshr,
     +  layer

      character*8 cmname

      dimension stress(ntens),stran(ntens),dstran(ntens),
     +  ddsdde(ntens,ntens),statev(nstatv),props(nprops),
     +  ddsddt(ntens),drplde(ntens),time(2),predef(1),dpred(1),
     +  coords(3),drot(3,3),dfgrd0(3,3),dfgrd1(3,3)

      doubleprecision E,mu,H,Hp,p,D,G,Kbulk,equple,volsts,volstn,
     +  sija,deij,sijn,arb1,const1,const2,q,D2,deqple,
     +  f,fprime,Dstar,mua,deque,gamma,yt,cospsa,psa,zeta,Hstar,
     +  theta,trimm,thetap,eta,H1,H2,qb,thetab,dD2,iter1,
     +  muth,mub,mup,Bija,lambda,r1,r2,alpha,Dsij,cutlim,gamma2,
     +  m,n,sijb,pi,psb,qbar,Gbar,Hstbar,c,epslon,taub,xstarb,
     +  omega,Xxt,Xinv,Xce,dX,kstar,eqplst

      integer index,i,j,k,l,x,y,count,nvalue,cut,prn
      logical modnr,yield

      dimension index(6,2),sija(3,3),deij(3,3),sijn(3,3),Dsij(3,3),
     +  arb1(3,3),Dstar(7,7),mua(3,3),Bija(3,3),eta(3,3),mub(3,3),
     +  muth(3,3),mup(3,3),prn(20),sijb(3,3)

      data index/1,2,3,1,2,1,1,2,3,2,3,3/,pi/3.14159265359d0/

c-set modnr to .true. to use modified Newton Raphson
c-set cutlim to the percentage that the predictor must
c-overstep the yield surface to induce plasticity.
      data modnr/.false./,cutlim/0.001/
      data prn/0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0/
c     data prn/1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1/

      if (prn(1).eq.1) then
        print *,'===>',kstep,kinc,dtime,' umat2'
        call MPRINT (stress,1,6)
        call MPRINT (stran,1,6)
        call MPRINT (dstran,1,6)
        call MPRINT (statev,1,7)
        call MPRINT (props,1,nprops)
      end if

      E = props(2)
```

```
      mu = props(3)
      p = props (4)
      D = props (5)
      refde=props (6)
      deqple = 0.0d0
      count=0
      yield=.false.
      cut=0
      G = E/(2.0d0*(1.0d0+mu))
      Kbulk = E/3.0d0/(1.0d0-2.0d0*mu)
      nvalue=(nprops-7)/2

c-check to see if this is the first increment
c-set the total Equation plastic strain to zero if it is
      if ((kstep.eq.1).and.(kinc.eq.1)) then
        do 10 i=1,nstatv
          statev(i)=0.0d0
10      continue
      end if
      equple=statev(1)
      do 15 i=2,7
        Bija(index(i-1,1),index(i-1,2))=statev(i)
        Bija(index(i-1,2),index(i-1,1))=statev(i)
15    continue

c-initialize matrices
      call MZERO (sijn,3,3)
      call MZERO (deij,3,3)
      call MZERO (Dstar,7,7)

c-convert stresses and strains to matrix form
      do 20 i=1,ntens
        sija(index(i,1),index(i,2))=stress(i)
        sija(index(i,2),index(i,1))=stress(i)
        if (index(i,1).ne.index(i,2)) then
          deij(index(i,1),index(i,2))=0.5d0*dstran(i)
        else
          deij(index(i,1),index(i,2))=dstran(i)
        end if
        deij(index(i,2),index(i,1))=deij(index(i,1),index(i,2))
20    continue

c-calculate volumetric stresses and strains
      volstn=(deij(1,1)+deij(2,2)+deij(3,3))
      volsts=(sija(1,1)+sija(2,2)+sija(2,2))/3.0d0

c-calculate deviatoric stresses at beginning of increment
c-and deviatoric strain increment
      do 30 i=1,3
```

```
      sija(i,i)=sija(i,i)-volsts
      deij(i,i)=deij(i,i)-volstn/3.0d0
30   continue

c-take out back stress portion of stress
    call MSUB (sija,3,3,Bija)

c-calculate elastic predictor stress
    const1=2.0d0*G
    call MEQU (deij,arb1,3,3)
    call MCONST (arb1,3,3,const1)
    call MADD (arb1,3,3,sija)
    call MEQU (arb1,sijn,3,3)

c-set final stress to elastic predictor stress
    call MEQU (sijn,sijb,3,3)

c-print out elastic predictor stress,etc
    if (prn(2).eq.1) print *,volsts,volstn
    if (prn(3).eq.1) call MPRINT (sijn,3,3)

c-calculate equivalent strain increment and jump to elastic tangent
c-matrix calculation if zero
    call MINNER (deij,deij,3,3,const1)
    deque=sqrt(const1*2.0d0/3.0d0)
    if (deque.eq.0.0d0) goto 3000

c-check the elastic predictor stress to determine if
c-plasticity has occured.  The predictor stress, must
c-be more outside the yield limit by more than cutlim*H of the
c-yield value
    call MEQU (sijn,Dsij,3,3)
    call MSUB (Dsij,3,3,sija)
    call UHARD(H,Hp,equple,props(8),nvalue)
    D2=1.0d0
    if (q(sijn).gt.(D2*H+cutlim*H)) then
      yield=.true.

c-check for partially elastic strain increment
c-and scale the strain increment and time step
      if (q(sija).lt.(H-cutlim*H)) then
160        continue
        call MEQU (sijn,Dsij,3,3)
        call MSUB (Dsij,3,3,sija)
        cut=cut+1
        r1=H-q(sija)
        r2=q(sijn)-q(sija)
        alpha=r1/r2
        call MCONST (Dsij,3,3,alpha)
```

```
        call MADD (sija,3,3,Dsij)
        if (q(sija).lt.(H-cutlim*H)) then
          goto 160
        end if
        call MCONST (deij,3,3,(1.0d0-alpha))
        dtime=dtime*(1-alpha)
        volsts=volsts+Kbulk*(alpha*volstn)
        volstn=volstn*(1-alpha)

        if (prn(4).eq.1) print *,H,r1,r2
        if (prn(4).eq.1) print *,alpha,q(sija)
        if (prn(4).eq.1) call MPRINT (deij,3,3)

      end if
    end if

c-check that the actual strain icrement to use is non-zero
    call MINNER (deij,deij,3,3,const1)
    deque=sqrt(const1*2.0d0/3.0d0)
    if (deque.eq.0.0d0) goto 3000

    if (prn(5).eq.1) print *,yield,cut,deque
    if (prn(5).eq.1) call MPRINT (sija,3,3)

c-perform update for plastic increment
    if (yield) then

c-calculate the strain and initial stress directions
      call MEQU(deij,eta,3,3)
      call MCONST(eta,3,3,1.0d0/deque)

      const1=q(sija)
      call MEQU(sija,mua,3,3)
      call MCONST(mua,3,3,3.0d0/2.0d0/const1)

      call MINNER (mua,eta,3,3,const1)
      psa=acos(min(1.0d0,max(-1.0d0,2.0d0/3.0d0*const1)))
      cospsa=cos(psa)

c-calculate constants
      gamma=3.0d0*G*deque/q(sija)

c-check for numerical noise in calculation of cospsa
      if (cospsa.eq.0.0d0) cospsa=0.0001d0

      zeta=(yt(1.0d0,gamma,cospsa)-1.0d0)/gamma

      if (prn(6).eq.1) print *,gamma,cospsa
      if (prn(7).eq.1) print *,psa,zeta
```

```
c-provide a non-zero guess for deqple
      deqple=abs(zeta)*deque

      if ((refde.lt.10.0d0).and.(refde.gt.0.0d0)) then
c
c======= calculate theta (P.A.Fotiu) =====================
200     continue

        eqplst=equple+zeta*deque/2.0d0
        call UHARD (kstar,Hp,eqplst,props(8),nvalue)

        if (props(7).eq.0.0d0) then
          Hstbar=0.0d0
        else
          call UHARD (H1,const1,equple+zeta*deque,props(8),
     +              nvalue)
          call UHARD (H2,const2,equple,props(8),nvalue)
          Hstbar=3.0d0/2.0d0*(H1-H2)/(deque*zeta)
        end if

        if (prn(8).eq.1) print *,'####',eqplst,kstar

        Gbar=G/kstar
        qbar=q(sija)*cospsa/kstar

        if (prn(9).eq.1) print *,Gbar,qbar,Hstbar

        taub=refde*dtime
        epslon=deque/taub
        c=3.0d0*Gbar/(3.0d0*Gbar+Hstbar)

        if (prn(10).eq.1) print *,taub,epslon,c

        Xinv=abs(D*(qbar-1.0d0)**p)
        if (Xinv.eq.0.0d0) Xinv=1.0d-8

        call D2f (c*epslon,1.0d0,D,p,Xce)
        dX=(Xinv**((1.0d0-p)/p))/(p*D**(1.0d0/p))

        if (prn(11).eq.1) print *,Xinv,Xce,dX

        kappa=(3.0d0*Gbar+Hstbar)*(c*epslon-Xinv)/(Xce-qbar)

        if (abs(kappa*taub).gt.100.0d0) then
          print *,'Reference strain rate too large'
          refde=refde/10.0d0
          print *,'Lowering refde to:',refde
          goto 200
```

```
          end if

          xstarb=c*epslon*taub-(Xce-qbar)/(3.0d0*Gbar+Hstbar)*
     +          (1-exp(-1.0d0*kappa*taub))

          call D2f (xstarb/taub,1.0d0,D,p,Xxt)

          if (prn(12).eq.1) print *,kappa,xstarb,XXt

          omega=dX*(c*epslon-Xinv)/(Xce-qbar)

          theta=(Xxt-qbar)/(Xce-qbar)/
     +         (1.0d0-exp(-1.0d0*omega*kappa*taub))


c-use the backward Euler scheme
        else if (refde.eq.0.0d0) then
          theta=1.0d0
        else
          theta=refde/100.0d0
        end if

        theta=TRIMM(theta)
cc
c-Newton method iteration to find plastic
c-strain increment
1000    if (count.eq.20) then
          print *,'Not converging'
          deqple=(iter1+deqple)/2.0d0
          goto 2000
        end if

        iter1=deqple

c-update the inverse of the overstress factor and derivative
        call D2f(deqple,dtime,D,p,D2)
        call DD2f(deqple,dtime,D,p,dD2)

c-update Hstar and HpB
        if (props(7).eq.0.0d0) then
          Hstar=0.0d0
          HpB=0.0d0
        else
          call UHARD (H1,const1,equple+theta*deqple,props(8),nvalue)
          call UHARD (H2,const2,equple,props(8),nvalue)
          Hstar=3.0d0/2.0d0*(H1-H2)/deqple
          HpB=const1
        end if
```

```
c-update rate independent yield stress and gradient
      call UHARD (H,Hp,equple+theta*deqple,props(8),nvalue)

c-calculate yield function and derivative value
c-check for modified NR update
      if ((.not.modnr).or.(count.eq.0)) then
        fprime=-1.0d0*(3.0d0*G*theta+3.0d0/2.0d0*HpB)-
     +          dD2*H-D2*Hp
      end if
      f=q(sija)*yt(theta,gamma,cospsa)-deqple*(3.0d0*G*theta+
     +   Hstar)-D2*H

      if (prn(14).eq.1) print *,f,fprime
      if (prn(14).eq.1) print *,deqple,D2

c-check the value of the yield function
      if (abs(f).lt.cutlim*H*D2) then
        goto 2000
      end if

c-update the equivalent plastic strain increment
      deqple=deqple-f/fprime

c-check that the next guess is positive
      if (deqple.le.0.0d0) then
        deqple=deqple/(-10.0d0)
      else if (deqple.eq.0.0d0) then
        deqple=1.0d-8
      end if

c-update counter
      count=count+1

c-loop back to make another Newton Increment
      goto 1000

    end if

2000  continue
    if (prn(13).eq.1) print *,theta,deqple

c-the equivalent plastic strain increment has been finalised
c-update the stress
    if (yield) then

c-update Hstar and HpB
      if (props(7).eq.0.0d0) then
        Hstar=0.0d0
        HpB=0.0d0
```

```
        H1=0.0d0
        H2=0.0d0
      else
        call UHARD (H1,const1,equple+deqple,props(8),nvalue)
        call UHARD (H2,const2,equple,props(8),nvalue)
        Hstar=3.0d0/2.0d0*(H1-H2)/deqple
        HpB=const1
      end if

c-calculate new theta in the event of non-proportional loading
      qb=q(sija)*yt(1.0d0,gamma,cospsa)-(3.0d0*G+Hstar)*deqple
      if (psa.gt.0.0d0) then
        if ((refde.lt.10.0d0).and.(refde.gt.0.0d0)) then
          gamma2=3.0d0*G*deque/qb
          C=exp(-1.0d0*gamma2)
          thetap=(gamma2*(1.0d0+C**2.0d0+(1.0d0-C**2.0d0)*
     +      cospsa)-(1.0d0-C**2.0d0)-(1.0d0-C)**2.0d0*cospsa)/
     +      gamma2/((1.0d0-C)**2.0d0+(1.0d0-C**2.0d0)*cospsa)
        else
          thetap=theta
        end if

        thetap=TRIMM(thetap)

        if (prn(15).eq.1) print *,qb,thetap,gamma2

c-update the stresses for the plastic case
        const1=q(sija)*yt(thetap,gamma,cospsa)
        thetab=(const1-thetap*(3.0d0*G+Hstar)*deqple)/
     +         (const1-(3.0d0*G+Hstar)*deqple)
        if (thetab.lt.1.0d0) thetab=1.0d0

        if (prn(16).eq.1) print *,thetab,yt(theta,gamma,cospsa)

        call MEQU (eta,arb1,3,3)
        call MCONST (arb1,3,3,thetab*gamma)
        call MADD (arb1,3,3,mua)
        call MCONST (arb1,3,3,1.0d0/yt(thetab,gamma,cospsa))
        call MEQU (arb1,mub,3,3)

        call MEQU (eta,arb1,3,3)
        call MCONST (arb1,3,3,theta*gamma)
        call MADD (arb1,3,3,mua)
        call MCONST (arb1,3,3,1.0d0/yt(theta,gamma,cospsa))
        call MEQU (arb1,muth,3,3)

        call MEQU (eta,arb1,3,3)
        call MCONST (arb1,3,3,thetap*gamma)
        call MADD (arb1,3,3,mua)
```

```
        call MCONST (arb1,3,3,1.0d0/yt(thetap,gamma,cospsa))
        call MEQU (arb1,mup,3,3)
      else
        call MEQU (mua,mub,3,3)
        call MEQU (mua,muth,3,3)
        call MEQU (mua,mup,3,3)
      end if

      call MEQU (mub,sijb,3,3)
      call MCONST (sijb,3,3,qb*2.0d0/3.0d0)

c-copy final deviatoric stress
      call MEQU (sijb,arb1,3,3)

c-calculate final stress angle
      call MINNER (mub,eta,3,3,const1)
      psb=acos(min(1.0d0,max(-1.0d0,2.0d0/3.0d0*const1)))

c-update backstress
      call MCONST (muth,3,3,H2-H1)
      call MADD (Bija,3,3,muth)

c-Backstress is updated and can be stored again
       do 80 i=2,7
         statev(i)=Bija(index(i-1,1),index(i-1,2))
80     continue

    end if

c-update equivalent plastic strain
    equple=equple+deqple
    statev(1)=equple

c-update volumetric part of stress increment
    volsts=volsts+Kbulk*volstn

c-add updated back stress to the current stress
    call MADD (sijb,3,3,Bija)

c-convert deviatoric stress to total stress and store
    do 90 i=1,3
      sijb(i,i)=sijb(i,i)+volsts
90  continue
    do 100 i=1,6
      stress(i)=sijb(index(i,1),index(i,2))
100 continue

    if (prn(17).eq.1) call MPRINT (stress,1,6)
    if (prn(18).eq.1) call MPRINT (statev,1,7)
```

```
3000  continue
c-calculate the plastic tangent matrix
    if (yield) then
      lambda=(theta*gamma+cospsa)/(thetap*gamma+cospsa)*
   +         yt(thetap,gamma,cospsa)/yt(theta,gamma,cospsa)
      call D2f (deqple,dtime,D,p,D2)
      call DD2f (deqple,dtime,D,p,dD2)
      call UHARD (H,Hp,equple-(1.0d0-theta)*deqple,
   +            props(8),nvalue)
      const1=deqple*thetap*3.0d0*G/(q(sija)*yt(thetap,gamma,cospsa))
      const2=theta*D2*Hp+dD2*H+theta*3.0d0*G+
   +        3.0d0/2.0d0*theta*Hp

      do 110 x=1,6
        do 120 y=1,6
          i=index(x,1)
          j=index(x,2)
          k=index(y,1)
          l=index(y,2)
          Dstar(x,y)=mup(i,j)*mup(k,l)*(2.0d0/3.0d0*const1-
   +                 (theta*2.0d0*G*lambda)/const2)*2.0d0*G
          if ((i.eq.k).and.(j.eq.l)) then
            Dstar(x,y)=Dstar(x,y)+2.0d0*G*(1.0d0-const1)
          end if
          if (k.ne.l) Dstar(x,y)=2.0d0*Dstar(x,y)
c-incorporate conversion from engineering strain
          if (y.ge.4) Dstar(x,y)=Dstar(x,y)/2.0d0
120       continue
110     continue

c-calculate elastic tangent matrix
    else
      do 130 i=1,3
        Dstar(i,i)=2.0d0*G
        Dstar(i+3,i+3)=G
130     continue
    end if

c-convert Dstar into total stress space DDSDDE matrix
    call DCONV (Dstar,DDSDDE,Kbulk)

    if (prn(18).eq.1) call MPRINT (DDSDDE,6,6)

c-give angles in degrees
    psa=psa*180.0d0/pi
    psb=psb*180.0d0/pi
    qb=q(arb1)
```

```
c-print out initial and final angles
    if (prn(19).eq.1) then
      print *,theta,thetap,gamma
      print *,psa,psb,q(arb1)
    end if

    return
    end
c*******************************************************

c*******************************************************
    doubleprecision function TRIMM (theta)
c*******************************************************

    doubleprecision theta

    if (theta.lt.0.5d0) then
      trimm=0.5d0
    else if (theta.gt.1.0d0) then
      trimm=1.0d0
    else
      trimm=theta
    end if

    return
    end

c*******************************************************

c*******************************************************
    subroutine DD2f (deqple,dtime,D,p,dD2)
c*******************************************************

    doubleprecision deqple,dtime,D,p,dD2

    dD2=(deqple)**((1.0d0-p)/p)/p/((D*dtime)**(1.0d0/p))

    return
    end
c*******************************************************

c*******************************************************
    doubleprecision function yt(theta,gamma,cospsa)
c*******************************************************

    doubleprecision theta,gamma,cospsa

    yt=sqrt((theta*gamma)**2.0d0+2.0d0*theta*gamma*cospsa+1.0d0)
```

```
      return
      end
c***********************************************************

c**************************************************
c file UMAT1.FOR
c---------------------------------------------
c Programmed for MSc(eng)
c by Derek Hulley
c**************************************************

c***********************************************************
      subroutine UMAT1(stress,statev,ddsdde,sse,spd,scd,rpl,ddsddt,
     +  drplde,drpldt,stran,dstran,time,dtime,temp,dtemp,predef,
     +  dpred,cmname,ndi,nshr,ntens,nstatv,props,nprops,coords,drot,
     +  pnewdt,celent,dfgrd0,dfgrd1,noel,npt,layer,kspt,kstep,kinc,
     +  psa,psb,qb)
c-----------------------------------------------------
c    Type of Umat selected was 1: ABAQUS rate dependent 3D
c
c    Number of state variables = 1
c-----------------------------------------------------

      doubleprecision stress,stran,dstran,ddsdde,statev,props,
     +  sse,spd,scd,rpl,ddsddt,drplde,drpldt,pnewdt,time,dtime,
     +  temp,dtemp,predef,coords,dfgrd1,dfgrd0,celent,drot,dpred

      integer ntens,nstatv,nprops,kinc,kstep,kspt,ndi,npt,noel,nshr,
     +  layer

      character*8 cmname

      dimension stress(ntens),stran(ntens),dstran(ntens),
     +  ddsdde(ntens,ntens),statev(nstatv),props(nprops),
     +  ddsddt(ntens),drplde(ntens),time(2),predef(1),dpred(1),
     +  coords(3),drot(3,3),dfgrd0(3,3),dfgrd1(3,3)

      doubleprecision E,mu,H,Hp,p,D,G,Kbulk,equple,sija,deij,deque,
     +  sijn,arb1,const1,q,D2,deqple,ehat,etilda,signot,sigbar,
     +  f,fprime,eijet,nij,dpleij,B1,H2,R,Q1,Dstar,volsts,volstn,
     +  dD2,cutlim,alpha,r1,r2,Dsij,iter1,psa,psb,mua,mub,eta,
     +  pi,sijb,qb

      integer index,i,j,k,l,x,y,count,nvalue,prn
      logical modnr,yield

      dimension index(6,2),sija(3,3),deij(3,3),sijn(3,3),sijb(3,3),
     +  arb1(3,3),ehat(3,3),eijet(3,3),nij(3,3),dpleij(3,3),
     +  Dstar(7,7),prn(10),Dsij(3,3),mua(3,3),mub(3,3),eta(3,3)
```

```
      data index/1,2,3,1,2,1,1,2,3,2,3,3/,modnr/.false./,cutlim/0.01/
      data pi/3.14159265359d0/
      data prn/0,0,0,0,0,0,0,0,0,0/
c     data prn/1,1,1,1,1,1,1,1,1,1/

c-Print out incoming variables
      if (prn(1).eq.1) then
        print *,'===>',kstep,kinc,dtime,' umat1'
        call MPRINT (stress,1,6)
        call MPRINT (stran,1,6)
        call MPRINT (dstran,1,6)
        call MPRINT (statev,1,7)
        call MPRINT (props,1,nprops)
      end if

      E = props(2)
      mu = props(3)
      p = props (4)
      D = props (5)
      deqple = 0.0d0
      count=0
      yield=.false.
      G = E/(2.0d0*(1.0d0+mu))
      Kbulk = E/3.0d0/(1.0d0-2.0d0*mu)
      nvalue=(nprops-7)/2

c-check that the number of statev is 1
      if (nstatv.lt.1) then
        print *,'**ERROR: The number of statev must be 1'
        stop
      end if

c-check to see if this is the first increment
c-set the total Equation plastic strain to zero if it is
      if ((kstep.eq.1).and.(kinc.eq.1)) then
        do 10 i=1,nstatv
          statev(i)=0.0d0
10      continue
      end if
      equple=statev(1)

c-initialize matrices
      call MZERO (sijn,3,3)
      call MZERO (deij,3,3)
      call MZERO (dpleij,3,3)
      call MZERO (Dstar,7,7)

c-convert stresses and strains to matrix form
```

```
      do 20 i=1,ntens
        sija(index(i,1),index(i,2))=stress(i)
        sija(index(i,2),index(i,1))=stress(i)
        if (index(i,1).ne.index(i,2)) then
          deij(index(i,1),index(i,2))=0.5d0*dstran(i)
        else
          deij(index(i,1),index(i,2))=dstran(i)
        end if
        deij(index(i,2),index(i,1))=deij(index(i,1),index(i,2))
20    continue

c-calculate volumetric stresses and strains
      volstn=(deij(1,1)+deij(2,2)+deij(3,3))
      volsts=(sija(1,1)+sija(2,2)+sija(2,2))/3.0d0

c-calculate deviatoric stresses at beginning of increment
c-and deviatoric strain increment
      do 30 i=1,3
        sija(i,i)=sija(i,i)-volsts
        deij(i,i)=deij(i,i)-volstn/3.0d0
30    continue

c-calculate elastic predictor stress
      const1=2.0d0*G
      call MEQU (deij,arb1,3,3)
      call MCONST (arb1,3,3,const1)
      call MADD (arb1,3,3,sija)
      call MEQU (arb1,sijn,3,3)

c-copy elastic predictor into final stress tensor
      call MEQU (sijn,sijb,3,3)

c-check the elastic predictor stress to determine if
c-plasticity has occured.  The predictor stress must be outside
c-the yield surface by at least cutlim(a predefined fraction)*
c-the current yield stress
      call UHARD(H,Hp,equple,props(8),nvalue)
      D2=1.0d0
      if (q(sijn).gt.(D2*H+cutlim*H)) then
        yield=.true.

c-check for partially elastic strain increment
c-and scale the strain increment and time step
        if (q(sija).lt.(H-cutlim*H)) then
160         continue
          call MEQU (sijn,Dsij,3,3)
          call MSUB (Dsij,3,3,sija)
          cut=cut+1
          r1=H-q(sija)
```

```
        r2=q(sijn)-q(sija)
        alpha=r1/r2
        call MCONST (Dsij,3,3,alpha)
        call MADD (sija,3,3,Dsij)
        if (q(sija).lt.(H-cutlim*H)) then
          goto 160
        end if
c-correct the entry values
        call MEQU (deij,arb1,3,3)
        call MCONST (arb1,3,3,alpha)
        call MADD (eijet,3,3,arb1)
        call MCONST (deij,3,3,(1.0d0-alpha))
        dtime=dtime*(1-alpha)
        volsts=volsts+Kbulk*(alpha*volstn)
        volstn=volstn*(1-alpha)

        if (prn(2).eq.1) print *,H,r1,r2
        if (prn(2).eq.1) print *,alpha,q(sija)
        if (prn(2).eq.1) call MPRINT (deij,3,3)
        if (prn(2).eq.1) call MPRINT (eijet,3,3)

      end if
    end if

c-calculate an equivalent strain increment and jump to elastic tangent
c-matrix calculation if zero
    call MINNER (deij,deij,3,3,deque)
    deque=sqrt(2.0d0/3.0d0*deque)
    if (deque.eq.0.d0) goto 3000

c-calculate the strain and initial stress directions
    call MEQU(deij,eta,3,3)
    call MCONST(eta,3,3,1.0d0/deque)

    const1=q(sija)
    if (const1.eq.0.0d0) const1=1.0d-3
    call MEQU(sija,mua,3,3)
    call MCONST(mua,3,3,3.0d0/2.0d0/const1)

    call MINNER (mua,eta,3,3,const1)
    psa=acos(min(1.0d0,max(-1.0d0,2.0d0/3.0d0*const1)))

c-calculate the initial total elastic strain
    call MEQU (sija,eijet,3,3)
    call MCONST (eijet,3,3,1.0d0/2.0d0/G)

c-calculate ehat
    call MEQU (eijet,ehat,3,3)
    call MADD (ehat,3,3,deij)
```

```
c-calculate etilda
      call MINNER (ehat,ehat,3,3,const1)
      etilda=sqrt(3.0d0/2.0d0*const1)

c-print out more values
      if (prn(2).eq.1) print *,yield,etilda
      if (prn(3).eq.1) call MPRINT (ehat,3,3)

c-perform update for plastic increment
      if (yield) then

c-provide a non-zero guess for deqple
        call MINNER (deij,deij,3,3,const1)
        deqple=sqrt(2.0d0/3.0d0*const1)

c-Newton method iteration to find plastic
c-strain increment

1000    if (count.eq.100) then
          print *,'Not converging'
          deqple=(iter1+deqple)/2.0d0
          goto 2000
        end if

c-update the inverse of the overstress factor
        call D2f(deqple,dtime,D,p,D2)

c-calculate sigma naught and sigma bar
c-Use the predicted equivalent plastic strain at the end
c-of the increment
        call UHARD (signot,Hp,equple+deqple,props(8),nvalue)
        sigbar=D2*signot

c-calculate yield function and derivative value
c-check for modified NR update
        if ((.not.modnr).or.(count.eq.0)) then
          fprime=-3.0d0*G-Hp*D2-signot*(deqple)**((1.0d0-p)/p)/
     +          (p*(D*dtime)**(1.0d0/p))
        end if

        f=G*(2.0d0*etilda-3.0d0*deqple)-sigbar

c-check the value of the yield function
        if ((f.lt.cutlim*H).and.(f.gt.-1.0d0*cutlim*H)) then
          if (prn(4).eq.1) print *,deqple,f,fprime
          goto 2000
        end if
```

```
c-update the equivalent plastic strain increment
      deqple=deqple-f/fprime

c-check that the next guess is positive
      if (deqple.le.0.0d0) then
        deqple=deqple/(-10.0d0)
      else if (deqple.eq.0) then
        deqple=1.0d-8
      end if

c-update counter
      count=count+1

c-loop back
      goto 1000

    end if

2000  continue

c-the equivalent plastic strain increment has been finalised
c-update the stress
    if (yield) then
      const1=2.0d0*G/(1.0d0+3.0d0*G/sigbar*deqple)
      call MEQU (ehat,sijb,3,3)
      call MCONST (sijb,3,3,const1)
    end if

c-copy final stress values
    call MEQU (sijb,arb1,3,3)

c-calculate the final stress direction
    const1=q(sijb)
    call MEQU(sijb,mub,3,3)
    call MCONST(mub,3,3,3.0d0/2.0d0/const1)

    call MINNER (mub,eta,3,3,const1)
    psb=acos(min(1.0d0,max(-1.0d0,2.0d0/3.0d0*const1)))

c-update equivalent plastic strain
    equple=equple+deqple
    statev(1)=equple

3000  continue
c-develope plastic stiffness matrix
    if (yield) then
      call UHARD (H,Hp,equple,props(8),nvalue)
      call D2f(deqple,dtime,D,p,D2)
      call DD2f(deqple,dtime,D,p,dD2)
```

```
      H2=D2*Hp
      D3=H*dD2
      B1=(H2+D3)/3.0d0/G

      if (prn(5).eq.1) print *,q(sijb),etilda
      Q1=q(sijb)/etilda
      R=3.0d0/2.0d0/sigbar/etilda*(1.0d0-deqple*(H2+D3)/sigbar)/
     +    (1.0d0+B1)

c-calculate elasto-plastic consistent tangent modulus
      do 65 x=1,6
        do 70 y=1,6
          i=index(x,1)
          j=index(x,2)
          k=index(y,1)
          l=index(y,2)
          Dstar(x,y)=0.0d0-R*sijb(i,j)*sijb(k,l)
          if ((i.eq.k).and.(j.eq.l)) then
            Dstar(x,y)=Dstar(x,y)+Q1
          end if
          if (k.ne.l) Dstar(x,y)=2.0d0*Dstar(x,y)
c-incorporate conversion from engineering strain
          if (y.ge.4) Dstar(x,y)=Dstar(x,y)/2.0d0
70      continue
65    continue

c-calculate elastic tangent matrix
    else
      do 75 i=1,3
        Dstar(i,i)=2.0d0*G
c-incorporate conversion from engineering strain
        Dstar(i+3,i+3)=G
75    continue
    end if

c-convert Dstar into total stress space DDSDDE matrix
    call DCONV (Dstar,DDSDDE,Kbulk)

c-update volumetric part of stress increment
    volsts=volsts+Kbulk*volstn

c-convert deviatoric stress to total stress and store
    do 50 i=1,3
      sijb(i,i)=sijb(i,i)+volsts
50  continue
    do 60 i=1,6
      stress(i)=sijb(index(i,1),index(i,2))
60  continue
```

```
c-print out updated total elastic strain and stress
    if (prn(6).eq.1) call MPRINT (statev,1,7)
    if (prn(7).eq.1) call MPRINT (stress,1,6)

c-print out tangent modulus
    if (prn(8).eq.1) call MPRINT (ddsdde,6,6)

c-give angles in degrees
    psa=psa*180.0d0/pi
    psb=psb*180.0d0/pi
    qb=q(arb1)

c-print out initial and final angles
    if (prn(9).eq.1) then
      print *,psa,psb,q(arb1)
    end if

    return
    end
c*****************************************************

c*****************************************************
    SUBROUTINE UHARD(SYIELD,HARD,EQPLAS,TABLE,NVALUE)
c*****************************************************
c   Subroutine to calculate the yield stress and hardening
c   parameter from the table given by props(8-)
c*****************************************************

    doubleprecision syield,hard,eqplas,table,eqpl0,eqpl1,deqpl,
   +     dsyiel,syiel1,syiel0
    integer nvalue

    DIMENSION TABLE(2,NVALUE)
C
C   SET YIELD STRESS TO LAST VALUE OF TABLE, HARDENING TO ZERO
    SYIELD=TABLE(1,NVALUE)
    HARD=0.0
C
C   IF MORE THAN ONE ENTRY, SEARCH TABLE
C
    IF(NVALUE.GT.1) THEN
     DO 10 K1=1,NVALUE-1
       EQPL1=TABLE(2,K1+1)
       IF(EQPLAS.LT.EQPL1) THEN
        EQPL0=TABLE(2,K1)
        IF(EQPL1.LE.EQPL0) THEN
          WRITE(*,1)
1          FORMAT(//,30X,'***ERROR - PLASTIC STRAIN MUST BE ',
   1           'ENTERED IN ASCENDING ORDER')
```

```
c           CALL XIT
          print*,eqpl0,eqpl1,eqplas
         ENDIF
C
C        CURRENT YIELD STRESS AND HARDENING
C
        DEQPL=EQPL1-EQPL0
        SYIEL0=TABLE(1,K1)
        SYIEL1=TABLE(1,K1+1)
        DSYIEL=SYIEL1-SYIEL0
        HARD=DSYIEL/DEQPL
        SYIELD=SYIEL0+(EQPLAS-EQPL0)*HARD
        GOTO 20
        ENDIF
 10      CONTINUE
 20      CONTINUE
      ENDIF

      RETURN
      END
c*******************************************************

c*******************************************************
      subroutine DEVSPC (xcoord,ycoord,stress)
c*******************************************************
c   Subroutine to convert a stress point into a coordinate in the x-y plane
c   Used to represent stress points in deviatoric space
c*******************************************************

      doubleprecision xcoord,ycoord,stress
      doubleprecision inv1,inv2,inv3,a,b,c,c1,c2,c3,
     +  s1,s2,s3,p1,p2,p3,n,x,y,z,volsts,theta,factor

      integer i,prn(10)

      dimension stress(6)

      data prn/0,0,0,0,0,0,0,0,0,0/
c     data prn/1,1,1,1,1,1,1,1,1,1/

c-set the scaling factor for the stresses.
c-This factor affects the x-y coordinate by (factor)
      factor=1.0d6

c-check that there is indeed stress
      call MINNER (stress,stress,1,6,a)
      if (a.eq.0.0d0) then
        xcoord=0.0d0
        ycoord=0.0d0
```

```
        return
      end if

c-calculate invariants of stress tensor
      inv1=0.0d0
      do 10 i=1,3
        inv1=inv1+stress(i)
10    continue

      a=stress(1)*stress(2)-(stress(4)**2.0d0)
      b=stress(2)*stress(3)-(stress(5)**2.0d0)
      c=stress(3)*stress(1)-(stress(6)**2.0d0)

      inv2=a+b+c

      inv3=stress(1)*b+stress(4)*(stress(4)*stress(3)-stress(6)*
     +     stress(5))+stress(6)*(stress(4)*stress(5)-stress(6)*
     +     stress(2))

      inv1=inv1/factor
      inv2=inv2/(factor**2)
      inv3=inv3/(factor**3)

      if (prn(1).eq.1) print *,inv1,inv2,inv3

c-find a root of the characteristic equation
      c1=inv1
      c2=inv2
      c3=inv3

      if (prn(2).eq.1) print *,c1,c2,c3

c-start a new search here
c-call routine to find roots of cubic
      call ROOTS (c1,c2,c3,s1,s2,s3)

      if (prn(3).eq.1) print *,s1,s2,s3

c-arrange the principal stress from highest to lowest
      p1=max(s1,s2,s3)
      p3=min(s1,s2,s3)
      p2=s1+s2+s3-p1-p3

c-project the stress point onto the PI plane (1,1,1)
c-distance to the PI plane is the volumetric stress
      volsts=(p1+p2+p3)/3.0d0
      x=p1-volsts
      y=p2-volsts
      z=p3-volsts
```

```
c-calculate the effective theta from z axis
      n=sqrt(x**2.0d0+y**2.0d0+z**2.0d0)
      if (n.eq.0.0d0) then
        xcoord=0.0d0
        ycoord=0.0d0
        return
      end if
      if ((-1.0d0*(x+y)+2.0d0*z)/n/sqrt(6.0d0).lt.-1.0d0) then
        theta=acos(-1.0d0)
      else if ((-1.0d0*(x+y)+2.0d0*z)/n/sqrt(6.0d0).gt.1.0d0) then
        theta=acos(1.0d0)
      else
        theta=acos((-1.0d0*(x+y)+2.0d0*z)/n/sqrt(6.0d0))
      end if

c-work out the coordinates for effective viewing along the
c-volumetric stress line
      ycoord=cos(theta)*n
      xcoord=sin(theta)*n*(y-x)/abs(x-y)

      return
      end
c*******************************************************

c*******************************************************
      subroutine ROOTS (c1,c2,c3,s1,s2,s3)
c*******************************************************
c        Calculates the roots of the characteristic equation
c*******************************************************

      doubleprecision c1,c2,c3,s1,s2,s3
      complex*16 f1,f2,f3,f4,sol1,sol2,sol3

      integer prn(2)
      data prn/0,0/

      f1=4.0d0*c2**3-c2**2*c1**2-18.0d0*c2*c1*c3+
     +  27.0d0*c3**2+4.0d0*c3*c1**3
      if (prn(1).eq.1) print *,f1
      f1=1.0d0/18.0d0*sqrt(f1)*sqrt((3.0d0,0))
      if (prn(1).eq.1) print *,f2
      f2=(-1.0d0/6.0d0*c2*c1+0.5d0*c3+1.0d0/27.0d0*c1**3+f1)**
     +  (1.0d0/3.0d0)
      if (prn(1).eq.1) print *,f3
      f3=1.0d0/3.0d0*c2-1.0d0/9.0d0*c1**2
      f4=1.0d0/3.0d0*c1

      sol1=f2-f3/f2+f4
```

```
      sol2=-0.5d0*f2+0.5d0*f3/f2+f4+0.5d0*(0.0d0,1)*sqrt(3.0d0)*
     +    (f2+f3/f2)
      sol3=-0.5d0*f2+0.5d0*f3/f2+f4-0.5d0*(0.0d0,1)*sqrt(3.0d0)*
     +    (f2+f3/f2)

      s1=sol1
      s2=sol2
      s3=sol3

      if (prn(2).eq.1) print *,f1,f2,f3,f4,sol1,sol2,sol3

      return
      end
c*******************************************************

c*******************************************************
      doubleprecision function F (s,c1,c2,c3)
c*******************************************************
c     Calculates the value of the characteristic equation
c*******************************************************

      doubleprecision s,c1,c2,c3

      F=s**3.0d0-c1*s**2.0d0+c2*s-c3

      return
      end
c*******************************************************

c*******************************************************
      doubleprecision function Fprime (s,c1,c2)
c*******************************************************
c     Calculates the derivative of the characteristic equation
c*******************************************************

      doubleprecision s,c1,c2

      Fprime=3.0d0*s**2.0d0-c1*2.0d0*s+c2

      return
      end
c*******************************************************

c*******************************************************
      subroutine DCONV (Dstar,DDSDDE,Kbulk)
c-------------------------------------------------------
c     Converts deviatoric stress space tangent modulus into
c     total stress space tangent modulus
c-------------------------------------------------------
```

```
      doubleprecision Dstar,DDSDDE,kbulk,A,B,arb
      integer i,j
      dimension Dstar(7,7),DDSDDE(6,6),A(6,7),B(7,6),arb(6,7)

      call MZERO (A,6,7)
      call MZERO (B,7,6)

      Dstar(7,7)=Kbulk
      do 80 i=1,3
        do 90 j=1,3
          if (i.ne.j) B(i,j)=-1.0d0/3.0d0
          if (i.eq.j) then
            A(i,j)=1.0d0
            B(i,j)=2.0d0/3.0d0
          end if
90      continue
80    continue
      do 100 i=4,6
        A(i,i)=1.0d0
        B(i,i)=1.0d0
100   continue
      do 110 i=1,3
        B(7,i)=1.0d0
        A(i,7)=1.0d0
110   continue

      call MMULT (A,Dstar,6,7,7,7,arb)
      call MMULT (arb,B,6,7,7,6,DDSDDE)

      return
      end
c************************************************************

c************************************************************
      doubleprecision function q(stress)
c-----------------------------------------------------------
c     Passes back root 3 times the equivalent uniaxial stress
c-----------------------------------------------------------

      doubleprecision stress,const
      dimension stress(3,3)

      call MINNER(stress,stress,3,3,const)
      const=const*1.5d0
      q=sqrt(const)

      return
      end
```

```
c*******************************************************

c*******************************************************
      subroutine D2f(deqple,dtime,D,p,D2)
c-------------------------------------------------------
c     Calculates the inverse of the overstress factor
c-------------------------------------------------------

      doubleprecision deqple,dtime,D,p,D2

      D2=((deqple**(1.0d0/p)+(D*dtime)**(1.0d0/p))/
     +     (D*dtime)**(1.0d0/p))

      return
      end
c*******************************************************

c*******************************************************
cFILE: MATMAN.FOR
c*******************************************************

c*******************************************************
      subroutine MPRINT (a,ar,ac)
c-------------------------------------------------------
c     Prints matrix a
c-------------------------------------------------------

      doubleprecision a
      integer ar,ac,i,j
      dimension a(ar,ac)

      do 10 i=1,ar
        write (*,1020)
1020    format (' ',$)
        do 20 j=1,ac
          write (*,1000) a(i,j)
1000      format (d10.4,' ',$)
20      continue
        write (*,1010)
1010    format (' ')
10    continue

      return
      end
c*******************************************************

c*******************************************************
      subroutine MZERO (a,ar,ac)
c-------------------------------------------------------
```

```
c   zeros matrix a
c-----------------------------------------------------

    doubleprecision a
    integer ar,ac,i,j
    dimension a(ar,ac)

    do 10 i=1,ar
      do 20 j=1,ac
        a(i,j)=0.0d0
20    continue
10  continue

    return
    end
c*******************************************************

c*******************************************************
    subroutine MEQU (a,b,ar,ac)
c-----------------------------------------------------
c   Sets matrix b=a
c-----------------------------------------------------

    doubleprecision a,b
    integer ar,ac,i,j
    dimension a(ar,ac),b(ar,ac)

    do 10 i=1,ar
      do 20 j=1,ac
        b(i,j)=a(i,j)
20    continue
10  continue

    return
    end
c*******************************************************

c*******************************************************
    subroutine MCONST (a,ar,ac,const)
c-----------------------------------------------------
c   Multiplies matrix a by a constant const.
c-----------------------------------------------------

    doubleprecision a,const
    integer i,j,ar,ac
    dimension a(ar,ac)

    do 10 i=1,ar
      do 20 j=1,ac
```

```
        a(i,j)=a(i,j)*const
20      continue
10    continue

      return
      end
c*******************************************************

c*******************************************************
      subroutine MADD (a,ar,ac,b)
c-------------------------------------------------------
c     Adds matrices a and b, and places the result in a
c-------------------------------------------------------

      doubleprecision a,b
      integer ar,ac,i,j
      dimension a(ar,ac),b(ar,ac)

      do 10 i=1,ar
        do 20 j=1,ac
          a(i,j)=a(i,j)+b(i,j)
20      continue
10    continue

      return
      end
c*******************************************************

c*******************************************************
      subroutine MSUB (a,ar,ac,b)
c-------------------------------------------------------
c     Subtracts matrices b from a, and places the result in a
c-------------------------------------------------------

      doubleprecision a,b
      integer ar,ac
      dimension a(ar,ac),b(ar,ac)
      integer i,j

      do 10 i=1,ar
        do 20 j=1,ac
          a(i,j)=a(i,j)-b(i,j)
20      continue
10    continue

      return
      end
c*******************************************************
```

```
c*********************************************************
      subroutine MTRANS (a,ar,ac,c)
c-----------------------------------------------------------
c     Transposes matrix a and places it in c
c-----------------------------------------------------------

      doubleprecision a,c
      integer ar,ac,i,j
      dimension a(ar,ac),c(ac,ar)

      do 10 i=1,ar
        do 20 j=1,ac
          c(j,i)=a(i,j)
20      continue
10    continue

      return
      end
c*********************************************************

c*********************************************************
      subroutine MMULT (a,b,ar,ac,br,bc,c)
c-----------------------------------------------------------
c     This routine multiplies two matrices a and b by each
c     other.  The result is placed in c.
c-----------------------------------------------------------

      integer ar,ac,br,bc,i,j,k
      doubleprecision a,b,c
      dimension a(ar,ac),b(br,bc),c(ar,bc)

      if (ac.ne.br) then
        print *,'Matrix dimensions do not match'
        stop
      end if
      do 10 i=1,ar
        do 20 j=1,bc
          c(i,j)=0.0d0
          do 30 k=1,ac
            c(i,j)=c(i,j)+a(i,k)*b(k,j)
30        continue
20      continue
10    continue

      return
      end
c*********************************************************

c*********************************************************
```

```
      subroutine MINNER (a,b,ar,ac,const)
c---------------------------------------------------------
c     Calculates the inner product between the matrices a and
c     b and places the result in const
c---------------------------------------------------------

      doubleprecision a,b,const
      integer ar,ac,i,j
      dimension a(ar,ac),b(ar,ac)

      const=0.0d0
      do 10 i=1,ar
        dk 20 j=1,ac
          const=const+a(i,j)*b(i,j)
20      continue
10    continue

      return
      end
c**********************************************************
```