Department of Construction Sciences
Solid Mechanics

ISRN LUTFD2/TFHF–10/5154-SE(1-54)

ELASTO-PLASTICITY IN FEM INTEPERETED AS A DAE SYSTEM

Master Thesis by
Eric Borgqvist

SUPERVISOR
DOCENT MATHIAS WALLIN, DIV. OF SOLID MECHANICS

EXAMINER
PROF. MATTI RISTINMAA, DIV. OF SOLID MECHANICS

Preface

The following Master Thesis has been the last part of my examination for a Master of Science degree in Engineering Mathematics. The thesis was carried out in Lunds Tekniska Högskola at the division of Solid Mechanics during the time period April-October 2010.

First of all I would like to direct my sincere thanks to my supervisor Docent Mathias Wallin, who introduced me to this very interesting topic and for all the time he managed to spend with me working on this thesis. It would not have been possible to finish this thesis without all the questions and discussions I had with Matthias. I would also like to thank my examinator Prof. Matti Ristinmaa and all the people at the division of Solid Mechanics for these very enjoyable six months.

Finally, I would like to thank my friends, family and my beloved Fang Shuai, for their constant support and encouragement.

Beijing, November 2010

Eric Borgqvist

Abstract

The aim of this thesis has been to present an efficient algorithm to solve the balance- and evolution equations governing elasto-plasticity. The internal variables are given by evolution equations on the local (element) level while the displacements are given on a so called global level. The global and local levels are commonly solved independently of each other, but in this thesis they are solved in a different manner. After discretization in space has been made with the finite element method a system of Differential Algebraic Equations (DAE) is established which includes both the displacements and the internal variables. A discretization in time is made with a Diagonally Implicit Runge Kutta (DIRK) method and the DAE system is then solved iteratively with the Multi Level Newton Raphson Algorithm. A suitable time step for the entire system can be achieved at a very low cost with an embedded DIRK method and thus make the method more effective. Two models are looked upon, one in small strains and one for large strains. Both models incorporate an internal variable which describes the damage of the material. The damage evolution is known to be sensitive to the integration algorithm and is therefore of interest when investigating the proposed method. The results are very satisfactory and shows that an efficient integration is indeed obtained with the proposed method.

Contents

1 Introduction **1**
- 1.1 Introduction . . . 1

2 Models **3**
- 2.1 Plasticity . . . 3
- 2.2 General structure of material models . . . 5
- 2.3 The visco-plastic model, small strains . . . 6
- 2.4 Large strains model . . . 8
  - 2.4.1 Kinematics theory . . . 8
  - 2.4.2 Model . . . 10
- 2.5 Balance equations . . . 12

3 Numerics **13**
- 3.1 Finite Element formulation . . . 13
- 3.2 Newton Raphson Method . . . 16
  - 3.2.1 Multi Level Newton Raphson . . . 16
- 3.3 Runge Kutta . . . 18
  - 3.3.1 DIRK-method . . . 21
  - 3.3.2 Error Control . . . 23
  - 3.3.3 Embedded DIRK method . . . 24

4 Procedure **25**
- 4.1 Solution procedure . . . 25
- 4.2 Integration algorithm small strains . . . 29
- 4.3 Integration algorithm large strains . . . 33
  - 4.3.1 Isochoric integration . . . 33
  - 4.3.2 Integration of evolution equations . . . 34

5 Results **37**
- 5.1 Results, small strains . . . 37
  - 5.1.1 Geometry, Rectangular disc with hole . . . 37
  - 5.1.2 Results, viscoplastic model . . . 38
  - 5.1.3 Discussion . . . 42
- 5.2 Results, Finite deformation . . . 44
  - 5.2.1 Geometry, Necking of bar . . . 44
  - 5.2.2 Results . . . 45
  - 5.2.3 Discussion . . . 48

A Useful definitions and theorems **49**
A.1 Numerical differentiation 49
A.2 Order of consistency 49
A.3 Implicit function theorem 50

B Algorithmic tangent stiffness matrices **51**
B.1 Algorithmic tangent stiffness 51
B.1.1 Differentiation of $\hat{\sigma}_{ij}$ 51
B.1.2 Differentiation of α 55
B.2 Algorithmic tangent stiffness matrix derivation, finite strains . . 56
B.2.1 Part 1 56
B.2.2 Part 2 57
B.2.3 Part 3 57

Chapter 1

Introduction

1.1 Introduction

The aim of this thesis is to present an efficient and easy to implement algorithm to solve the balance and evolution equations governing elasto-plasticity. Material models are becoming increasingly complex and accurate integration algorithms are needed, which can serve as a motivation for this thesis. The discretization of the balance equations leads to a system of non-linear algebraic equations, from which it is possible to obtain the nodal displacements on a 'global level'. The evolution equations are included on the element level, which is commonly solved by some integration algorithm (like Implicit Euler) independently of the global level. In this thesis however, a different interpretation of the governing equations will be made. The same interpretation that is made in [3] will be followed, where both the discretization of the balance equations together with the constitutive model defined locally, is seen as a system of differential algebraic equation (DAE). Powerful numerical techniques used in numerical analysis can be employed with this interpretation.

The technique that is going to be used is a Diagonally Implicit Runge-Kutta (-DIRK) method together with the Multi-Level Newton Raphson Algorithm. Both the displacements and the internal variables will be approximated in the time domain with the DIRK-method. This will make it possible to obtain a local error estimation for the entire system. A suitable time step can then be determined by some predefined tolerances with the help of the error estimation. The proposed Runge-Kutta method has an accuracy of order 2 and is both A- and S-stable (cf. [3]). The method can thus handle complex material models in a very efficient manner. The structure of the proposed algorithm is very similar to the structure of an Implicit Euler (IE) algorithm. It will turn out that very little has to be changed when using this algorithm if an existing FE-code capable of handling elasto-plasticity (where the IE-method has been used) already exists. All the major changes can then be done in a "main" program and most of the subroutines can be left intact

The algorithm will be tested and evaluated on two material models, one for small deformations and one for large (finite) deformations. Both of these models will have an internal variable which describes the damage evolution of the solid. The damage evolution is known to be sensitive to the integration procedure due to its exponential development cf. [18]. It is therefore of interest to compare how this method handles these sensitive material models.

In this text, it is assumed that the reader is familiar and has some background with the finite element method. Tensors will be widely used, where both index and direct notation will be employed. Both tensors and matrices will be written with **bold** text and it has to be understood from the context when which is meant. The index notation is however mostly used in the small deformation model while the direct notation is used on the large deformation model. The identity tensor in index notation will be denoted by δ_{ij} and the identity tensor in direct notation will be denoted by $\boldsymbol{I}$.

Chapter 2

Models

2.1 Plasticity

In the present section, a brief summary of plasticity will be presented. A more comprehensive review about plasticity can be found in for example [13] and [10]. A material is defined to be elastic when the current stress, σ, can be uniquely determined by the current strains ε on the material. For example, Hookes law in an uniaxial state of stress is given by:

$$\sigma = E\varepsilon \tag{2.1}$$

where E is the modulus of elasticity. The stress of the solid during both loading and unloading is determined by (2.1). However, in elasto-plasticity theory this relation is only valid until the effective stress σ_y (for example von Mises effective stress) has reached a certain threshold value σ_{y0}, where σ_{y0} is known as the initial yield stress. The solid is said to have undergone permanent deformation after the initial yield stress has been reached. Consider figure 2.1.

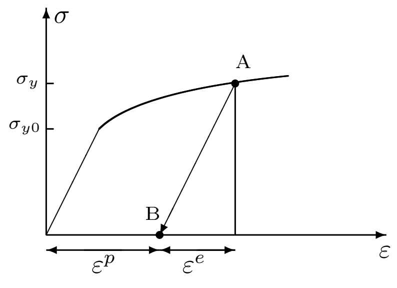


Figure 2.1: Elasto-plastic loading and unloading

Permanent deformation is characterized by the plastic strains which is developed after $\sigma_y > \sigma_{y0}$. Assume that there is a linear relation between the stresses and strains before yielding has occurred. After the yield stress has been reached, the stresses no longer varies linearly, but unloading is still done linearly after $\sigma_y > \sigma_{y0}$ as seen in figure (2.1). Unloading until $\sigma_y = 0$, will leave some strains in the body, these strains are denoted by ε^p as shown in figure (2.1). The total strains at point A in the figure then consists of

$$\varepsilon = \varepsilon^e + \varepsilon^p \tag{2.2}$$

If loading should be done from point B, then this load would again be applied linearly until point A is reached and then the plasticity of the body will continue to develop. The stresses are said to be path dependent in plasticity since there exists infinite possibilities to reach the point A.

A yield function f is introduced in order to describe the plastic deformation process. The yield function describes a convex surface in the stress space and as long as the effective stresses σ_y are inside the surface, then elastic deformation is said to take place. Consider figure 2.2.

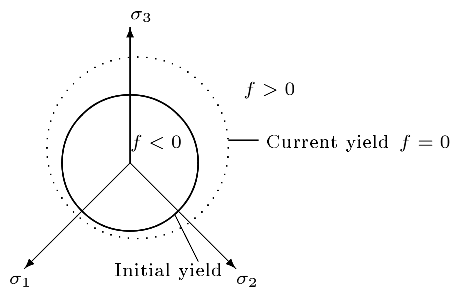


Figure 2.2: Yield surface in the deviatoric plane

Figure (2.2) is plotted in the so called deviatoric plane. The axis in the figure are directed along the principal directions of the stress tensor σ_{ij}, and the yield surface is drawn in a plane perpendicular to the hydrostatic axis $n = \frac{1}{\sqrt{3}}(1,1,1)$ where n has the normed principal directions as bases (cf. [13]).

Linear elastic deformation takes place when inside the surface, $f < 0$. If however $f = 0$ the initial yield stress has been reached and then plasticity occurs. The yield surface changes during plastic deformation and this is described by a set of internal variables $q_1, q_2, ...q_m$. The yield surface is a function of these internal variables, i.e. $f = f(\sigma_{ij}, q_1, q_2, ..., q_m)$. The internal variables are usually given in terms of some differential equations, called evolution equations. These evolution equations can usually be derived from the laws of thermodynamics but the focus of this thesis is however the solution scheme to these evolution laws and the reader is therefore suggested to read some of the references in order to get a better insight into these equations. The evolution laws in this theses will be taken as postulates and then the point of departure will be taken from there.

2.2 General structure of material models

For a rate independent elasto-plastic solid the internal variables and stresses are forced to be on the yield surface $f = 0$ while in viscoplasticity it is allowed to have an $f > 0$. In the former case $f = 0$ can be seen as a constraint equation to our system. The following structure for the developed material models (defined locally on the solid) is assumed in the elasto-plastic case:

$$\left.\begin{bmatrix} \sigma_{ij} & = & g_{ij}(\varepsilon_{pq}, q_1, q_2, ..., q_m) \\ \dot{q}_k & = & 0 \end{bmatrix}\right\} \text{if } f(\sigma_{ij}, q_1, q_2, ..., q_m) \leq 0 \tag{2.3}$$

$$\left.\begin{bmatrix} \sigma_{ij} & = & g_{ij}(\varepsilon_{pq}, q_1, q_2, ..., q_m) \\ \dot{q}_k & = & \dot{\lambda} L_k(\sigma_{ij}, q_1, q_2, ..., q_m) \\ f(\sigma_{ij}, q_1, q_2, ..., q_m) & = & 0 \end{bmatrix}\right\} \text{if } f(\sigma_{ij}, q_1, q_2, ..., q_m) > 0 \tag{2.4}$$

where λ is the so called plastic multiplier. The plastic multiplier λ is given from KKT conditions for rate independent plasticity in a minimization problem cf.[17]. In this thesis however, we simply see it as variable that is postulated in our evolution equations. The equations $\dot{q_k} = \dot{\lambda} R_k(\sigma_{ij}, q_1, q_2, ..., q_m)$ are the evolution equations and $g_{ij}(\varepsilon_{pq}, q_1, q_2, ..., q_m)$ is the elastic law which calculates the stress. However the small strain model considered in this thesis is a visco-plastic model, where the yield function is allowed to be greater than zero $f > 0$. The structure of the visco-plastic model is assumed to be

$$\left.\begin{bmatrix} \sigma_{ij} & = & g_{ij}(\varepsilon_{pq}, q_1, q_2, ..., q_m) \\ \dot{q}_k & = & 0 \end{bmatrix}\right\} \text{if } f(\sigma_{ij}, q_1, q_2, ..., q_m) \leq 0 \tag{2.5}$$

$$\left.\begin{bmatrix} \sigma_{ij} & = & g_{ij}(\varepsilon_{pq}, q_1, q_2, ..., q_m) \\ \dot{q}_k & = & \frac{1}{\eta}\left[\frac{f}{\sigma_{y0}}\right]^r L_k(\sigma_{ij}, q_1, q_2, ..., q_m) \end{bmatrix}\right\} \text{if } f(\sigma_{ij}, q_1, q_2, ..., q_m) > 0 \tag{2.6}$$

The exponent r is a real positive number and η is a viscosity parameter. Compared with the elasto-plastic model, the plastic multiplier has been replaced with $\frac{1}{\eta}\left[\frac{f}{\sigma_0}\right]^r$ and the constraint constraint $f(\sigma_{ij}, q_1, q_2, ..., q_m) = 0$ from (2.4) has been dropped. It can however be shown that the elasto-plastic model can be obtained from the viscoplastic model by letting $\eta \to 0$ (cf. [7] and [16]).

2.3 The visco-plastic model, small strains

The special material model that will be implemented in the small deformation case is a mixed von Mises hardening model with Armstrong/Frederick Kinematic terms coupled with damage in visco-plasticity. The model is the same as can be found in [3] and [7] but an extra damage variable has been added to the set of internal variables. The following internal variables, q_k, will be adopted:

$$\{q_k\} = \{\varepsilon^p_{ij}, X_{ij}, \varepsilon^p_{eff}, \alpha\}$$

ε^p_{ij} is the plastic strain, X_{ij} is the back stress from the kinematic hardening, ε^p_{eff} is the effective plastic strain and α is the damage variable. The yield function for this material model is given by:

$$f(\sigma_{ij}, X_{ij}, \varepsilon^p_{eff}, \alpha) = \frac{1}{2}(\frac{s_{ij}}{1-\alpha} - X^{dev}_{ij})(\frac{s_{ij}}{1-\alpha} - X^{dev}_{ij}) - \frac{\Big(\sigma_y(\varepsilon^p_{eff})\Big)^2}{3}$$

s_{ij} and X^{dev}_{ij} are the deviatoric part of the stress σ_{ij} and back stress X_{ij} respectively which are given by

$$s_{ij} = \sigma_{ij} - \frac{1}{3}\sigma_{kk}\delta_{ij}, \quad X^{dev}_{ij} = X_{ij} - \frac{1}{3}X_{kk}\delta_{ij} \tag{2.7}$$

$\sigma_y(\varepsilon^p_{eff})$ is a function which describes the hardening (the expansion of the yield surface) and has the following form:

$$\sigma_y = \sigma_{yo} + K(\varepsilon^p_{eff}) \tag{2.8}$$

where $K(\varepsilon^p_{eff})$ is a hardening function depending on the plastic effective strain. The evolution equations for the internal variables will now be presented. The Armstrong/Frederick model is used for the kinematic variables X_{ij}

$$\dot{X}_{ij} = c\dot{\varepsilon}^p_{ij} - bX_{ij} \tag{2.9}$$

The plastic strains are assumed to develop according to:

$$\dot{\varepsilon}^p_{ij} = \frac{1}{\eta}\left[\frac{f}{\sigma_0^2}\right]^r N_{ij} \tag{2.10}$$

where the direction of the plastic flow is given by

$$N_{ij} = \frac{(\frac{s_{ij}}{1-\alpha} - X^{dev}_{ij})}{||(\frac{s_{pq}}{1-\alpha} - X^{dev}_{pq})||} \tag{2.11}$$

Where the euclidean norm has been used in the definition of N_{ij}. Note that the plastic strains has the property $\varepsilon^p_{ii} = 0$, which means that this internal variable does not affect the volume of the solid. With this evolution equation for the plastic strains, the following definition for the effective plastic strain rate ($\dot{\varepsilon}^p_{eff}$) is obtained

$$\dot{\varepsilon}^p_{eff} = \sqrt{\frac{2}{3}\dot{\varepsilon}^p_{ij}\dot{\varepsilon}^p_{ij}} = \sqrt{\frac{2}{3}}\frac{1}{\eta}\left[\frac{f}{\sigma_0^2}\right]^r \tag{2.12}$$

Further on, the evolution of the damage variable α is assumed to evolve according to

$$\dot{\alpha} = -\frac{1}{\eta}\left(\frac{f}{\sigma_0^2}\right)^r \frac{Y}{S_d(1-\alpha)^m} \tag{2.13}$$

where S_d and m are two constitutive parameters, where S_d is known as the damage modulus. Y is a measure of the energy release given by (cf. [18])

$$Y = -\frac{1}{2}\varepsilon^e_{ij}D^{hooke}_{ijkl}\varepsilon^e_{kl} \tag{2.14}$$

where D^{hooke}_{ijkl} is the fourth order tensor given by hooke's law (cf. [13]) Hooke's law will be used to describe the relation between the stresses and the elastic strains within the yield surface ($f \leq 0$)

$$\sigma_{ij} = (1-\alpha)D^{hooke}_{ijkl}\varepsilon^e_{kl} = (1-\alpha)K_b\varepsilon_{kk}\delta_{ij} + (1-\alpha)2G(e_{ij} - e^{(p)}_{ij}) \tag{2.15}$$

$K_b = \frac{E}{3(1-2\nu)}$ is the bulk modulus and $G = \frac{E}{3(1-2\nu)}$ is the shear modulus. The deviatoric part of the strains (e_{ij} and e^p_{ij}) above is defined in a similar manner as (2.7).

A system of ordinary differential equations for the internal variables can now be formed. The system with the same form as (2.6) is summarized below for convenience

$$\begin{aligned}
\sigma_{ij} &= (1-\alpha)K_b\varepsilon_{kk}\delta_{ij} + (1-\alpha)2G(e_{ij} - e^p_{ij}) && (2.16)\\
\dot{\varepsilon}^p_{ij} &= \frac{1}{\eta}\left[\frac{f}{\sigma_0^2}\right]^r N_{ij} && (2.17)\\
\dot{X}_{ij} &= \frac{1}{\eta}\left[\frac{f}{\sigma_0^2}\right]^r \left(cN_{ij} - b\sqrt{\frac{2}{3}}X_{ij}\right) && (2.18)\\
\dot{\varepsilon}^p_{eff} &= \frac{1}{\eta}\left[\frac{f}{\sigma_0^2}\right]^r \sqrt{\frac{2}{3}} && (2.19)\\
\dot{\alpha} &= -\frac{1}{\eta}\left(\frac{f}{\sigma_0^2}\right)^r \frac{Y}{S(1-\alpha)^m} && (2.20)
\end{aligned}$$

2.4 Large strains model

2.4.1 Kinematics theory

In this section some necessary terminology will be introduced in order to work with large deformation processes in the finite element method. For a further review, the reader is referred to [10] or [15].

The reason for why a non-linearity arise in large deformation models is because equilibrium is considered both in a reference configuration $\mathcal{N}_0 \in \mathbb{R}^3$ at the time instance t_0 and the current configuration $\mathcal{N} \in \mathbb{R}$ at time instance t. The non-linear map relating the two configurations are given by $\boldsymbol{x} = \boldsymbol{\chi}(\boldsymbol{x_0}, t) : \mathcal{N}_0 \times T \rightarrow \mathcal{N}$, where $\boldsymbol{x}_0$ denotes the position of some particle in the reference configuration and $\boldsymbol{x}$ the position of the same particle in the current configuration in the time interval T. The mapping of an infinitesimal vector $d\boldsymbol{x_0}$ in the reference configuration to the current configuration is defined by the partial derivatives to the function $\boldsymbol{x}(\boldsymbol{x_0})$ as

$$d\boldsymbol{x} = \boldsymbol{\nabla}_0(\boldsymbol{x})d\boldsymbol{x}_0 = \boldsymbol{F}d\boldsymbol{x}_0 \tag{2.21}$$

where $\boldsymbol{F}$ is the deformation gradient and $\boldsymbol{\nabla}_0$ is differential operator acting on $\boldsymbol{x}$. In a Cartesian frame, the operator can be expressed as:

$$\boldsymbol{\nabla}_0(\boldsymbol{x}) = \left[\frac{\partial}{\partial x_j^0}(x_i)\right] \tag{2.22}$$

and the components of the deformation gradient are then given by

$$[\boldsymbol{F}]_{ij} = \begin{pmatrix} \partial x_1/\partial x_1^0 & \partial x_1/\partial x_2^0 & \partial x_1/\partial x_3^0 \\ \partial x_2/\partial x_1^0 & \partial x_2/\partial x_2^0 & \partial x_2/\partial x_3^0 \\ \partial x_3/\partial x_1^0 & \partial x_3/\partial x_2^0 & \partial x_3/\partial x_3^0 \end{pmatrix} \tag{2.23}$$

The position of the particle in the current configuration can be split into two parts with the help of the current displacements $\boldsymbol{u}$.

$$\boldsymbol{x} = \boldsymbol{x}_0 + \boldsymbol{u} \tag{2.24}$$

This means that the deformation gradient in a similar manner can be split into

$$\boldsymbol{F} = \boldsymbol{I} + \boldsymbol{D_u} \tag{2.25}$$

Where the components of the tensor $\boldsymbol{D_u}$ is given by

$$[\boldsymbol{D_u}]_{ij} = \begin{pmatrix} \partial u_1/\partial x_1^0 & \partial u_1/\partial x_2^0 & \partial u_1/\partial x_3^0 \\ \partial u_2/\partial x_1^0 & \partial u_2/\partial x_2^0 & \partial u_2/\partial x_3^0 \\ \partial u_3/\partial x_1^0 & \partial u_3/\partial x_2^0 & \partial u_3/\partial x_3^0 \end{pmatrix} \tag{2.26}$$

In the material model developed, it will be assumed that the deformation gradient $\boldsymbol{F}$ can be split into an elastic and plastic part according to

$$\boldsymbol{F} = \boldsymbol{F}^e \boldsymbol{F}^p \tag{2.27}$$

where the plastic deformation gradient $\boldsymbol{F}^p$ is assumed to be isochor, i.e. it will not affect the volume in the map from $\boldsymbol{x}_0$ to $\boldsymbol{x}$. This statement is equivalent to that the determinant of the plastic gradient is equal to one, $\det \boldsymbol{F}^p = 1$. This can be compared with the infinitesimal theory where the strains are divided into $\varepsilon_{ij} = \varepsilon^e_{ij} + \varepsilon^p_{ij}$ and the plastic strains has the property $\varepsilon^p_{kk} = 0$. The plastic deformation gradient $\boldsymbol{F}^p$ will be taken as an internal variable in the model presented in next section. An illustration of the map from the original configuration $\mathcal{N}_0$ to the current configuration $\mathcal{N}$ with the intermediate configuration $\mathcal{N}_p$ is displayed in figure (2.3). The figure is directly taken from [19].

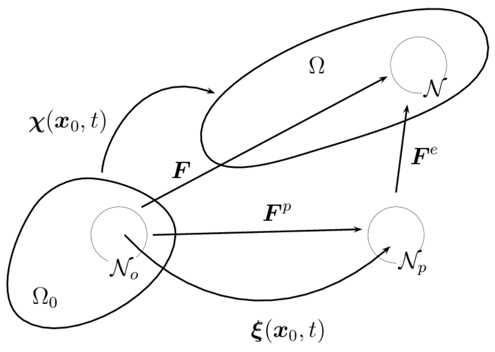


Figure 2.3: Illustration of the mapping of particle $\boldsymbol{x}_0$ to the intermediate and current configuration. Figure is taken from [19]

The strain measure that is going to be used is given by

$$\boldsymbol{E} = \frac{1}{2}(\boldsymbol{C} - \boldsymbol{I}) \tag{2.28}$$

where the Cauchy-Green tensor $\boldsymbol{C}$ is given by $\boldsymbol{F}^T\boldsymbol{F}$. The stresses in the current configuration $\mathcal{N}_0$ is denoted by $\boldsymbol{T}$. The balance equation that will be presented later will however be integrated in the reference configuration. A stress measurement in this configuration $\mathcal{N}_0$ is called the second Piola-Kirchoff stress tensor $\boldsymbol{S}$ defined by:

$$\boldsymbol{S} = J\boldsymbol{F}^{-1}\boldsymbol{T}\boldsymbol{F}^{-T} \tag{2.29}$$

where $J = \det \boldsymbol{F}$. Further on, it is assumed that the differentiated second Piola-Kirchhoff stress $d\boldsymbol{S}$ can be written as a function of the differentiated Cauchy-Green strains $d\boldsymbol{E}$ according to

$$d\boldsymbol{S} = (1 - \alpha)\boldsymbol{D} : d\boldsymbol{E} - d\alpha\boldsymbol{S} \tag{2.30}$$

where $\boldsymbol{D}$ is known as the tangent stiffness tensor. With these introductory remarks on finite deformation theory, we are now ready to present the material model of interest.

2.4.2 Model

Useful kinematic relations between the deformation gradient and other important tensors used in the large strains model is summarized in table (2.1). Note that all the quantities in the box that has a 'hat' upon them are isochor, i.e the determinant is equal to one.

$\hat{\boldsymbol{F}} = J^{-1/3}\boldsymbol{F}$	$J = \det \boldsymbol{F}$
$\hat{\boldsymbol{F}}^e = (J^e)^{-1/3}\boldsymbol{F}^e$	$J^e = \det \boldsymbol{F}^e$
$\hat{\boldsymbol{F}}^k = (J^k)^{-1/3}\boldsymbol{F}^k$	$J^k = \det \boldsymbol{F}^k$
$\boldsymbol{C} = \boldsymbol{F}^T\boldsymbol{F}$	$\hat{\boldsymbol{C}} = J^{-2/3}\boldsymbol{C}$
$\boldsymbol{C}^e = \boldsymbol{F}^{eT}\boldsymbol{F}^e$	$\hat{\boldsymbol{C}}^e = (J^e)^{-2/3}\boldsymbol{C}^e$
$\boldsymbol{C}^k = \boldsymbol{F}^{kT}\boldsymbol{F}^k$	$\hat{\boldsymbol{C}}^k = (J^k)^{-2/3}\boldsymbol{C}^k$

Table 2.1: Useful quantities in deformation model

The model that will be developed is an elasto-plastic incompressible material model based upon [19], but with an extra damage variable incorporated. The yield function in the model will be a function of the Mandel stress $\boldsymbol{\Sigma}$ which is given by the expression

$$\boldsymbol{\Sigma} = (1-\alpha)K_b \ln J^e \boldsymbol{1} + (1-\alpha)G \ln \hat{\boldsymbol{C}}^{e,dev} \tag{2.31}$$

where K_b and G are the bulk and shear modulus respectively. The damage variable is denoted by α. The logarithmic elastic Cauchy-Green tensor $\ln \hat{\boldsymbol{C}}^{(e,dev)}$ can be calculated through eigenvector decomposition (cf. [11]). The second Piola Kirchoff stress tensor $\boldsymbol{S}$ that was introduced before, is related to the Mandel stress through

$$\begin{aligned} \boldsymbol{S}^e &= (\boldsymbol{C}^e)^{-1}\boldsymbol{\Sigma} \\ \boldsymbol{S} &= (\boldsymbol{F}^p)^{-1}\boldsymbol{S}^e(\boldsymbol{F}^p)^{-T} \end{aligned} \tag{2.32}$$

where $\boldsymbol{S}^e$ is a second-Piola Kirchoff stress mapped from the intermediate configuration $\mathcal{N}_p$. The plastic deformation gradient $\boldsymbol{F}^p$ will be taken as an internal variable as stated previously. Another internal variable used in the yield condition $f = 0$ will be the kinematic deformation gradient $\boldsymbol{F}^k$, which is based upon introducing another configuration $\mathcal{N}_k$ from which particles in the reference configuration can be mapped onto (cf. [19]). The back stress tensor $\boldsymbol{X}$ subsequently used in the yield condition is given by

$$\boldsymbol{X} = \xi \ln \boldsymbol{C}^{k,dev} \tag{2.33}$$

where ξ is a constitutive parameter. The damage variable is governed by the same evolution equation as in the small strains case, but with another measure of the energy release rate Y, i.e.

$$Y = -\frac{1}{2}K_b(\ln J^e)^2 - \frac{G}{4}(\ln \boldsymbol{C}^{e,dev} : \ln \boldsymbol{C}^{e,dev}) \tag{2.34}$$

The yield condition in the large deformation model can now be defined as

$$f(\boldsymbol{\Sigma}(\boldsymbol{F}^e), \boldsymbol{X}(\boldsymbol{F}^k), \alpha) = \sqrt{\frac{3}{2}(\frac{\boldsymbol{\Sigma}^{dev}}{1-\alpha} - \boldsymbol{X}) : (\frac{\boldsymbol{\Sigma}^{dev}}{1-\alpha} - \boldsymbol{X})} - \sigma_{y0} \tag{2.35}$$

The material model written in the same format as in (2.4) for the internal variables are postulated to be:

$$\begin{array}{lcl} \boldsymbol{\Sigma} & = & (1-\alpha)K_b \ln J^e \boldsymbol{1} + (1-\alpha)G \ln \hat{\boldsymbol{C}}^{e,dev} \\ \dot{\boldsymbol{F}}^p & = & = \dot{\lambda}\boldsymbol{N}^p\boldsymbol{F}^p \\ \dot{\boldsymbol{F}}^k & = & \dot{\lambda}\boldsymbol{F}^k\boldsymbol{N}^k \\ \dot{\alpha} & = & -\dot{\lambda}\frac{Y}{S_d(1-\alpha)^m} \end{array} \tag{2.36}$$

The tensors $\boldsymbol{N}^p$ and $\boldsymbol{N}^k$ are given by the expression

$$\begin{array}{lcl} \boldsymbol{N}^p & = & \frac{3}{2\sigma_{y0}(1-\alpha)}\left(\frac{\boldsymbol{\Sigma}^{dev}}{1-\alpha} - \boldsymbol{X}\right) \\ \boldsymbol{N}^k & = & \frac{3}{2\sigma_{y0}}\left(\frac{\boldsymbol{\Sigma}^{dev}}{1-\alpha} - \boldsymbol{X}\right) - \Gamma\frac{3}{2}\boldsymbol{X} \end{array} \tag{2.37}$$

where Γ is a constitutive parameter. The reader is again referred to [19] and [20] for a more comprehensive review on the material model.

2.5 Balance equations

The balance equations that are going to be solved is now presented. This is however only done in the large strains case, since the balance laws in small strains can be obtained as a special case of the equations presented here. The reader is refered to [12] or [13] for a derivation of the balance equations in small strains.

The balance equations in large strains are expressed in the reference configuration with the help of the principle of virtual power. The derivation of this can be found in [15]. The expression is given by

$$\int_{V^0} \boldsymbol{E}^{(w)} : \boldsymbol{S} dV^0 - \Pi^{ext} = \boldsymbol{0} \tag{2.38}$$

The integration is made over the volume of the solid in the reference configuration V^0. The second Piola-Kirchoff stress $\boldsymbol{S}$ was defined in (2.29). The virtual strain $\boldsymbol{E}^{(w)}$ is given by the expression

$$\boldsymbol{E}^{(w)} = \frac{1}{2}(\boldsymbol{\nabla}_0\boldsymbol{w} + (\boldsymbol{\nabla}_0\boldsymbol{w})^T + (\boldsymbol{\nabla}_0\boldsymbol{w})^T\boldsymbol{\nabla}_0\boldsymbol{u} + (\boldsymbol{\nabla}_0\boldsymbol{u})^T\boldsymbol{\nabla}_0\boldsymbol{w}) \tag{2.39}$$

where $\boldsymbol{w}$ is a weight function which can be chosen arbitrarily. By exchanging $\boldsymbol{w}$ with $\boldsymbol{u}$ then an additional expression which is equal to the Cauchy-Green strains $\boldsymbol{E}$ defined in (2.28) can be found. The external virtual work Π^{ext} is given by the expression

$$\Pi^{ext} = \int_{S^0} \boldsymbol{w}^T\boldsymbol{t}^0 dS^0 - \int_{V^0} \boldsymbol{w}^T\boldsymbol{b} dV^0 \tag{2.40}$$

where $\boldsymbol{t}^0$ are the external traction forces acting on the body in the reference configuration and $\boldsymbol{b}$ are the body forces. The surface S^0 consists of two parts $S^0 \in S_f \cup S_u$. On S_u the displacements of the body are assumed to be known (essential boundary conditions) while on S_f the external traction forces acting on the solid is assumed to be known (natural boundary conditions). The aim is to solve (2.38) together with the internal variables in order to compute deformation of the solid.

Chapter 3

Numerics

3.1 Finite Element formulation

The finite element formulation of the balance equations (2.38) and the system of internal variables will now be presented. This will only be presented for the large deformation case. Matrices (i.e. not tensors) will be used in this section. The reader is assumed to be familiar with the finite element terms described in this section and we refer the reader to [12] for a review on the FEM-formulation. The solid of interest is discretized into n_{elm} elements and the displacements $\boldsymbol{u}$ are described with n_u degrees of freedom. The displacements $\boldsymbol{u}(t)$ at time t of the solid is expressed with the help of n_e global shape functions N_k according to

$$\boldsymbol{u}(\boldsymbol{x},t) = \sum_{k=1}^{n_u} N_k(x)a_k(t) = \boldsymbol{N}(x)\boldsymbol{a}(t) \tag{3.1}$$

$\boldsymbol{a}(t)$ describes the nodal displacements and is a function of time only, while the shape functions N_k are functions of the position $\boldsymbol{x}$ only. The global shape functions N_k has further been ordered in a matrix $\boldsymbol{N}$ above.

The balance equations that was established in chapter 2.5 are now discretized since the underlying variables in the formulations depends on the displacement $\boldsymbol{u}$. In the small strains case, the discretized strains are given by direct differentiation of the current displacement with respect to the positions $\boldsymbol{x}$. In the large strains case however, the deformation gradient $\boldsymbol{F}$ is needed, which can be computed for one element with the help of (3.1) and (2.22). The weight functions in (2.38) are chosen according to Galerkin, $\boldsymbol{w}_m = \boldsymbol{Nc}$ where $\boldsymbol{c}$ is an arbitrary vector. Here $\boldsymbol{w}_m$ denotes a $n_e \times 1$ vector and in order to put this vector in the expression of the virtual strains an introduction of some new differential operators can be useful. How exactly this is made can be found in [15], but here we simply state that after some algebraic manipulations has been made, then the virtual strains $\hat{\boldsymbol{E}}$ and in a similar manner the Cauchy green strains $\boldsymbol{E}$ can be written as

$$\begin{aligned}\hat{\boldsymbol{E}} &= \boldsymbol{B}\boldsymbol{c} \\ \boldsymbol{E} &= \boldsymbol{B}\boldsymbol{a}\end{aligned} \tag{3.2}$$

where the matrix $\boldsymbol{B}$ is a function of the displacements $\boldsymbol{u}$. The second Piola-Krichoff stress is ordered into a matrix $\boldsymbol{S}$ and by putting in the finite element approximation into (2.38) and noting that $\boldsymbol{c}$ can be chosen arbitrarily, then the following expression is obtained

$$\boldsymbol{R_G} = \int_{V^0} \boldsymbol{B}^T \boldsymbol{S}(\boldsymbol{u}(t), \boldsymbol{q}(t)) dV^0 - \boldsymbol{f}^{ext} = \boldsymbol{0} \tag{3.3}$$

with

$$\boldsymbol{f}^{ext} = \int_{S^0} \boldsymbol{N}^T \boldsymbol{t} dS^0 - \int_{V^0} \boldsymbol{N}^T \boldsymbol{b} dV^0 \tag{3.4}$$

The first integral above is further defined as $\int_V \boldsymbol{B}^T \boldsymbol{S} dV \boldsymbol{a} = \boldsymbol{f}^{int}$. The volume and surface integral above is calculated using gauss integration (c.f [12]), where triangular elements with 1 gauss point have been used in the small strain case and rectangular isoparametic elements with 4 gauss points in the large strains case. However a mixed finite element formulation will be implemented in the large strain case, but it will not be covered in this thesis, since the addition of the mixed formulation does not matter in integration scheme developed. A short and not rigorous explanation of the mixed formulation is that it relaxes the relation $J = \det \boldsymbol{F}$ such that J is spread out through the element (instead of calculating it at the gauss points) and in a sense weakens the conditions on J. The mixed formulation is given in [19] and a deeper review of it can be found in [1]. Axisymmetric elements will be used in the large strains case, which also can be read more about in [1].

The stresses and internal variables given in the general format (2.4) and (2.6), have now been discretized as well, since these underlying equations also depends on the displacements. Order now the differentiated and discretized internal variables $\dot{q}_k(\sigma_{ij}(u), q_1, q_2, \ldots, q_m)$ in a a vector $\dot{\boldsymbol{q}}$ and the functions $\dot{\lambda} L_k(\sigma_{ij}(u), q_1, q_2, \ldots, q_m)$ into an matrix $\boldsymbol{L}(\boldsymbol{u}(t), \boldsymbol{q}(t))$. Further on, if an elasto-plastic model is considered where the additional condition $f = 0$ exits, then this condition is also ordered into the matrix $\boldsymbol{L}(\boldsymbol{u}(t), \boldsymbol{q}(t))$. A residual for the evolution laws and the constraint equation $f = 0$ can then be formed

$$\boldsymbol{R_L} = \boldsymbol{A}\dot{\boldsymbol{q}}(t) - \boldsymbol{L}(\boldsymbol{u}(t), \boldsymbol{q}(t)) = \boldsymbol{0} \tag{3.5}$$

The matrix $\boldsymbol{A}$ is introduced to make sure that no $\dot{\boldsymbol{q}}$ is included in the constraint equation $f = 0$, i.e the corresponding row in $\boldsymbol{A}$ which gives the constraint equation $f = 0$ is filled with zeros and the rest of the matrix is a corresponding identity matrix. If this system is inserted into an additional matrix $\boldsymbol{F}(t, \boldsymbol{y}(t), \dot{\boldsymbol{y}}(t))$, then a differential algebraic equation system of first order can be formed according to

$$\boldsymbol{F}(t,\boldsymbol{y}(t),\dot{\boldsymbol{y}}(t)) = \left\{ \begin{array}{c} \boldsymbol{R_G} \\ \boldsymbol{R_L} \end{array} \right\} = \left\{ \begin{array}{l} \int_V \boldsymbol{B}^T \boldsymbol{S}(\boldsymbol{u}(t),\boldsymbol{q}(t))dV - \boldsymbol{f}^{ext} \\ \boldsymbol{A}\dot{\boldsymbol{q}}(t) - \boldsymbol{L}(\boldsymbol{u}(t),\boldsymbol{q}(t)) \end{array} \right\} = \boldsymbol{0} \quad (3.6)$$

where the unknowns $\boldsymbol{y}(t)$ are given by

$$\boldsymbol{y}(t) = \left(\begin{array}{c} \boldsymbol{u}(t) \\ \boldsymbol{q}(t) \end{array} \right) \quad (3.7)$$

with the initial conditions

$$\boldsymbol{y}(t_0) = \left(\begin{array}{c} \boldsymbol{u}(t_0) \\ \boldsymbol{q}(t_0) \end{array} \right) = \left(\begin{array}{c} \boldsymbol{u}_0 \\ \boldsymbol{q}_0 \end{array} \right) = \boldsymbol{y}_0 \quad (3.8)$$

The system (3.6) is now interpreted as a DAE system with index 1. This will be solved by first approximating $\boldsymbol{y}(t)$ and $\dot{\boldsymbol{y}}(t)$ with the Diagonally Implicit Runge-Kutta (-DIRK) method and then the system $\boldsymbol{F} = 0$ will be solved iteratively with the Multi Level Newton Raphson Algorithm (MLNRA).

NOTE: The common way to proceed is to treat the global equations $\boldsymbol{R}_G$ and the local residual equations $\boldsymbol{R}_L$ independently of each other. The displacements are then solved from the global level and the internal variables are computed at the local level with the newly computed displacement from $\boldsymbol{R}_G = 0$. In the interpretation (3.6), the displacements together with the internal variables will be approximated with the DIRK-method, which makes it possible to get an error estimation for the entire system.

3.2 Newton Raphson Method

The general Newton Raphson method will now be explained. Consider a general function given in the format $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{0}$ from $\mathbb{R}^m \to \mathbb{R}^m$. A Taylor expansion is made around $\boldsymbol{x}^k$ and all the higher order derivatives(higher than 1) are left out from the approximation

$$\boldsymbol{F}(\boldsymbol{x}) \approx \boldsymbol{F}(\boldsymbol{x}^k) + \frac{d\boldsymbol{F}(\boldsymbol{x})}{d\boldsymbol{x}}|_{\boldsymbol{x}=\boldsymbol{x}^k}(\boldsymbol{x} - \boldsymbol{x}^k) = \boldsymbol{0} \tag{3.9}$$

$\frac{d\boldsymbol{F}}{d\boldsymbol{x}}$ is called the Jacobian matrix. Solving the above equation for $\boldsymbol{x}$ gives

$$\boldsymbol{x} = \boldsymbol{x}^k - \left(\frac{d\boldsymbol{F}}{d\boldsymbol{x}}|_{\boldsymbol{x}=\boldsymbol{x}^k}\right)^{-1} \boldsymbol{F}(\boldsymbol{x}^k) \tag{3.10}$$

An iteration scheme can now be constructed by putting $\boldsymbol{x} = \boldsymbol{x}^{k+1}$, where k denotes the iteration step. The start guesses have to be given at $k = 0$ and then the next $\boldsymbol{x}^{k+1}$ is calculated until the norm of the difference between $\boldsymbol{x}^{k+1}$ and $\boldsymbol{x}^k$ is less than a predefined tolerance value, i.e $||\boldsymbol{x}^{k+1} - \boldsymbol{x}^k|| < TOL$, where $||.||$ denotes some norm.

3.2.1 Multi Level Newton Raphson

The DAE-system that was established in (3.6) is split into two parts, one global system of residual equations and one local system of residual equations:

$$\begin{aligned} \boldsymbol{R_G}(\boldsymbol{u}(t), \boldsymbol{q}(t)) = \boldsymbol{0} \\ \boldsymbol{R_L}(\boldsymbol{u}(t), \boldsymbol{q}(t)) = \boldsymbol{0} \end{aligned} \tag{3.11}$$

The system (3.11) has already been discretized in space. The system will be approximated in time with a Diagonally Implicit Runge Kutta method and then solved iteratively with the so called Multi Level Newton Raphson Algorithm. A description of the time discretization is made in the next chapter and in this chapter we concentrate on the MLNRA. We assume that the discretization in time for both the displacements and the internal variables have already been made in the following text in this section.

First we have to apply the implicit function theorem, which is stated in the appendix. The use of this theorem will tell us that the internal variables can be written as a function of the displacements $\boldsymbol{u}$ i.e. $\boldsymbol{q} = \boldsymbol{q(u)}$ if we are close to the solution $\boldsymbol{R_L}(\boldsymbol{u}(t), \boldsymbol{q}(t)) = 0$ and $\boldsymbol{R_L}$ is sufficiently smooth (c.f [3]). The system is solved at the 'global' level in order to get the increment of the nodal displacements $\boldsymbol{a}^{k+1} - \boldsymbol{a}^k = \Delta\boldsymbol{a}$, which is then used to get the displacements $\boldsymbol{u}$ according to (3.1). The Newton Raphson iteration scheme (3.10) applied to $\boldsymbol{R_G}$ together with the implicit function theorem gives

$$\boldsymbol{K}_T\Delta\boldsymbol{a} = -\boldsymbol{R_G}(\boldsymbol{u}^k, \boldsymbol{q}(\boldsymbol{u^k})) \tag{3.12}$$

where the tangent stiffness matrix $\boldsymbol{K}_T$ is defined as, i.e

$$\boldsymbol{K}_T = \frac{d\boldsymbol{R_G}}{d\boldsymbol{u}}|_{\boldsymbol{u}^k} + \frac{d\boldsymbol{R_G}}{d\boldsymbol{q}}\frac{d\boldsymbol{q}}{d\boldsymbol{u}}|_{\boldsymbol{u}^k} \tag{3.13}$$

By performing the differentiation on $\boldsymbol{R}_G$ one arrives with the following expression on the stiffness matrix

$$\boldsymbol{K}_T = \int_{V^0} \boldsymbol{B}^T\boldsymbol{D}\boldsymbol{B}dV^0 + \int_{v^0} \frac{d\boldsymbol{B}^T}{d\boldsymbol{u}}\boldsymbol{S}dV^0 \tag{3.14}$$

where $\boldsymbol{D}$ is the consistent tangent stiffness matrix which is obtained by differentiating the stresses. The derivation of the consistent stiffness matrix (both for the large- and small strain models) are made in the appendix. The increment displacements are then determined through

$$\begin{aligned} \Delta\boldsymbol{a} &= \boldsymbol{K}_T^{-1}\boldsymbol{R}_G(\boldsymbol{u}^k, \boldsymbol{q}(\boldsymbol{u}^k)) \\ \boldsymbol{a}^{k+1} &= \boldsymbol{a}^k + \Delta\boldsymbol{a} \end{aligned} \tag{3.15}$$

In order to find the the displacements at $k+1$ we need $\boldsymbol{q}(\boldsymbol{u}^k)$ which is obtained by solving the local system of residual equations:

$$\boldsymbol{R_L}(\boldsymbol{u}^k, \boldsymbol{q}(\boldsymbol{u}^k)) = \boldsymbol{A}\dot{\boldsymbol{q}} - \boldsymbol{L}(\boldsymbol{u}^k, \boldsymbol{q}(\boldsymbol{u}^k)) = \boldsymbol{0} \tag{3.16}$$

Where $\dot{\boldsymbol{q}}$ has been approximated in time with some time stepping method in order to find $\boldsymbol{q}(\boldsymbol{u}^k)$. The implemented time stepping method is the DIRK approximation, which is described next.

3.3 Runge Kutta

Let us first begin by introducing the Runge-Kutta method for a general problem. Most of the theory in this section can be found in [9] and [3]. The Runge-Kutta method is based on an integration scheme called the quadrature rule. The exact solution of the ordinary differential equation (ODE)

$$y' = f(t), \quad y(t_0) = y_0 \tag{3.17}$$

whose right hand side is independent of y, is given by

$$y = y_0 + \int_{t_0}^{t} f(\tau) d\tau$$

There exists a very rich theory and powerful methods to compute integrals numerically. It is thus natural to utilize this theory in the numerical solution to ODEs. Consider the integral:

$$I = \int_{t_0}^{t} f(\tau) d\tau \tag{3.18}$$

The integral is approximated by a quadrature through evaluating the function value at some selected point τ_i and multiplying them by a weight w_i:

$$I \approx \sum_{i=1}^{s} f(\tau_i) w_i \tag{3.19}$$

Consider the differential equation:

$$\boldsymbol{y}' = \boldsymbol{f}(t, \boldsymbol{y}), \quad \boldsymbol{y}(t_0) = \boldsymbol{y}_0, \quad \boldsymbol{y} \in \mathbb{R}^m, t \in [t_0, T] \tag{3.20}$$

The time interval is split into $N + 1$ subintervals $t_0 < t_1 ... < t_n < t_{n+1} < ... < t_{N-1} < t_N$, which are called time steps. The step sizes are given by: $\Delta t_n = t_{n+1} - t_n$. We assume that the solution $\boldsymbol{y}_n$ is given at the time step t_n and then we are interested of the solution at t_{n+1}. An exact integration of equation (3.20) and a change of integration variable gives

$$\begin{aligned} \boldsymbol{y}(t_{n+1}) \quad &= \quad \boldsymbol{y}(t_n) + \int_{t_n}^{t_{n+1}} \boldsymbol{f}(t, \boldsymbol{y}(t)) dt = \\ & \quad\quad \boldsymbol{y}(t_n) + \Delta t_n \int_0^1 \boldsymbol{f}(t_n + \tau\Delta t, \boldsymbol{y}(t_n + \tau\Delta t)) d\tau \end{aligned} \tag{3.21}$$

Now a quadrature rule is applied to (3.21), which results in

$$\boldsymbol{y}(t_{n+1}) \approx \boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \Delta t_n \sum_{i=1}^{s} b_i \boldsymbol{f}(t_n + c_i \Delta t, \boldsymbol{y}(t_n + c_i \Delta t)) \tag{3.22}$$

where the weight factors b_i and the coefficients c_i that determines where and when the functions are evaluated have been introduced. The time points $t_{ni} = t_n + c_i \Delta t_n$ are denoted to be at load step n and stage i, where the total number of stages are s. From the expression above we see that there still exists unknowns at the stages, i.e the stage values $\boldsymbol{y}(t_n + c_i \Delta t_n)$. But this can in the same way as with (3.22) be approximated with a quadrature rule

$$\boldsymbol{y}(t_n + c_i \Delta t_n) \approx \boldsymbol{Y}_{ni} = \boldsymbol{y}(t_n) + \Delta t_n \sum_{j=1}^{s} a_{ij} \boldsymbol{f}(t_n + c_j \Delta t_n, \boldsymbol{Y}_{ni}) \tag{3.23}$$

With the new weighting factors a_{ij} but at the same time points $t_{ni} = t_n + c_i \Delta t$. For convenience, the terms inside the sum in (3.23) are defined as

$$\dot{\boldsymbol{Y}}_{ni} = \boldsymbol{f}(t_n + c_j \Delta t_n, \boldsymbol{Y}_{ni}) \tag{3.24}$$

and is referred to as the to be the stage derivatives. Now there are two sets of unknowns $\boldsymbol{Y}_{ni}$ and $\dot{\boldsymbol{Y}}_{ni}$, but they are however coupled through (3.23) and thus it is possible to choose either $\boldsymbol{Y}_{ni}$ or $\dot{\boldsymbol{Y}}_{ni}$ as our primary set of unknowns. The Runge-Kutta method can further be divided into two groups, explicit and implicit Runga-Kutta methods, which are given by:

$$\boldsymbol{Y}_{ni} = \boldsymbol{y}_n + \Delta t_n \sum_{j=1}^{s} a_{ij} \dot{\boldsymbol{Y}}_{nj}, \quad \text{Implicit Method} \tag{3.25}$$

$$\boldsymbol{Y}_{ni} = \boldsymbol{y}_n + \Delta t_n \sum_{j=1}^{i-1} a_{ij} \dot{\boldsymbol{Y}}_{nj}, \quad \text{Explicit Method} \tag{3.26}$$

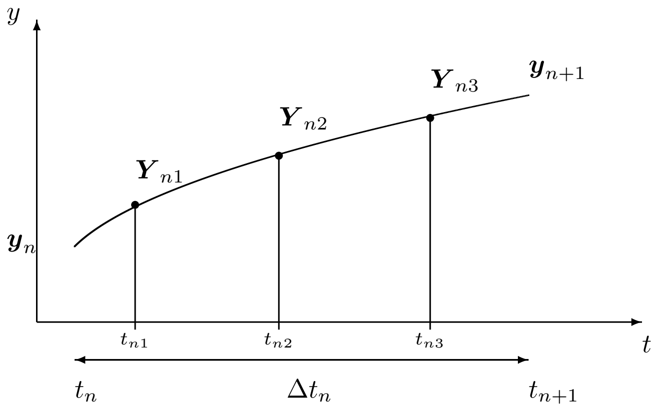


Figure 3.1: A three stage method

where at least one $a_{ij} \neq 0$ for $j \geq i$ in the implicit RK case. Consider figure 3.1. The interval $[t_n, t_{n+1}]$ has been split into the stages $(\boldsymbol{Y}_{n1}, \boldsymbol{Y}_{n2}, \boldsymbol{Y}_{n3})$ at the time points (t_{n1}, t_{n2}, t_{n3}) in the curve.

In case of an explicit method with $j < i$, the stage values $\boldsymbol{Y}_{ni}$ can be calculated from the values of the previous stages $\boldsymbol{Y}_{nj}$. The stage derivatives can thus be determined explicitly. For example, the stage values in the classical RK method are determined according to

$$\begin{aligned}
\boldsymbol{Y}_{n1} &= \boldsymbol{y}_n \\
\boldsymbol{Y}_{n2} &= \boldsymbol{y}_n + \frac{\Delta t_n}{2}\boldsymbol{Y}_{n1} \\
\boldsymbol{Y}_{n3} &= \boldsymbol{y}_n + \Delta t_n(-\boldsymbol{Y}_{n1} + 2\boldsymbol{Y}_{n2})
\end{aligned}$$

And then the value at $\boldsymbol{y}_{n+1}$ in the classical RK method is given by

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \Delta t_n \left(\frac{1}{6}\dot{\boldsymbol{Y}}_{n1} + \frac{2}{3}\dot{\boldsymbol{Y}}_{n2} + \frac{1}{6}\dot{\boldsymbol{Y}}_{n3}\right) \tag{3.27}$$

If an implicit method is being considered, then a system of non-linear equations has be to solved in order to determine the stage derivatives $\dot{\boldsymbol{Y}}_{ni}$. A general method to do this is as follows. Choose $\dot{\boldsymbol{Y}}_{ni}$ as the primary set of unknowns and then define a residual as (see (3.24))

$$\boldsymbol{r}_Y(\dot{\boldsymbol{Y}}_{n1}, \dot{\boldsymbol{Y}}_{n2}, ..., \dot{\boldsymbol{Y}}_{ns}) = \begin{pmatrix} \dot{\boldsymbol{Y}}_{n1} - \boldsymbol{f}(t_n + c_1\Delta t_n, \boldsymbol{y}_n + \Delta t_n \sum_{j=1}^{s} a_{1j}\dot{\boldsymbol{Y}}_{nj}) \\ \dot{\boldsymbol{Y}}_{n2} - \boldsymbol{f}(t_n + c_2\Delta t_n, \boldsymbol{y}_n + \Delta t_n \sum_{j=1}^{s} a_{2j}\dot{\boldsymbol{Y}}_{nj}) \\ \vdots \\ \dot{\boldsymbol{Y}}_{ns} - \boldsymbol{f}(t_n + c_s\Delta t_n, \boldsymbol{y}_n + \Delta t_n \sum_{j=1}^{s} a_{sj}\dot{\boldsymbol{Y}}_{nj}) \end{pmatrix} \tag{3.28}$$

Then solve the system $\boldsymbol{r}_Y = \boldsymbol{0}$ in order to get the stage derivatives (with for example the Newton Raphson method described in the previous chapter). Once the stage derivatives have been calculated, then the new step can be computed to be (according to (3.22)).

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \Delta t_n \sum_{i=1}^{s} b_i \dot{\boldsymbol{Y}}_{ni} \tag{3.29}$$

We will however consider a special family of implicit Runge-Kutta methods that simplify the stage calculations significantly.

3.3.1 DIRK-method

The coefficients a_{ij}, c_i and b_i are usually ordered into a so called Butcher tableau as shown in the table below.

$$\begin{array}{c|c} \boldsymbol{c} & \boldsymbol{a} \\ \hline & \boldsymbol{b}^T \end{array}$$

A Runge Kutta method can either be explicit or implicit as stated previously. The butcher tableau of the two different kinds are shown below.

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \dots & a_{1s} \\ c_2 & a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \end{array}$$

(a) Butcher tableau for Implicit RK method

$$\begin{array}{c|cccc} 0 & 0 & 0 & \dots & 0 \\ c_2 & a_{21} & 0 & \dots & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & 0 \\ \hline & b_1 & b_2 & \dots & b_s \end{array}$$

(b) Butcher tableau for Explicit RK method

The coefficients in the butcher tableau determines the efficiency, stability and accuracy of the methods. An explicit Runge-Kutta method has its upper right part of the $\boldsymbol{a}$ matrix filled with zeros, including the diagonal. While in general an implicit method has the whole matrix filled with non-zero values. The classical RK method that was considered previously has the butcher tableau according to table 3.1.

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ 1 & -1 & 2 & \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

Table 3.1: Classical RK method

A special family of stiffly accurate RK methods called Diagonally Implicit Runge Kutta methods are now of special interest in a finite element context. A DIRK method is very similar to the Implicit Euler scheme, which implies that old source code where the Implicit Euler scheme has been used can be easily modified to incorporate the DIRK method instead. A DIRK method has a butcher tableau according to table 3.2.

$$\begin{array}{c|cccc}
c_1 & a_{11} & 0 & \dots & 0 \\
c_2 & a_{21} & a_{22} & \dots & 0 \\
\vdots & \vdots & & \ddots & \vdots \\
c_s & b_1 & b_2 & \dots & b_s \\
\hline
 & b_1 & b_2 & \dots & b_s
\end{array}$$

Table 3.2: Diagonally Implicit Runge-Kutta method

A fully implicit method is of little interest when applied to finite element problems since the sparse structure of the stiffness matrix will be destroyed. Further, a stiffly accurate method is preferred, which is achieved by letting the last stage $\boldsymbol{Y}_{ns}$ coincide with the new solution $\boldsymbol{y}_{n+1}$. This property guarantees that the algebraic constrains is fulfilled at the last step (cf.[3]). A DIRK method that can handle stiff problems are the Singly Diagonally Implicit Runge Kutta (SDIRK) method, which have the properties $a_{si} = b_i$ and $a_{ij} = 0$ when $i < j$. The calculation of the stages (see (3.25)) can be simplified to

$$\boldsymbol{Y}_{ni} = \boldsymbol{y}_n + \Delta t_n \sum_{j=1}^{i} a_{ij}\dot{\boldsymbol{Y}}_{ni} = \boldsymbol{S}_{ni} + \Delta t_n a_{ii}\dot{\boldsymbol{Y}}_{ni} \tag{3.30}$$

where the "starting value" has been defined according to:

$$\boldsymbol{S}_{ni} = \boldsymbol{y}_n + \Delta t_n \sum_{j=1}^{i-1} a_{ij}\dot{\boldsymbol{Y}}_{ni} \tag{3.31}$$

Note that $\boldsymbol{S}_{ni}$ only depends on the previous calculated stage derivatives i.e $j < i$ and is therefore known at stage i. The integration (3.30) is essential for our further analysis. Let us compare the DIRK step (3.30) with an Implicit Euler step

$$\begin{aligned}
\boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + \Delta t_n \boldsymbol{f}(t_n, \boldsymbol{y}_{n+1}) \\
\boldsymbol{Y}_{ni} &= \boldsymbol{S}_{ni} + \Delta t_n a_{ii} \boldsymbol{f}(t_n + c_i\Delta t_n, \boldsymbol{Y}_{ni})
\end{aligned} \tag{3.32}$$

The structure between the two equations are very similar. The difference between them is that $\boldsymbol{y}_n$ is replaced with $\boldsymbol{S}_{ni}$ and Δt_n is replaced with $a_{ii}\Delta t_n$ at the stage calculation. In order to solve the system for $\boldsymbol{Y}_{ni}$ at stage t_{ni} in a general way, a similar procedure as shown for the general implicit RK method in chapter 3.1 can be utilized. Begin with switching to $\boldsymbol{Y}_{ni}$ to be the primary set of unknowns through

$$\dot{\boldsymbol{Y}}_{ni} = \frac{\boldsymbol{Y}_{ni} - \boldsymbol{S}_{ni}}{\Delta t_n a_{ii}} \tag{3.33}$$

Note again the resemblance to the IE approximation of the derivative $\dot{\boldsymbol{y}} \approx \frac{\boldsymbol{y}(t_{n+1}) - \boldsymbol{y}(t_n)}{\Delta t_n}$. A residual at stage i can then be formed which only has $\boldsymbol{Y}_{ni}$ as the primary set of unknowns:

$$\boldsymbol{r}_{ni}(\boldsymbol{Y}_{ni}) = \frac{\boldsymbol{Y}_{ni} - \boldsymbol{S}_{ni}}{\Delta t_n a_{ii}} - \boldsymbol{f}(t_n + c_j \Delta t_n, \boldsymbol{Y}_{ni}) = \mathbf{0} \tag{3.34}$$

This residual equation can then be solved with for example the Newton Raphson method that was explained previously.

3.3.2 Error Control

A motivation for using a DIRK method is not only to obtain an integration method with higher accuracy but also that a suitable time step Δt_n can be determined with virtually no extra cost. Assume that $\boldsymbol{y}_{n+1}$ is the solution at t_{n+1} from a RK-method with order q and that $\hat{\boldsymbol{y}}_{n+1}$ is the solution from a RK-method with order p and $p = q + 1$. See the appendix for the definition of the order of a method. We have:

$$\boldsymbol{y}_{n+1} = \bar{\boldsymbol{y}}(t_{n+1}) + \boldsymbol{l}\Delta t_n^{q+1} + \mathcal{O}(\Delta t_n^{q+2}) \tag{3.35}$$

$$\hat{\boldsymbol{y}}_{n+1} = \bar{\boldsymbol{y}}(t_{n+1}) + \mathcal{O}(\Delta t_n^{q+2}) \tag{3.36}$$

$\bar{\boldsymbol{y}}(t_{n+1})$ is the exact solution with initial condition $\boldsymbol{y}(t_n) = \boldsymbol{y}_n$ and $\boldsymbol{l}$ is a vector which depends on the underlying differential equation but not upon Δt_n. Subtracting these two solutions with each other we obtain

$$\boldsymbol{y}_{n+1} - \hat{\boldsymbol{y}}_{n+1} \approx \boldsymbol{l}\Delta t_n^{q+1} \tag{3.37}$$

which is the main part of the local integration error. Now we want the error to fulfill

$$||\boldsymbol{\kappa}|| = ||\boldsymbol{y}_{n+1} - \hat{\boldsymbol{y}}_{n+1}|| < TOL \quad \text{,where } TOL = \varepsilon_r|\boldsymbol{y}_n| + \varepsilon_a \tag{3.38}$$

ε_r is a predefined relative tolerance and ε_a is a predefined absolute tolerance. The norm of the error is defined to be

$$||\boldsymbol{\kappa}|| = ||\boldsymbol{l}\Delta t_n^{q+1}|| = C\Delta t_n^{q+1} \tag{3.39}$$

where C is a constant independent of Δt_n. In order to determine the new step size we require that the local integration error should be equal to the predefined tolerance

$$C\Delta t_{new}^{q+1} = TOL \tag{3.40}$$

By solving this equation for C and substituting it into (3.39) one arrives at the following estimate for the new step size

$$\Delta t_{new} = \Delta t_n \left(\frac{TOL}{||\kappa||}\right)^{\frac{1}{q+1}} \tag{3.41}$$

This is the estimation of the step size that will be made in the implementation. The calculation of the new step size Δt_{new} is most efficiently done by an embedded DIRK method.

3.3.3 Embedded DIRK method

By using an embedded DIRK method the new time step, Δt_{new} can be calculated at a very low cost. Suppose again that we have two RK-methods, one with order p and the other one with order $p-1$, which gives the solution $\boldsymbol{y}_{n+1}$ and $\hat{\boldsymbol{y}}_{n+1}$ respectively. If the two methods have the same coefficients in the $\boldsymbol{a}$ matrix and $\boldsymbol{c}$ vector in the butcher tablau, then the two methods are said to be embedded. A consequence of an embedded method is that its enough to solve the system $\boldsymbol{r}_{ni}(\boldsymbol{Y}_{ni})$ once since both methods will provide the same stage values $\boldsymbol{Y}_{ni}$, in order to caluclate $\boldsymbol{y}_{n+1}$. The solution for the RK-methods are given by (3.29):

$$\begin{array}{rcl} \hat{\boldsymbol{y}}_{n+1} & = & \boldsymbol{y}_n + \Delta t_n \sum_{i=1}^{s} \hat{b}_i \dot{\boldsymbol{Y}}_{ni} \\ \boldsymbol{y}_{n+1} & = & \boldsymbol{y}_n + \Delta t_n \sum_{i=1}^{s} b_i \dot{\boldsymbol{Y}}_{ni} \end{array} \tag{3.42}$$

Thus an error estimation from (3.39) can be made which is

$$\boldsymbol{\kappa} = \hat{\boldsymbol{y}}_{n+1} - \boldsymbol{y}_{n+1} = \Delta t_n \sum_{i=1}^{s} (b_i - \hat{b}_i) \dot{\boldsymbol{Y}}_{ni} \tag{3.43}$$

For our particular DAE system we have that the solution $\boldsymbol{y}_{n+1}$ is split into the displacements $\boldsymbol{u}_{n+1}$ and the internal variables $\boldsymbol{q}_{n+1}$. An error measure for the displacements and internal variables are defined to be

$$e_u = \sqrt{\frac{1}{n_u} \sum_{l=1}^{n_u} \left(\frac{\kappa_u}{\varepsilon_r |u_n^l| + \varepsilon_a^l} \right)^2} \qquad e_q^k = \max_k \left| \frac{\kappa_q^k}{\varepsilon_r |q_n^l| + \varepsilon_a^l} \right| \tag{3.44}$$

n_u is the number of degrees of freedom and k represents the different internal variables. The same error measurement that can be found in [3] has been used. The new step size is then calculated according to (3.41) with these error measures and some additional safety factors:

$$\Delta t_{new} = \Delta t_n \cdot \left\{ \begin{array}{ll} \max(f_{min}, f_{safety} \cdot e_m^{-1/(q+1)}) & \text{if } e_m > 1 \\ \min(f_{max}, f_{safety} \cdot e_m^{-1/(q+1)}) & \text{if } e_m \leq 1 \end{array} \right) \tag{3.45}$$

e_m is the maximum of the error measures $e_m = \max(e_u, e_q^k)$. The factor f_{safety} prevents to much oscillations in the step sizes, f_{min} and f_{max} damp extreme step size changes. See reference ([6], [5] and [3] for details).

Chapter 4

Procedure

4.1 Solution procedure

Now when we have the necessary tools for solving (3.6), the solution procedure is presented. The system will be solved at the stages $\boldsymbol{Y}_{ni}$ using the Multi Level Newton Raphson method together with the DIRK method. The stage values at the time $t_{ni} = t_n + c_i \Delta t_n$ are given by

$$\boldsymbol{Y}_{ni} = \begin{pmatrix} \boldsymbol{u}_{ni} \\ \boldsymbol{q}_{ni} \end{pmatrix} \tag{4.1}$$

Assume that equilibrium at load step n exists and we know the quantities $\boldsymbol{y}_n = (\boldsymbol{u}_n, \boldsymbol{q}_n)^T$ and now we want to find the solution $\boldsymbol{y}_{n+1}$. The solution scheme for a 2-stage SDIRK method will be used to illustrate a displacement driven format applied to a bar. Consider figure (4.1).

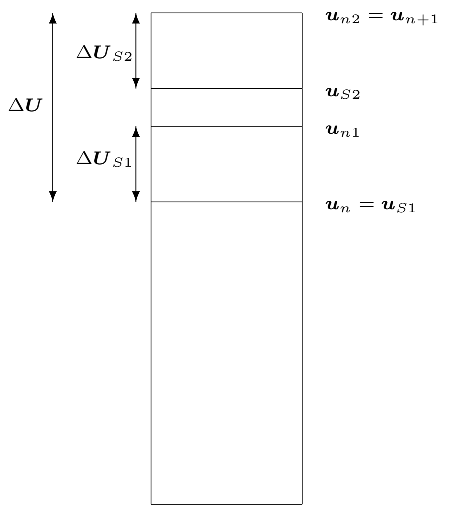

$\Delta \boldsymbol{U}$
$\Delta \boldsymbol{U}_{S2}$
$\Delta \boldsymbol{U}_{S1}$
$\boldsymbol{u}_{n2} = \boldsymbol{u}_{n+1}$
$\boldsymbol{u}_{S2}$
$\boldsymbol{u}_{n1}$
$\boldsymbol{u}_{n} = \boldsymbol{u}_{S1}$

Figure 4.1: Illustration of a rectangular bar undergoing deformation from load step n to $n+1$

STAGE $i = 1$:
Assume that the bar in figure (4.1) has fixed boundary conditions at the left, right and bottom side while the bar will be pulled at the top with a constant speed $\dot{u} = \frac{\Delta u}{\Delta t}$. The total displacement during the entire load step of the bar will then be given by:

$$\Delta U = \dot{u} \Delta t_n \tag{4.2}$$

First the start values S are calculated, which at stage 1 is equal to the solution at load step n, i.e.

$$\boldsymbol{S}_{n1} = \begin{pmatrix} \boldsymbol{u}_S \\ \boldsymbol{q}_S \end{pmatrix} = \begin{pmatrix} \boldsymbol{u}_n \\ \boldsymbol{q}_n \end{pmatrix} \tag{4.3}$$

Then the increment displacements from n to $\boldsymbol{Y}_{n1}$ at $t_{n1} = t_n + c_1 \Delta t_n$ is computed to be

$$\Delta U_{S1} = c_1 \Delta U \tag{4.4}$$

The stage values $\boldsymbol{Y}_{n1}$ can then be found with the help of the Multi Level Newton Raphson algorithm together with the DIRK-approximation

$$\boldsymbol{F}(t_{n1},\boldsymbol{Y}_{n1},\frac{\boldsymbol{Y}_{n1}-\boldsymbol{S}_{n1}}{a_{ii}\Delta t_n}) = \left(\begin{array}{lcl} \boldsymbol{R_G}(\boldsymbol{Y}_{n1}) & = & \int_V \boldsymbol{B}^T\boldsymbol{S}(\boldsymbol{Y}_{n1})dV - \boldsymbol{f}^{ext} \\ \boldsymbol{R_L}(t_{ni},\boldsymbol{Y}_{n1}) & = & \boldsymbol{A}\left(\frac{\boldsymbol{q}_{n1}-\boldsymbol{q}_S}{a_{ii}\Delta t_n}\right) - \boldsymbol{L}(\boldsymbol{u}_{n1},\boldsymbol{q}_{n1}) \end{array}\right) = \left(\begin{array}{c} \boldsymbol{0} \\ \boldsymbol{0} \end{array}\right) \tag{4.5}$$

is solved. See box (4.2). The stage derivatives are stored next, which is needed for the new start value at the next stage according to (3.33).

$$\dot{\boldsymbol{Y}}_{n1} = \frac{\boldsymbol{Y}_{n1} - \boldsymbol{y}_n}{\Delta t a_{11}} \tag{4.6}$$

Stage $i = 2$:

The start value is computed to be

$$\boldsymbol{S}_{n2} = \boldsymbol{y}_n + \sum_{j=1}^{i-1} \Delta t_n a_{2j} \dot{\boldsymbol{Y}}_{nj} = \boldsymbol{y}_n + a_{21}\Delta\dot{\boldsymbol{Y}}_{n1} \tag{4.7}$$

Then the increment displacement has to be calculated. Note that the start value for the displacement in order to calculate the solution at stage 2 is not given by displacements at stage 1, i.e. $\boldsymbol{u}_{S2} \neq \boldsymbol{u}_{n1}$. Let us define the displacement increment from n to stage $\boldsymbol{Y}_{n2}$ by $\Delta U_2 = c_2\Delta U$. Since we consider a stiffly accurate 2 stage Runge-Kutta method, which has the property $\boldsymbol{y}_{n+1} = \boldsymbol{Y}_{n2}$ it follows that $c_2 = 1$ and thus $\Delta U_2 = \Delta U$. But this however not the increment displacement that will be loaded upon the bar, since the displacement are added from the start value $\boldsymbol{u}_{S2}$. From the figure it follows that the displacement $\Delta\boldsymbol{U}$ can be written as

$$\Delta U = (u_{S2} - u_n) + \Delta U_{S2} \tag{4.8}$$

Putting in (4.7) and using the fact that we have a constant speed of the increment of displacements, i.e $\dot{u}_{n1} = \frac{\Delta U}{\Delta t_n}$ and solving for ΔU_{S2} gives the sought increment displacement to be

$$\Delta U_{S2} = c_2\Delta U - \sum_{j=1}^{i-1} \Delta t_n a_{2j}\frac{\Delta U}{\Delta t_n} = \Delta U - a_{21}\Delta U \tag{4.9}$$

The system is then solved again with the multi level Newton Raphson algorithm to get $\boldsymbol{Y}_{n2}$ and then we put $\boldsymbol{y}_{n+1} = \boldsymbol{Y}_{n2}$.

The solution procedure of solving these systems is summarized in table (4.1) and (4.2). The embedded Elsiepen 2-stage Runge-Kutta method will be used, which is given in table (4.3). It is possible to add a restart box in the updating procedure if the error estimation fulfills $\boldsymbol{e}_m > 1$ and thus get a more accurate result, but at the expense of more global newton iterations. This is done in the algorithms in [3]. This restart box is however not included in the current implementation, since no restart is usually done when solving the system with the Implicit Euler scheme together with a crude time step estimation.

```
Given:
  • Initial conditions $\boldsymbol{y}_0 = \begin{pmatrix} \boldsymbol{a}_0 \\ \boldsymbol{q}_0 \end{pmatrix}$
  • Initial time step $\Delta t_0$

  • DIRK-scheme with $s$ stages, $a_{ij}$, $c_i$, $b_i$ where $1 \leq i \leq s$, $1 \leq j \leq s$,
  While $t_{n+1} < t_{end}$
      For stage, i=1:s
          – * Calculate new time for stage $t_{ni} = t_n + c_i \Delta t_n$
            * Calculate new start value $\boldsymbol{S}_{ni} = \boldsymbol{y}_n + \Delta t_n \sum_{j=1}^{i-1} a_{ij} \dot{\boldsymbol{Y}}_{ni}$
            * Solve system $\boldsymbol{F}(t_{ni}, \boldsymbol{Y}_{ni}, \frac{\boldsymbol{Y}_{ni} - \boldsymbol{S}_{ni}}{a_{ii} \Delta t_n}) = \boldsymbol{0}$ (3.6) with ML-NRA, see table (4.2)
            * Store stage derivatives $\dot{\boldsymbol{Y}}_{ni} = \frac{\boldsymbol{Y}_{ni} - \boldsymbol{S}_{ni}}{\Delta t a_{ii}}$ according to (3.33)
          – Estimate error $e_m$ according to (3.44)

          – Estimate new time step according to (3.45)
  • Update, $\boldsymbol{y}_{n+1} = \boldsymbol{Y}_{\boldsymbol{ns}}, t_{n+1} = t_n + \Delta t_n$
```

Table 4.1: Update procedure

```
• Given $\boldsymbol{a}^0 = \boldsymbol{a}^S$ and $\boldsymbol{q}^0 = \boldsymbol{q}^S$

• Determine new load level $\boldsymbol{f}_{ni}^{ext}(t_{ni})$ or increment displacements $\Delta \boldsymbol{U}_{Si}$
  Iterate $k = 1, 2 \ldots$ until $\Phi^k = ||\boldsymbol{f}^{ext} - \boldsymbol{f}^{int}|| < TOL$
      – Calculate stiffness matrix $\boldsymbol{K}_T(\boldsymbol{u}^{k-1}, \boldsymbol{q}^{k-1})$
      – Solve $\Delta \boldsymbol{a}^k$ from $\boldsymbol{K}_T \Delta \boldsymbol{a}^k = \boldsymbol{f}^{ext}$
      – Update nodal displacements $\boldsymbol{a}^k = \boldsymbol{a}^{k-1} + \Delta \boldsymbol{a}^k$
      – Solve the local system $\boldsymbol{R}_L(\boldsymbol{u}^k, \boldsymbol{q}^k) = 0$ for $\boldsymbol{q}^k$. See chapter 4.2 and 4.3.
      – Determine the new stress $\boldsymbol{S}(\boldsymbol{u}^k, \boldsymbol{q}^k)$
      – Determine the new tangential stiffness matrix $\boldsymbol{D}^k$ used in $\boldsymbol{K_T}$
      – compute the new out of balance force $\boldsymbol{R_G}(\boldsymbol{u^k}, \boldsymbol{q}^k)$
• Accept new quantities, $\boldsymbol{a}_{ni} = \boldsymbol{a}^k$, $\boldsymbol{q}_{ni} = \boldsymbol{q}^k$
```

Table 4.2: Multi level Newton Raphson Algorithm

$$
\begin{array}{c|cc}
\alpha & \alpha & \\
1 & 1-\alpha & \alpha \\
\hline
 & 1-\alpha & \alpha \\
 & 1-\hat{\alpha} & \hat{\alpha}
\end{array}
$$

$$\alpha = 1 - \tfrac{1}{2}\sqrt{2}$$
$$\hat{\alpha} = 2 - \tfrac{5}{4}\sqrt{2}$$

Table 4.3: Butcher tableau for Elsiepens method, with $s = 2$ and $q = 1$.

4.2 Integration algorithm small strains

The non-linear evolution equations for small strains that was derived in chapter 2.3 will now be integrated and solved with the DIRK-method. The integration step is calculated from the last accepted equilibrium $\boldsymbol{y}_n$ to the new stage $\boldsymbol{Y}_{ni}$. The integrated variables will be solved by reducing them to two non linear scalar equations that can be solved independently from each other. The derivation of these two non-linear equations will closely follow the solution procedure in ([7]), but no damage variable was incorporated in that article. First we begin by introducing a new variable ζ to be:

$$\zeta = \frac{a_{cc}\Delta t_n}{\eta}\left(\frac{f}{\sigma_0^2}\right)^r \tag{4.10}$$

No superscript is shown for the variables at the current stage while the the variables determined by the start value in(3.31) has a superscript S. The parameter a_{cc} is taken from the butcher tablau at row c and column c where c denotes the number of the current stage. Putting in the yield condition and rearranging gives the following constraint equation

$$\zeta\sigma_0^{2r}\frac{\eta}{a_{cc}\Delta t_n} - \left[\frac{1}{2}\left(\frac{s_{ij}}{1-\alpha} - X_{ij}^{dev}\right)\left(\frac{s_{ij}}{1-\alpha} - X_{ij}^{dev}\right) - \frac{\left(\sigma_y(\varepsilon_{eff}^p)\right)^2}{3}\right]^r = 0 \tag{4.11}$$

Then a so called trial stress step is made (the predictor, compare (2.15)):

$$s_{ij}^{(t)} = (1-\alpha^S)2G(e_{ij} - e_{ij}^{(p,S)}) \tag{4.12}$$

This trial stess will be put into the yield condition and if $f(s_{ij}^{(t)}, X_{ij}^S, \varepsilon_{eff}^{(p,S)}, \alpha^S) \leq 0$ then the stresses are updated as $s_{ij} = s_{ij}^{(t)}$ and the internal variables as: $\varepsilon_p = \varepsilon_p^S, X_{ij} = X_{ij}^S, \varepsilon_{eff}^p = \varepsilon_{eff}^{(p,S)}, \alpha = \alpha^S$. However if the trial stress is outside the yield surface then the variables must be integrated. To this end the system of differential equations(2.17-2.20) are integrated using the DIRK scheme (3.30). This leads to the following set of non-linear equations

$$\begin{aligned}
\varepsilon_{ij}^p &= \varepsilon_{ij}^{(p,S)} + \zeta N_{ij} && (4.13)\\
X_{ij} &= X_{ij}^S + \zeta\left(cN_{ij} - b\sqrt{\frac{2}{3}}X_{ij}\right) && (4.14)\\
\varepsilon_{eff}^p &= \varepsilon_{eff}^{(p,S)} + \zeta\sqrt{\frac{2}{3}} && (4.15)\\
\alpha &= \alpha^S - \zeta\frac{Y}{S_d(1-\alpha)^m} && (4.16)
\end{aligned}$$

This system could be solved by using the Newton Raphson algorithm explained previously but let us instead reduce the system of non-linear equations, such that a better performing algorithm is obtained. The goal is to arrive with two scalar equations with the unknowns ζ and α. Once ζ and α is determined then the rest of the variables can be calculated. The deviatoric part of the stresses given by (2.15) can be rewritten using the plastic strain (4.13)

$$s_{ij} = 2G(1-\alpha)(e_{ij} - e_{ij}^{(p,S)}) - (1-\alpha)\zeta N_{ij} = \frac{1-\alpha}{1-\alpha^S} s_{ij}^{(t)} - 2G(1-\alpha)\zeta N_{ij} \quad (4.17)$$

Rearranging the back stress (4.14) gives:

$$X_{ij} = \mu(X_{ij}^S + \zeta c N_{ij}) \quad (4.18)$$

with $\mu = \frac{1}{1+b\sqrt{\frac{2}{3}}\zeta}$. Let us introduce the following variables for the difference tensor between the deviatoric part of the stress and back stress

$$\frac{s_{ij}}{1-\alpha} - X_{ij}^{dev} = \frac{s_{ij}^{(t)}}{1-\alpha^S} - 2G\zeta N_{ij} - \mu(X_{ij}^{(dev,S)} + c\zeta N_{ij}) = \Xi_{ij} - \gamma N_{ij} \quad (4.19)$$

where the introduced variables are defined as

$$\Xi_{ij} = \frac{s_{ij}^{(t)}}{1-\alpha^S} - \mu X_{ij}^{(S)} \quad (4.20)$$

$$\gamma = \zeta\,(2G + \mu c) \quad (4.21)$$

With the help of these introduced variables, it is possible to further rewrite $\frac{s_{ij}}{1-\alpha} - X_{ij}^{dev}$ additionally. First we remark that $\frac{s_{ij}}{1-\alpha} - X_{ij}^{dev}$ can be written with the definition of the plastic flow N_{ij} according to:

$$\frac{s_{ij}}{1-\alpha} - X_{ij}^{dev} = ||\frac{s_{pq}}{1-\alpha} - X_{pq}^{dev}||\frac{\frac{s_{ij}}{1-\alpha} - X_{ij}^{dev}}{||\frac{s_{pq}}{1-\alpha} - X_{pq}^{dev}||} = ||\frac{s_{pq}}{1-\alpha} - X_{pq}^{dev}||N_{ij} \quad (4.22)$$

If (4.19) is rearranged and with (4.22) substituted, then the following expression can be obtained

$$||\Xi_{ij}|| = ||(\gamma + ||\frac{s_{ij}}{1-\alpha} - X_{ij}||)N_{ij}|| = \gamma + ||\frac{s_{ij}}{1-\alpha} - X_{ij}|| \quad (4.23)$$

Solving the above equation for $||\frac{s_{ij}}{1-\alpha} - X_{ij}||$ and then substituting it into the constraint equation (4.11) gives then the following expression which only has ζ as an unknown.

$$||\frac{s_{ij}^{(t)}}{1-\alpha^S} - \mu(\zeta)X_{ij}^S|| - \gamma(\zeta) - \sqrt{2\sigma_0^2(\frac{\zeta\eta}{a_{cc}\Delta t_n})^{1/r} + \frac{2}{3}\left(\sigma_y(\zeta)\right)^2} = 0 \qquad (4.24)$$

From this scalar equation it is then possible to compute ζ with for example the Newton Raphson method. Using the fact that $N_{ij} = \frac{\Xi_{ij}}{||\Xi_{pq}||}$ gives that the all the internal variables except for α can be computed according to

$$\begin{array}{rcl} \varepsilon_{ij}^p & = & \varepsilon_{ij}^{(p,S)} + \zeta N_{ij} \\ X_{ij} & = & \mu(\zeta)(X_{ij}^S + \zeta c N_{ij}) \\ \varepsilon_{eff}^p & = & \varepsilon_{eff}^{(p,S)} + \zeta\sqrt{\frac{2}{3}} \end{array} \qquad (4.25)$$

Rearranging the integrated evolution equation for α and using that

$$Y = \frac{1}{2}\varepsilon_{ij}^e\sigma_{ij} = \frac{1}{2}(\varepsilon_{ij} - \varepsilon_{ij}^p)D_{ijkl}^{hooke}(\varepsilon_{kl} - \varepsilon_{kl}^p) \qquad (4.26)$$

we end up with the following scalar equation for the damage

$$(\alpha - \alpha^S)(1-\alpha)^m - \frac{\zeta}{2S_d}(\varepsilon_{ij} - \varepsilon_{ij}^p)D_{ijkl}^{hooke}(\varepsilon_{kl} - \varepsilon_{kl}^p) = 0 \qquad (4.27)$$

All the internal variables are now known and the stresses can thus be updated. The whole integration procedure is summarized in the following box.

```
• Calculate the trial stress, $\sigma_{ij}^{(t)}$ according to 4.12.
  if $f(\sigma_{ij}^{(t)}, X_{ij}^S, \varepsilon_{eff}^{(p,S)}, \alpha^S) \leq 0$ then
    – Update according to
        $\sigma_{ij} = \sigma_{ij}^{(t)},\ X_{ij} = X_{ij}^S,\ \varepsilon_{eff}^p = \varepsilon_{eff}^{(p,S)}, \alpha = \alpha^S$
  else
    – Calculate $\zeta$ from 4.24:
        $||\frac{s_{ij}^{(t)}}{1-\alpha^S} - \mu(\zeta)X_{ij}^{(S)}|| - \gamma(\zeta) - \sqrt{2\sigma_0^2(\frac{\zeta\eta}{a_{cc}\Delta t_n})^{1/r} + \frac{2}{3}\left(\sigma_y(\zeta)\right)^2} = 0$
      where $\Xi_{ij}$ is given from (4.20) and $\gamma$ from (4.21)
    – Update the following internal variables with the help of the now
      calculated $\zeta$ according to
        $N_{ij} = \frac{\Xi_{ij}}{||\Xi_{pq}||}$
        $\varepsilon_{eff}^p = \varepsilon_{eff}^S + \sqrt{\frac{2}{3}}\zeta$
        $X_{ij} = \mu(X_{ij}^S + \zeta c N_{ij}$
        $\sigma_{ij} = \sigma_{ij}^{(t)} - 2G\zeta N_{ij}$
    – Compute the damage variable $\alpha$ from (4.27)
        $(\alpha - \alpha^S)(1-\alpha)^m - \frac{\zeta}{S_d}(\varepsilon_{ij} - \varepsilon_{ij}^p)D_{ijkl}^{hooke}(\varepsilon_{kl} - \varepsilon_{kl}^p) = 0$
    – Update the stress given by (4.17), $s_{ij} = \frac{1-\alpha}{1-\alpha^S}s_{ij}^{(t)} - 2G(1-\alpha)N_{ij}$
```

Table 4.4: Integration procedure for small deformation model

4.3 Integration algorithm large strains

4.3.1 Isochoric integration

The set of evolution equations that was established in section 2.4 will now be integrated. The load step is denoted with n and the current stage with i. The start values are again denoted with an superscript S and the updated internal variables at the current unknown stage are left undenoted. The plastic deformation gradient is commonly integrated with an exponential integration algorithm(see [4]), in order to maintain the plastic deformation gradient isochor. However the ingratiation that we want to use is the DIRK-method, which if used directly will not guarantee the property $\det \boldsymbol{F}^p = 1$. This is because the integration is handled in an additive fashion. If applied directly to (2.36) we will get:

$$\boldsymbol{F}^p = \boldsymbol{F}^{(p,S)} + \Delta t_n a_{ii} \lambda \boldsymbol{N}^p \boldsymbol{F}^p \tag{4.28}$$

Solving for $\boldsymbol{F}^p$ gives

$$\boldsymbol{F}^p = (1 - \Delta t_n a_{ii} \lambda \boldsymbol{N}^p)^{-1} \boldsymbol{F}^{(p,S)} \tag{4.29}$$

Even if assume that we have made sure that $\det \boldsymbol{F}^{(p,S)}$ is isochor then the determinant of $\boldsymbol{F}^p$ delivered in (4.29) will in general not satisfy $\det \boldsymbol{F}^p = 1$ at the stages. Here we suggest a solution, which adds an extra constraint equation to our evolution laws which guarantees $\det \boldsymbol{F}^p = 1$. Let us define the volumetric split of the plastic deformation gradient to be

$$\boldsymbol{F}^p = (J^p)^{1/3} \hat{\boldsymbol{F}}^p \tag{4.30}$$

where $J^p = \det \boldsymbol{F}^p$. Differentiating (4.30) gives

$$\dot{\boldsymbol{F}}^p = \frac{1}{3}(J^p)^{-2/3} \dot{J}^p \hat{\boldsymbol{F}}^p + (J^p)^{1/3} \dot{\hat{\boldsymbol{F}}}^p = \boldsymbol{l}^p \boldsymbol{F}^p \tag{4.31}$$

The velocity gradient $\boldsymbol{l}$ that was introduced above is defined from the evolution law (2.36) as:

$$\boldsymbol{l} = \dot{\lambda} \boldsymbol{N}^p \tag{4.32}$$

Using the time derivative of determinant (c.f [2]) which is given by

$$\det \boldsymbol{A}' = \det \boldsymbol{A} \ \mathrm{tr}(\boldsymbol{A}^{-1} \boldsymbol{A}') \tag{4.33}$$

where $\boldsymbol{A}$ is a nonsingular matrix, the following relationship can be obtained:

$$\dot{\hat{\boldsymbol{F}}}^p = \mathrm{dev}(\boldsymbol{l}^p) \hat{\boldsymbol{F}}^p \tag{4.34}$$

Use of this equation together with (4.30) in (4.31) gives an alternative form of the evolution equation for the plastic gradient given in (2.36).

$$\dot{\boldsymbol{F}}^p = \frac{1}{3}\frac{\dot{J}^p}{J^p}\boldsymbol{F}^p + \mathrm{dev}(\boldsymbol{l}^p)\boldsymbol{F}^p \tag{4.35}$$

The term J^p is supposedly to be equal to one due to the incompressibility of the material and thus $\dot{J}^p = 0$. This however not the case if the DIRK method is applied directly as shown previously. An extra penalty variable κ^p is introduced

$$\kappa^p = \frac{1}{3}\frac{\dot{J}^p}{J^p} \tag{4.36}$$

in order to maintain the incompressibility of the plastic deformation gradient. By substituting the term $\frac{1}{3}\frac{\dot{J}^p}{J^p}$ with κ^p in (4.35) our modified evolution equations takes the form

$$\begin{aligned} \dot{\boldsymbol{F}}^p &= \Big(\kappa^p \boldsymbol{I} + \mathrm{dev}(\boldsymbol{l}^p)\Big)\boldsymbol{F}^p && (4.37) \\ J^p(\kappa^p) &= 1 && (4.38) \end{aligned}$$

The extra constraint equation $J^p(\kappa^p) = 1$ will force the determinant to be constant and thus $\dot{J}^p = 0$ which is what we want in (4.35).

Using the same arguments for the kinematic deformation gradient gives the evolution equation:

$$\begin{aligned} \dot{\boldsymbol{F}}^k &= \boldsymbol{F}^k\Big(\kappa^k \boldsymbol{I} + \dot{\lambda}\mathrm{dev}(\boldsymbol{N}^k)\Big) && (4.39) \\ J^k(\kappa^k) &= 1 && (4.40) \end{aligned}$$

In [8] a so called closed projection technique is utilized where an extra Lagrange multiplier is added in order to minimize to minimize the distance to the projected space $\{\vee \boldsymbol{F}^p | \det \boldsymbol{F}^p = 1\}$. The method proposed here is very similar to the closed projection technique given in [8].

4.3.2 Integration of evolution equations

With this modification of the evolution equation, we are now ready to integrate them with the DIRK-method. First, in the same manner as in the small deformation case, a so called trial step is made to check whether there is plastic deformation or not. The elastic trial deformation gradient is defined to be

$$\boldsymbol{F}^{e,tr} = \boldsymbol{F}\boldsymbol{F}^{p-1,S} \tag{4.41}$$

Where $\boldsymbol{F}$ is known since we know the total displacements $\boldsymbol{u}$. In the same manner as in the small strains case we test If $f(\boldsymbol{F}^{e,tr}, \boldsymbol{F}^{k,S}, \alpha^{(S)}) < 0$ then there is not plastic deformation and the internal variables are updated according to $\boldsymbol{F}^{(p)} = \boldsymbol{F}^{p,S}, \boldsymbol{F}^{(k)} = \boldsymbol{F}^{k,S}, \alpha = \alpha^{(S)}$. If however $f(\boldsymbol{F}^{e,tr}, \boldsymbol{F}^{k,(S)}, \alpha^{S}) > 0$ there will be plastic deformation, which we now devote our interest to. A DIRK-integration of (4.37) and (4.39) gives

$$\begin{aligned}
\boldsymbol{F}^{p} &= \boldsymbol{F}^{(p,S)} + \Big(\kappa^{p}\boldsymbol{I} + a_{ii}\Delta t_{n}\lambda \mathrm{dev}(\boldsymbol{N}^{p})\Big)\boldsymbol{F}^{p} &\quad (4.42)\\
\boldsymbol{F}^{k} &= \boldsymbol{F}^{(k,S)} + \boldsymbol{F}^{k}\Big(\kappa^{k}\boldsymbol{I} + a_{ii}\Delta t_{n}\lambda \mathrm{dev}(\boldsymbol{N}^{k})\Big) &\quad (4.43)
\end{aligned}$$

Equations (4.42) and (4.43) can be solved, i.e.

$$\begin{aligned}
\boldsymbol{F}^{p} &= \Big((1-\Delta t a_{ii}\kappa^{p})\boldsymbol{I} - \Delta\lambda \mathrm{dev}(\boldsymbol{N}^{p})\Big)^{-1}\boldsymbol{F}^{p,S} = \boldsymbol{A}^{p}\boldsymbol{F}^{(p,S)} &\quad (4.44)\\
\boldsymbol{F}^{k} &= \boldsymbol{F}^{(k,S)}\Big((1-\Delta t a_{ii}\kappa^{k})\boldsymbol{I} - \Delta\lambda \mathrm{dev}(\boldsymbol{N}^{k})\Big)^{-1} = \boldsymbol{F}^{(k,S)}\boldsymbol{A}^{k} &\quad (4.45)
\end{aligned}$$

where $\Delta\lambda = \Delta t_{n} a_{ii}\lambda$ is defined. The damage evolution equation is integrated in the same manner as in the small deformation case to get:

$$\alpha^{ni} = \alpha^{S} - \Delta\lambda\frac{Y}{S_{d}(1-\alpha)^{m}} \quad (4.46)$$

In order to establish the variables, $\{\boldsymbol{F}^{p}, \boldsymbol{F}^{k}, \alpha, \Delta\lambda\}$ the yield condition $f = 0$ needs be fulfilled and then a system of residual equations can be determined.

$$\left.\begin{aligned}
\boldsymbol{\mathcal{R}}_{\boldsymbol{F}^{p}} &= \boldsymbol{F}^{p} - \boldsymbol{A}^{p}\boldsymbol{F}^{(p,S)} = \boldsymbol{0}\\
\boldsymbol{\mathcal{R}}_{\boldsymbol{F}^{k}} &= \boldsymbol{F}^{k} - \boldsymbol{F}^{(k,S)}\boldsymbol{A}^{k} = \boldsymbol{0}\\
\mathcal{R}_{f} &= f(\boldsymbol{F}^{p}, \boldsymbol{F}^{k}, \Delta\lambda, \alpha, \kappa^{p}, \kappa^{k}, \boldsymbol{C}) = 0\\
\mathcal{R}_{\kappa^{p}} &= \det\left(\boldsymbol{A}^{p}\boldsymbol{F}^{(p,S)}\right) - 1 = 0\\
\mathcal{R}_{\kappa^{k}} &= \det\left(\boldsymbol{F}^{(k,S)}\boldsymbol{A}^{k}\right) - 1 = 0\\
\mathcal{R}_{\alpha} &= \alpha - \alpha^{S} + \Delta\lambda\tfrac{Y}{S_{d}(1-\alpha)^{m}} = 0
\end{aligned}\right. \quad (4.47)$$

This equation system is solved with the Newton Raphson method. The system consists of 22 unknowns and 22 equations, namely $\boldsymbol{F}^{p}, \boldsymbol{F}^{k}, \Delta\lambda, \kappa^{p}, \kappa^{k}, \alpha$. Collecting the internal variables $\boldsymbol{F}^{(p)}, \boldsymbol{F}^{k}, \alpha$ the increment in the plastic multiplier $\Delta\lambda$ and the penalty variables κ^{p}, κ^{k} in a vector $\boldsymbol{Q}$ enable us to write the Newton iteration algorithm as

$$\boldsymbol{Q}^{k+1} = \boldsymbol{Q}^{k} - \left[\frac{\partial \boldsymbol{R_{L}}}{\partial \boldsymbol{Q}}\right]^{-1}\boldsymbol{R_{L}} \quad (4.48)$$

The Jacobian matrix $\boldsymbol{J} = \frac{\partial \boldsymbol{R_L}}{\partial \boldsymbol{Q}}$ can be obtained by straight forward differentiation of the system $\boldsymbol{R_L}$, which requires a considerable amount of algebraic manipulation. However in this work the Jacobian has been established through numerical differentiation of the the residual equation, which is explained in the appendix. After that the residual system has been solved, then the Mandel stress $\boldsymbol{\Sigma}$ and second Piola Kirchoff stress can be determined according to (2.31) and (2.32) and we can solve the DAE system with the procedure explained in chapter 4.1.

Chapter 5

Results

5.1 Results, small strains

5.1.1 Geometry, Rectangular disc with hole

In the following section the performance of the proposed method is evaluated in small deformations in combination with plain strain. The geometry that has been considered is a rectangular disc with a hole in the middle. Only one fourth of the disc is being considered because the symmetric nature of the geometry in the problem, see figure 5.1

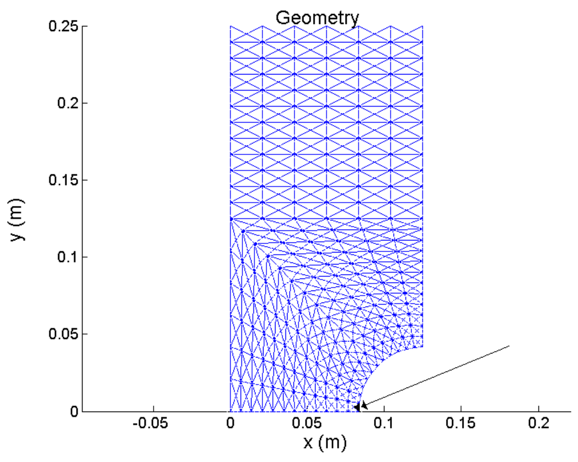


Figure 5.1: Geometry

The radius of the hole is $\frac{500}{12}$ mm, the width is 125 mm and the height is 250 mm. The geometry has been split into 864 triangular elements with a total of 938 degrees of freedom. The disc is pulled at the top in a displacement driven format. There are essential boundary conditions for the degrees of freedom in the y direction at the bottom of the structure and in the x-direction at the right boundary of the structure. These degrees of freedom are put equal to zero at these places. The modulus of elasticity is $E = 2 \cdot 10^5$ MPa and the poisson ratio is $\nu = 0.3$. The thickness of the structure is put to be 2 mm. The damage evolution will be investigated on the element marked in figure (5.1). A reference solution has been made by applying a very small time step to the algorithm in such a way that the if the time step is refined further, then no notable difference can be seen in the internal variables $\boldsymbol{q}$ and the displacements $\boldsymbol{u}$. All the results with the RK method has been run with $f_{max} = 1.3, f_{min} = 0.1, f_{safety} = 0.9$.

5.1.2 Results, viscoplastic model

The hardening function in this model was chosen to

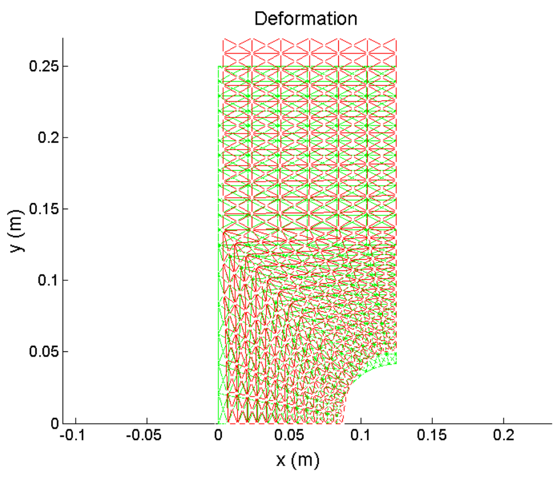
$$\sigma_y = \sigma_{y0} + K_\infty(1 - e^{-\frac{h\varepsilon^p_{eff}}{K_\infty}}) \tag{5.1}$$

Where h and K_∞ are two constitutive parameters. All the constitutive parameters used can be found in table (5.1). The geometry was pulled with a constant speed $\dot{u} = 1$ until the top of the structure had been displaced a total of 0.02 m.

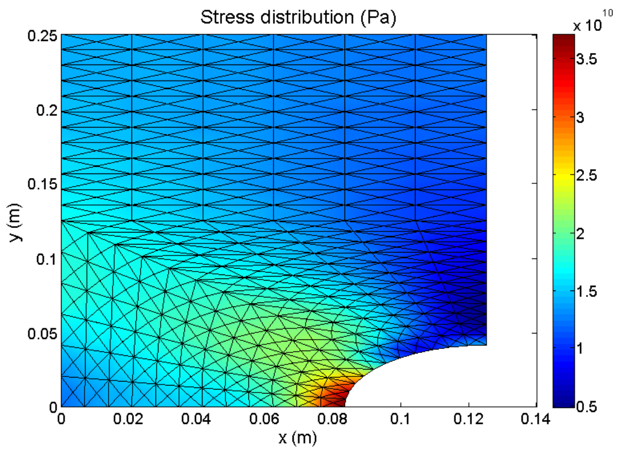
K_∞ (Pa)	h (Pa)	b (-)	c (Pa)	r (-)	σ_{y0} (Pa)	η (s)	m (-)	S (Pa)
$200 \cdot 10^6$	$2 \cdot 10^{10}$	525	$41080 \cdot 10^6$	2	$600 \cdot 10^6$	10^8	2	$4 \cdot 10^6$

Table 5.1: Constants used in this section for internal variables

The deformation and the effective stress is shown in figure (5.2) for illustration purposes.



(a) Deformation



(b) Stress distribution

Figure 5.2: Deformation and stress distribution of solid

The time stepping for the Implicit Euler method has been made with the heuristic rule:

$$\Delta t_{new} = \left\{ \begin{array}{lll} w_{dec}\Delta t_n & \text{if} & n_{newt} \geq n_{high} \\ w_{inc}\Delta t_n & \text{if} & n_{newt} < n_{low} \end{array} \right) \tag{5.2}$$

The time stepping method chosen according to some rule as function of the number of global newton iterations is quite common in the finite element context. The parameters $w_{inc}, w_{dec}, n_{high}, n_{low}$ in (5.2) is chosen by the user and n_{newt} denotes the number of global newton iterations that was made until convergence. The global newton loop has has been set to converge when $||\boldsymbol{f}^{int} - \boldsymbol{f}^{ext}|| < 10^{-6}$ where the euclidean norm has been used. The damage variable has been plotted against the amount of displacement at the top of the structure in figure (5.3), for the element marked with red color in (5.1). The time stepping parameters in figure (5.3) for the Implicit Euler method was chosen according to $w_{inc} = 1.2, w_{dec} = 0.2, n_{high} = 4, n_{low} = 3$ with an initial time step of $\Delta t_0 = 2\cdot10^{-4}$ s. The tolerances for the Runge-Kutta method in figure (5.3) was chosen to $\varepsilon_r = 0$, $\varepsilon^a_u = 10^{-3}, \varepsilon^a_{\varepsilon^p} = 10^{-4}, \varepsilon^a_{\varepsilon^p_{eff}} = 10^{-5}$, $\varepsilon^a_X = 5\cdot10^3, \varepsilon^a_\alpha = 10^{-3}$. The total number of newton iterations was 89 in the RK-method and 104 in the Implicit Euler method.

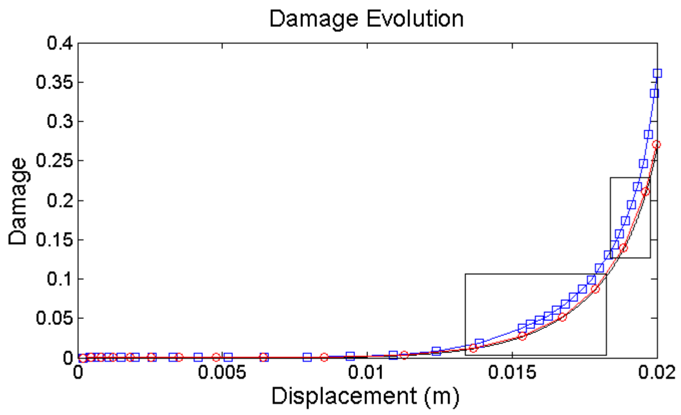


Figure 5.3: Damage Evolution. Curve marked with squares: Implicit Euler, Curve marked with circles: Runge-Kutta, Solid curve: Reference solution

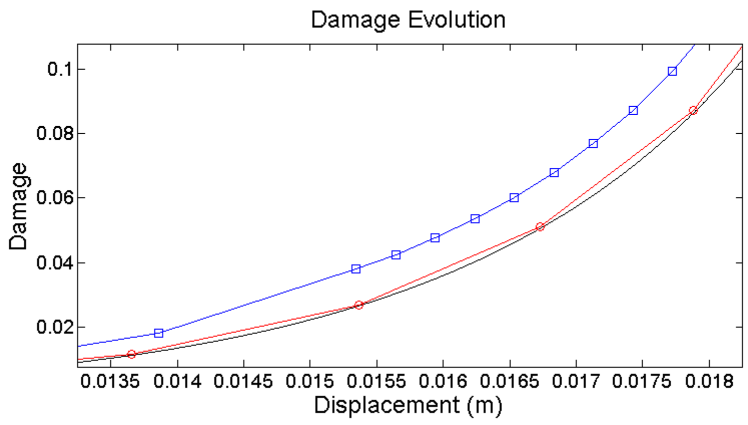


Figure 5.4: Damage Evolution zoomed in around displacement 0.016 m

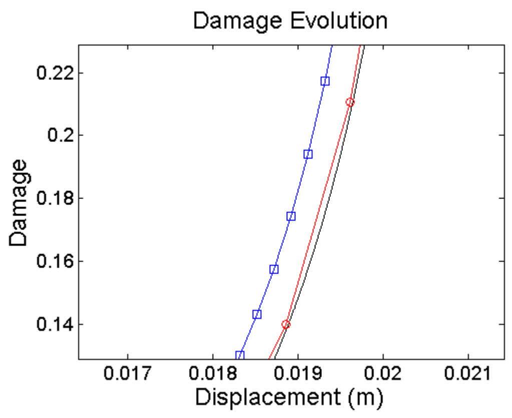


Figure 5.5: Damage Evolution zoomed in around 0.019 m

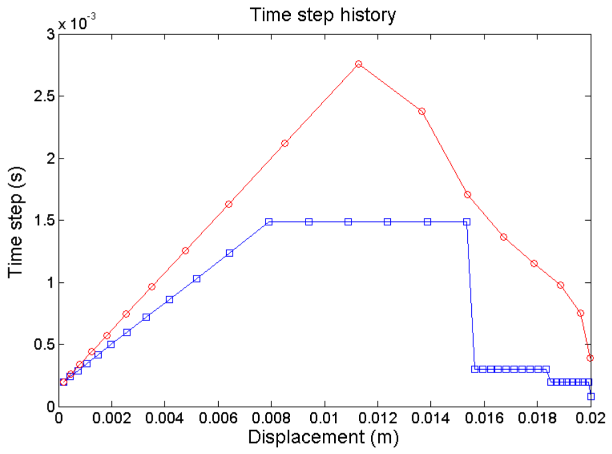


Figure 5.6: Time step history for the curve shown in (5.3)

Point	n_{high}	n_{low}	w_{inc}	w_{dec}
1	4	3	1.3	0.5
2	4	4	1.2	0.1
3	4	3	1.2	0.2
4	3	3	1.5	0.5
5	3	3	1.2	0.2

Table 5.2: IE adaptive parameters. The points are denoted with a squares in (5.7) when looking at the figure from left to right.

A work precision diagram of the method is shown (5.7). The relative error of the damage has been measured on the element marked in (5.1) and was measured after the solid had been displaced 0.02 m. The relative error is defined as $\frac{|\alpha_{ref}-\alpha|}{\alpha_{ref}}$. The parameters in the time stepping procedure for the IE-method that was used to obtain figure (5.7) is shown in table (5.2). For the Runge-Kutta method the parameters were put to $\varepsilon_r = 0$, $\varepsilon_u^a = 10^{-3}, \varepsilon_{\varepsilon^p}^a = 10^{-4}, \varepsilon_{\varepsilon_{eff}^p}^a = 10^{-5}$, $\varepsilon_X^a = 5{\cdot}10^3$ and the absolute tolerance for the damage was varied with $\varepsilon_\alpha^a = 10^{-k}$ with $k = \{2, 3, 4, 5, 6\}$. But the time step was forced to be above $\frac{0.2}{300}$ s at all times in the diagram.

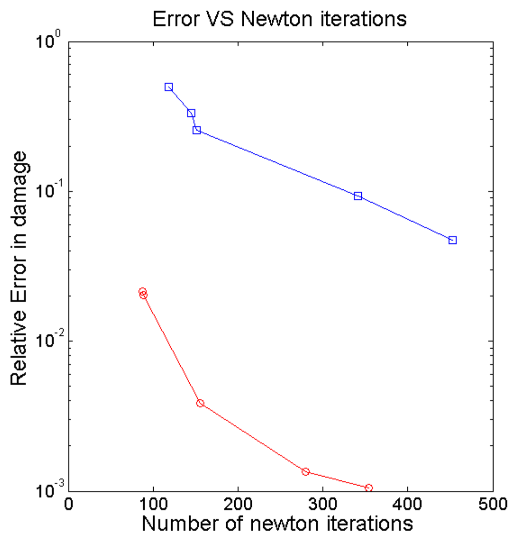


Figure 5.7: Work precision diagram of the methods.Curve marked with squares: Implicit Euler, Curve marked with circles: Runge-Kutta

5.1.3 Discussion

Figure (5.3), which shows the damage evolution is of great interest. The figure illustrates the performance of the Runge-Kutta method compared with the Implicit Euler method clearly. The time steps in (5.6) are quite large in the RK-method compared with the IE-method and still the RK method almost coincide with the reference solution at the given scales. In figure (5.3), the total number of newton iterations for the RK-method was about the same in the IE-method (89 compared with 104) in order to make a suitable comparison between the methods. The Runge-Kutta method seems to be clearly superior over the IE-method.

However one must keep in mind that the time stepping method in the RK-method is made with an predictor/corrector scheme and the method can thus adjust the time step in a much better way than the time stepping in the IE-method. In this sense the comparison is not entirely fair since the IE time stepping method is based upon the number of global newton iterations and thus decreases the time step first after it notices that it has deviated from the true solution. But the time stepping method for the IE-method was chosen in this way because this how it is commonly done. One must also remember that the DAE system has to be solved twice in each time step for the two-stage RK method.

The work precision diagram in figure (5.7) shows how the accuracy varies with the total number of newton iterations for the two different time adaptive methods. It shows that even tough less global newton iteration is made for the RK-method, the RK-method has a much smaller relative error of the damage compared with the Implicit Euler method. It is also notable that it is difficult to control the number of global newton iterations with the IE-method since there are not so many parameters to change in the proposed time-stepping method. In the RK-method it is much easier to control the accuracy since we are given a lot more tolerances which can be used directly to control the integration error.

5.2 Results, Finite deformation

5.2.1 Geometry, Necking of bar

The geometry used in the finite deformation case is shown in figure (5.8).

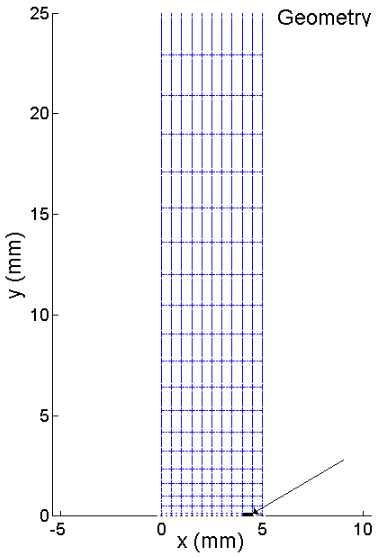


Figure 5.8: Geometry

The solid has been discretized into 200 axisymmetric elements with a mixed finite element formulation. The bar has a radius of 5 mm and a height of 25 mm . Essential boundary conditions exist on the bottom, left and right side of the structure where the corresponding degrees of freedom has been put to zero. The solid is again pulled on the top boundary with a constant speed of $\dot{u} = 1$ mm s^{-1} until the top has been displaced a total of 2 mm. The material parameters used in this particular problem is given in table (5.3)

σ_{y0}	G	K_b	ξ	Γ	S_d	m
(M Pa)	(M Pa)	(M Pa)	(M Pa)	(M Pa^{-1})	(M Pa)	(-)
400	$80.2 \cdot 10^3$	$164 \cdot 10^3$	6000	525	10	2

Table 5.3: Material parameters

We are now interested in finding out how the necking of the bar in (5.8) affects the damage evolution on both the RK- and the IE-methods. An imperfection has been introduced to the bar by displacing the lowest node to the left of the bar by 0.025 mm. This imperfection will lead into the bifurcation path which makes the necking effects appear. A reference solution has again been made in a similar manner as in the small deformation case.

5.2.2 Results

The deformation of the solid is shown in figure (5.9), where the necking effects can be observed.

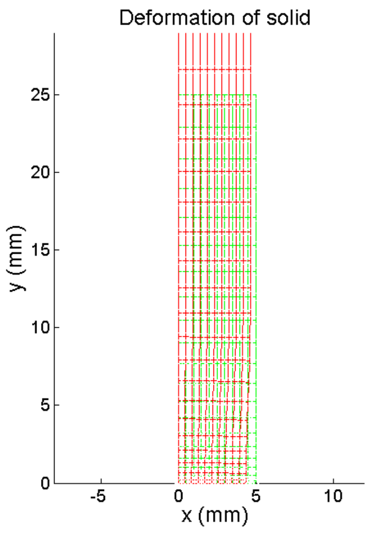


Figure 5.9: Deformation of solid. The deformation of the solid has been magnified two times in order to illustrate the necking more clear.

The global response of the structure with and without the imperfection of the node is illustrated in figure (5.10), which shows that necking is not happening to the structure unless the imperfection of the node is present.

The time stepping for the Implicit Euler method in the large deformation case is also based upon the number of global newton iterations. The heuristic rule has been chosen according to

$$\Delta t_{new} = \Delta t_n f_{inc} \left(\frac{f_{iter}}{n_{newt}} \right)^v \tag{5.3}$$

The damage evolution has been plotted in (5.11) and zoomed in a bit in figure (5.12). The total number of newton iterations for the IE-method was 489 and 470 for the RK-method. The relative error of the damage at the end point was 0.0469 and for the Implicit Euler method it was 0.1119. The damage was measured at the same gauss point for the RK-method and the IE-method at the element marked in figure (5.8). The relative tolerance was put to 0.1 and the absolute tolerances were put to 0 for the RK-method. The time stepping factors for the RK-method were $f_{min} = 0.2, f_{max} = 1.3, f_{safety} = 0.9$ and the time stepping factors in the Implicit Euler method were $f_{iter} = 3, v = 1, f_{inc} = 1.02$. The time step was forced to be greater than $1 \cdot 10^{-3}$ s.

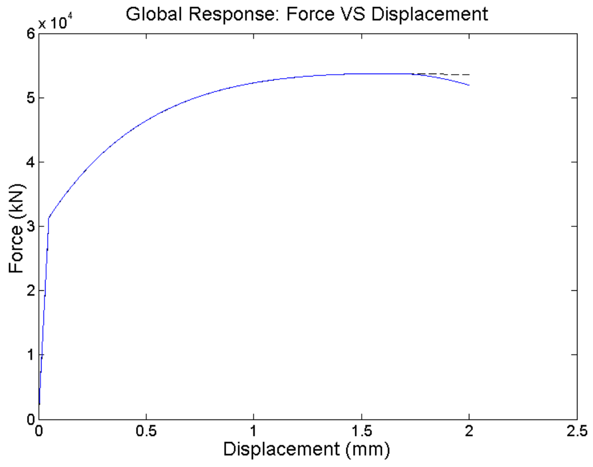


Figure 5.10: The global response of the solid. The displacement is plotted against the internal force developed on the top of the structure. The dashed line shows the global response without the imperfection of the bottom right node while the solid curve is the response with the imperfection of the bottom right node

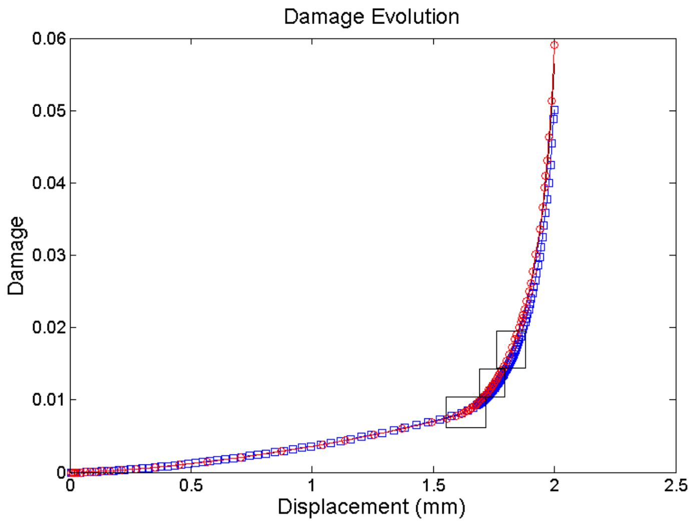
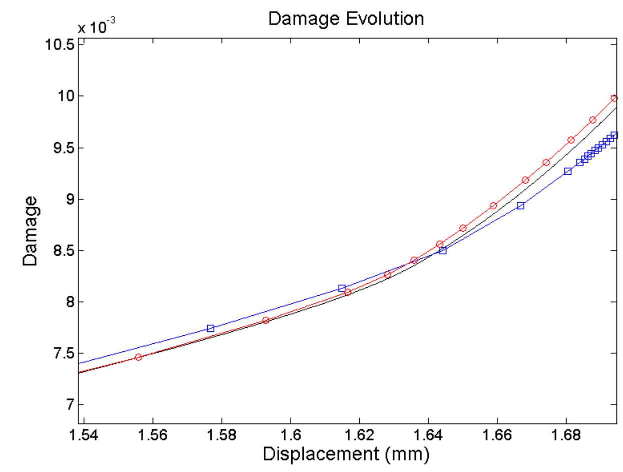
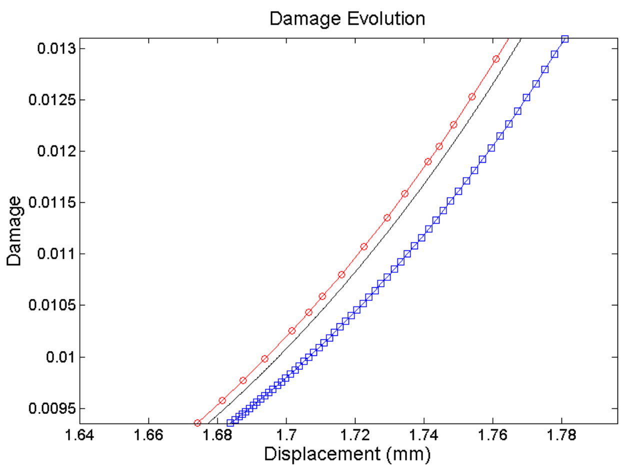


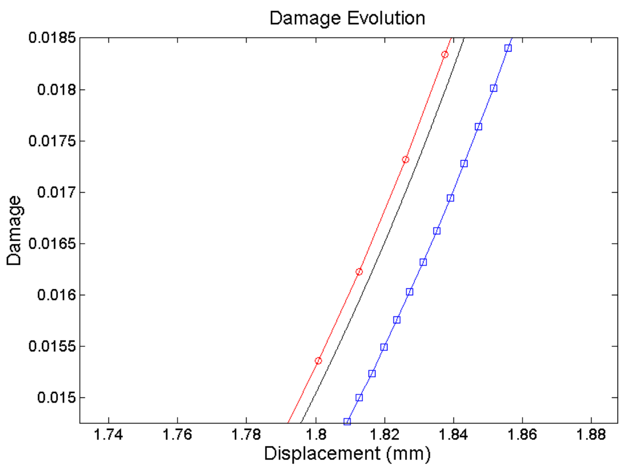
Figure 5.11: Damage evolution. Curve marked with squares: Implicit Euler, Curve marked with circles: Runge-Kutta, Solid curve: Reference solution



(a) Zoomed in around 1.62 mm



(b) Zoomed in around 1.72



(c) Zoomed in around 1.81 mm

Figure 5.12: Damage evolution

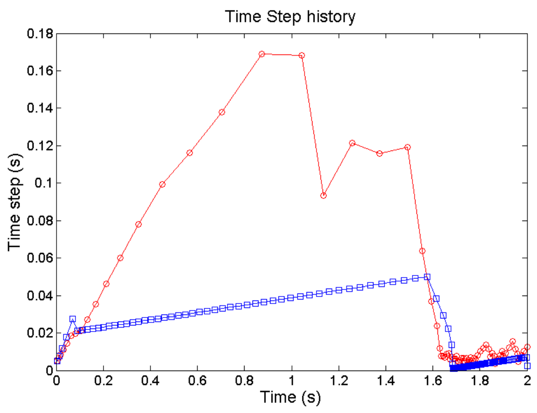


Figure 5.13: Time step history

5.2.3 Discussion

We can again conclude from figure (5.11) and (5.12) that the RK method is more efficient. One can see that the IE-method deviates more from the true solution than the RK method. The RK-method actually coincides with the reference curve before the necking effects occur as can be seen in (5.12(a)) and starts to deviate first after the bifurcation point is passed. A reason for why the RK-method fails to coincide with the reference solution at all times could be that the the predictor/corrector scheme fails. A singular point needs to be passed and the predictor step might fail to find the correct bifurcation path in order to proceed. It is however possible that by adding a restart box as was done in [3] might help to overcome this problem. But still the RK-method can take much larger time steps (about three times larger) than the IE-method and maintain a better accuracy. It can be seen in figure (5.12(a)) that the time stepping method proposed for the IE-method is not so good since it detects too late that it has deviated from the true solution. A predictor/corrector scheme, which the RK-method has, is preferred in order to maintain a high accuracy.

Appendix A

Useful definitions and theorems

A.1 Numerical differentiation

Given a map $\boldsymbol{F}(X) = \boldsymbol{Y} : \mathbb{R}^m \mapsto \mathbb{R}^n$ the numerical differentiation of $\boldsymbol{F}(X)$ is obtained by disturbing each component one at a time in $\boldsymbol{X}$ with the amount h. Let $\boldsymbol{e}_k$ denote the base vector for the variables in $\boldsymbol{X}$ where k is the index in the vector $\boldsymbol{X}$ which is being disturbed, then the k:th row of the differentiated map is given by:

$$\boldsymbol{F}'(\boldsymbol{X})_k = \frac{\boldsymbol{F}(\boldsymbol{X} + h_k \boldsymbol{e}_k) - \boldsymbol{F}(\boldsymbol{X})}{h_k} \tag{A.1}$$

The disturbance h can be chosen in many ways, but in this thesis we have chosen it according to:

$$h_k = max(|X_k|, \sqrt{\varepsilon}) \tag{A.2}$$

where ε denotes the machine accuracy and $|.|$ the absolute value.

A.2 Order of consistency

The following definition is taken [9]. Given an ODE of the form

$$\boldsymbol{y}' = \boldsymbol{f}(t, \boldsymbol{y}),\ t \geq t_0, \quad \boldsymbol{y}(t_0) = \boldsymbol{y}_0 \tag{A.3}$$

Assume that the $\boldsymbol{y}_{n+1}$ is obtained by some time stepping method $\boldsymbol{\mathcal{Y}}$ i.e

$$\boldsymbol{y}_{n+1} = \boldsymbol{\mathcal{Y}}(\boldsymbol{f}, \Delta t_n, \boldsymbol{y}_0, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_n) \tag{A.4}$$

The method is said to be of order p if given the correct solution at t_{n+1} i.e $\boldsymbol{y}(t_{n+1})$ then the error fullfills

$$\kappa = \boldsymbol{y}(t_{n+1}) - \boldsymbol{Y}(\boldsymbol{f}, \Delta t_n, \boldsymbol{y}(t_0), \boldsymbol{y}(t_1), \ldots, \boldsymbol{y}(t_n)) = \mathcal{O}(\Delta t_n^{p+1}) \tag{A.5}$$

for every analytic $\boldsymbol{f}$. An alternate definition is that a method is of order p if it recovers exactly every polynomial solution of degree p or less. The definition above tells us the local behavior of the method, i.e from t_n to t_{n+1} incurres an error of the order Δt_n^{p+1}.

A.3 Implicit function theorem

The following theorem has been taken from [14] and has been freely translated. Let $F(x, y)$ be a $\boldsymbol{C}^1$-function and (a, b) a point on the level curve $F(x, y) = C$. If

$$F'_y(a, b) \neq 0 \tag{A.6}$$

then there exists an open set U around (a, b) such that the restriction of the level curve to U implicitly defines a $\boldsymbol{C}^1$ function $y = f(x)$.

Appendix B

Algorithmic tangent stiffness matrices

B.1 Algorithmic tangent stiffness

The true algorithmic tangent stiffness, D_{ijkl} is required in order to get a quadratic convergence in the Newton-Raphson algorithm. The current stage att load step n is denoted by x in the following derivation and the component from the $\boldsymbol{a}$ matrix in the butcher tablau belonging to the current stage is denoted by a_{xx}. The algorithmic tangent stiffness can be derived trough differentiating the discretized form of the stress-strain relation given by (2.15).

$$\begin{array}{lcl} D_{ijkl} & = & (1-\alpha)\hat{D}_{ijkl} - \frac{\partial \alpha}{\partial \varepsilon_{kl}}\hat{\sigma}_{ij} \\ \hat{D}_{ijkl} & = & \frac{\partial \hat{\sigma}_{ij}}{\partial \varepsilon_{kl}} \\ \hat{\sigma}_{ij} & = & K_b \varepsilon_{pp}\delta_{ij} + 2G(e - e^{(p,S)}) - 2G\zeta N_{ij} \end{array} \tag{B.1}$$

First we will concentrate on the differentiation of $\hat{\sigma}_{ij}$.

B.1.1 Differentiation of $\hat{\sigma}_{ij}$

The first two terms are easily differentiated in the third row of (B.1):

$$\frac{\partial}{\partial \varepsilon_{kl}}\left(K_b \varepsilon_{pp}\delta_{ij}\right) = K_b \delta_{ij}\delta_{kl} \tag{B.2}$$

$$\frac{\partial}{\partial \varepsilon_{kl}}\left(2G(e_{kl} - e_{kl}^{(p,S)})\right) = 2G\left(\mathcal{I}_{ijkl} - \frac{1}{3}\delta_{ij}\delta_{kl}\right) \tag{B.3}$$

Where $\mathcal{I}_{ijkl}$ is defined as

$$\mathcal{I}_{ijkl} = \frac{1}{2}(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}) \tag{B.4}$$

It is however not entirely obvious what the differentiation of the last term in (B.1), $2G\zeta N_{ij}$ is equal to. The term has to be differentiated using the product rule since both ζ and N_{ij} depends on the current strains ε_{kl}. Let us first calculate $\frac{\partial \zeta}{\partial \varepsilon_{kl}}$.

Term 1

From (4.24) we have the following:

$$\Phi_\zeta(\zeta(\varepsilon_{kl}), \varepsilon_{kl}) = ||\Xi_{pq}|| - \gamma - \sqrt{2\sigma_0^2 \left(\frac{\zeta\eta}{a_{xx}\Delta t}\right)^{1/r} + \frac{2}{3}\left(\sigma_y(K)\right)^2} = 0 \tag{B.5}$$

Applying the chain rule gives:

$$\frac{d\Phi_\zeta}{d\varepsilon_{kl}} = \frac{\partial \Phi_\zeta}{\partial \zeta}\frac{d\zeta}{d\varepsilon_{kl}} + \frac{\partial \Phi_\zeta}{\partial \varepsilon_{kl}} = 0 \Rightarrow \frac{d\zeta}{d\varepsilon_{kl}} = -\left(\frac{\partial \Phi_\zeta}{\partial \zeta}\right)^{-1}\frac{\partial \Phi_\zeta}{\partial \varepsilon_{kl}} \tag{B.6}$$

Lets begin calculating $\frac{\partial \Phi_\zeta}{\partial \zeta}$ with the help of (B.5). Differentiating the first term gives

$$\frac{\partial ||\Xi_{ij}||}{\partial \zeta} = \frac{1}{2}\frac{1}{||\Xi_{pq}||}2\Xi_{ij}\frac{\partial \Xi_{ij}}{\partial \zeta} = N_{ij}\frac{\partial \Xi_{ij}}{\partial \zeta} \tag{B.7}$$

The definition of Ξ_{ij} from equation (4.20) gives

$$\frac{\partial \Xi_{ij}}{\partial \zeta} = -\frac{\partial \mu}{\partial \zeta}X_{ij}^S = \mu^2\sqrt{\frac{2}{3}}bX_{ij}^S \tag{B.8}$$

And thus we arrive at

$$\frac{\partial ||\Xi_{ij}||}{\partial \zeta} = \mu^2\sqrt{\frac{2}{3}}bX_{ij}^S N_{ij} \tag{B.9}$$

The second term $\frac{\partial \gamma}{\partial \zeta}$ is differentiated with the help of the definition (4.21) to get

$$\frac{\partial \gamma}{\partial \zeta} = 2G + \mu c + \zeta\frac{\partial \mu}{\partial \zeta}c = 2G + \mu c(1 - \zeta\sqrt{\frac{2}{3}}b\mu) \tag{B.10}$$

And finally differentiating the third and last term gives

$$\frac{\partial}{\partial \zeta}\left(\sqrt{2\sigma_0^2\left(\frac{\zeta\eta}{a_{xx}\Delta t}\right)^{1/r}+\frac{2}{3}\left(\sigma_y(K)\right)^2}\right)=$$
$$\frac{1}{||\Xi_{ij}^{(i)}||-\gamma}\left(\frac{\sigma_0^2\eta}{ra_{xx}\Delta t}\left(\frac{\zeta\eta}{a_{xx}\Delta t}\right)^{1/r-1}+\frac{2}{3}\sigma_y(K)\frac{\partial\sigma_y}{\partial\zeta}\right)$$

Combining these three terms according to B.5 then gives:

$$\frac{\partial\Phi_\zeta}{\partial\zeta}=\mu^2\sqrt{\frac{2}{3}}bX_{ij}^SN_{ij}-\left(2G+\mu c(1-\zeta\sqrt{\frac{2}{3}}b\mu)\right)$$
$$-\frac{1}{||\Xi_{ij}||-\gamma}\left(\frac{\sigma_0^2\eta}{ra_{xx}\Delta t}\left(\frac{\zeta\eta}{a_{xx}\Delta t}\right)^{1/r-1}+\frac{2}{3}\sigma_y(K)\frac{\partial\sigma_y}{\partial\zeta}\right)$$

Then we also have to calculate $\frac{\partial\Phi_\zeta}{\partial\varepsilon_{kl}}$ in order to find $\frac{d\zeta}{d\varepsilon_{kl}}$.

$$\frac{\partial\Phi_\zeta}{\partial\varepsilon_{kl}}=\frac{\partial||\Xi_{pq}||}{\partial\varepsilon_{kl}}=\frac{(s_{ij}^t-X_{ij}^{dev})}{||\Xi_{pq}||}\frac{\partial(s_{ij}^t-X_{ij}^{dev})}{\partial\varepsilon_{kl}}=$$
$$N_{ij}2G(\frac{\partial\varepsilon_{ij}}{\partial\varepsilon_{kl}}+\frac{1}{3}\frac{\partial\varepsilon_{qq}}{\partial\varepsilon_{kl}}\delta_{ij})=N_{ij}2G\mathcal{I}_{ijkl}=2GN_{kl}$$

Defining $\beta=-\frac{\partial\Phi_\zeta}{\partial\zeta}$ we finnaly arrive at

$$\frac{d\zeta}{d\varepsilon_{kl}}=\frac{2G}{\beta}N_{kl} \tag{B.11}$$

And thus we only have find the derivative to $N_{ij}=\frac{\Xi_{ij}}{||\Xi_{pq}||}$.

Term 2

The derivative of the normal can be rewritten to

$$\frac{dN_{ij}}{d\varepsilon_{kl}}=\frac{1}{||\Xi_{uv}||}\frac{d(\Xi_{ij})}{d\varepsilon_{kl}}-N_{ij}\frac{1}{||\Xi_{uv}||}N_{pq}\frac{d\Xi_{pq}}{d\varepsilon_{kl}} \tag{B.12}$$

Let us therefore calculate the derivative of $\Xi_{ij}=s_{ij}^{(t)}-\mu\left(\zeta\right)X_{ij}^S$

$$\frac{d(\Xi_{pq})}{d\varepsilon_{kl}}=\frac{ds_{ij}^t}{d\varepsilon_{kl}}-\frac{d\mu(\zeta)}{d\varepsilon_{kl}}X_{ij}^S \tag{B.13}$$

The derivatives in the expression above can be calculated to be:

$$\frac{ds_{ij}}{d\varepsilon_{kl}} = \mathcal{I}_{ijkl} - \frac{1}{3}\delta_{ij}\delta_{kl}, \quad \frac{d\mu(\zeta)}{d\varepsilon_{kl}} = -\mu^2\sqrt{\frac{2}{3}}b\frac{2G}{\beta}N_{kl} \tag{B.14}$$

Substituting the above expressions into (B.13) and then into (B.12) gives us finally an expression for $\frac{dN_{ij}}{d\varepsilon_{kl}}$

$$\begin{aligned}\frac{dN_{ij}}{d\varepsilon_{kl}} = \frac{2G}{||\Xi_{pq}||}\left(\mathcal{I}_{ijkl} - \delta_{ij}\delta_{kl}\right) + \frac{2Gb\mu^2}{\beta||\Xi_{uv}||}\sqrt{\frac{2}{3}}X^S_{ij}N_{kl} \\ -\frac{2G}{||\Xi_{pq}||}\left(1 + \frac{1}{\beta}\mu^2\sqrt{\frac{2}{3}}X^S_{pq}N_{pq}b\right)N_{ij}N_{kl}\end{aligned}$$

And now we are ready to assemble the terms.

Assembling the terms

We now have everything we need in order to evaluate $(\hat{D}_{ijkl})$.

$$\begin{aligned}\hat{D}_{ijkl} = \frac{\partial}{\partial\varepsilon_{kl}}\left(K_b\varepsilon_{kk}\delta_{ij} + 2G(e - e^{(p,S)})\right) - \frac{\partial}{\partial\varepsilon_{kl}}\left(2G\zeta N_{ij}\right) = \\ K_b\delta_{ij}\delta_{kl} + 2G\left(\mathcal{I} - \frac{1}{3}\delta_{ij}\delta_{kl}\right) - 2G\left(\frac{d\zeta}{d\varepsilon_{kl}}N_{ij} + \zeta\frac{dN_{ij}}{d\varepsilon_{kl}}\right)\end{aligned}$$

And now inserting the calculated terms and after some simplification one arrives at

$$\hat{D}_{ijkl} = 2G\left(\frac{K_b}{2G}\delta_{ij}\delta_{kl} + \gamma_1\left(\mathcal{I}_{ijkl} - \frac{1}{3}\delta_{ij}\delta_{kl}\right) + \gamma_2 N_{ij}N_{kl} + \Gamma_{ij}N_{kl}\right) \tag{B.15}$$

Where

$$\gamma_1 = 1 - \zeta\frac{2G}{||\Xi_{pq}||}, \quad \gamma_2 = \frac{2G\zeta}{||\Xi_{uv}||}\left(1 + \frac{1}{\beta}\mu^2\sqrt{\frac{2}{3}}X^S_{pq}N^S_{pq}b\right) - \frac{2G}{\beta} \tag{B.16}$$

$$\Gamma_{ij} = -\frac{\zeta 2Gb\mu^2}{\beta||\Xi_{pq}||}\sqrt{\frac{2}{3}}X^S_{ij} \tag{B.17}$$

B.1.2 Differentiation of α

From (4.27) we define

$$\Phi_\alpha = \alpha - \alpha^S + \zeta \frac{Y}{S_d(1-\alpha)^m} = 0 \tag{B.18}$$

Differentiation of (B.18) gives

$$\frac{d\alpha}{d\varepsilon_{kl}} \left(1 + \frac{m\zeta Y}{S_d(1-\alpha)^{m+1}} \right) + \frac{d\zeta}{d\varepsilon_{kl}} \frac{Y}{S_d(1-\alpha)^m} + \frac{dY}{d\varepsilon_{kl}} \frac{\zeta}{S_d(1-\alpha)^m} = 0 \tag{B.19}$$

Rearranging (B.19) one ends up with

$$\begin{array}{lcl} \frac{d\alpha}{d\varepsilon_{kl}} & = & -\rho \left(\frac{d\zeta}{d\varepsilon_{kl}} Y + \zeta \frac{dY}{d\varepsilon_{kl}} \right) \\ \rho & = & \frac{1-\alpha}{S_d(1-\alpha)^{m+1} + m\zeta Y} \end{array} \tag{B.20}$$

The term $\frac{d\zeta}{d\varepsilon_{kl}}$ has already been established in (B.11) and all we have left to compute is $\frac{dY}{d\varepsilon_{kl}}$. From the definition of Y (2.14)we have

$$Y = -\frac{1}{2} \varepsilon^e_{ij} D^{hooke}_{ijkl} \varepsilon^e_{kl} = Y = -\frac{1}{2} \hat{\sigma}_{ij} (D^{hooke}_{ijkl})^{-1} \hat{\sigma}_{kl} \tag{B.21}$$

Differentiating (B.21) gives

$$\frac{dY}{d\varepsilon_{kl}} = -\hat{\sigma}_{ij} (D^{hooke}_{ijpq})^{-1} \frac{\hat{\sigma}_{pq}}{\varepsilon_{kl}} \tag{B.22}$$

and now we are finally ready to establish the true algorithmic tangent stiffness D_{ijkl}.

$$D_{ijkl} = (1-\alpha)\hat{D}_{ijkl} + \hat{\sigma}_{ij} \rho \left(\frac{d\zeta}{d\varepsilon_{kl}} Y + \zeta \frac{dY}{d\varepsilon_{kl}} \right) \tag{B.23}$$

where $\hat{D}_{ijkl}$ is given from (B.15),ρ from (B.20) $\frac{d\zeta}{d\varepsilon_{kl}}$ from (B.11) and $\frac{dY}{d\varepsilon_{kl}}$ from (B.22).

B.2 Algorithmic tangent stiffness matrix derivation, finite strains

The algorithmic tangent stiffness matrix is obtained through:

$$\boldsymbol{\mathcal{D}} = 2\frac{d\boldsymbol{S}}{d\boldsymbol{C}} \tag{B.24}$$

Where

$$\boldsymbol{S} = (1-\alpha)\boldsymbol{F}^{p-1}\boldsymbol{S}^e\boldsymbol{F}^{p-T} = (1-\alpha)F^{p-1}_{ik}S^e_{kl}F^{p-1}_{jl} \tag{B.25}$$

Differentiation of this expression gives:

$$\frac{dS_{ij}}{dC_{op}} = (1-\alpha)\Bigg(\underbrace{F^{p-1}_{ik}F^{p-1}_{jl}\frac{dS^e_{kl}}{dC^e_{uv}}\frac{dC^e_{uv}}{dC_{op}}}_{Part1} + \underbrace{F^{p-1}_{ik}S^e_{kl}\frac{dF^{p-1}_{jl}}{dC_{op}} + \frac{dF^{p-1}_{ik}}{dC_{op}}S^e_{kl}F^{p-1}_{jl}}_{Part2}\Bigg)$$
$$-\underbrace{\frac{d\alpha}{dC_{op}}F^{p-1}_{ik}S^e_{kl}F^{p-1}_{jl}}_{Part3} \tag{B.26}$$

The differentiation of the mixed part in the finite element formulation will not be considered here. The differentiation of the internal variables $\boldsymbol{Q}$ with respect to $\boldsymbol{C}$ can be obtained with the numerical jacobian and the residual system (4.47) through:

$$\frac{\partial \boldsymbol{Q}}{\partial \boldsymbol{C}} = \left[\frac{\partial \boldsymbol{R_L}}{\partial \boldsymbol{Q}}\right]^{-1}\frac{\partial \boldsymbol{R_L}}{\partial \boldsymbol{C}} \tag{B.27}$$

where the numerical jacobian $\left[\frac{\partial \boldsymbol{R_L}}{\partial \boldsymbol{Q}}\right]$ is already known from solving the system (4.47) with the Newton Raphson method (4.48) and the term $\frac{\partial \boldsymbol{R_L}}{\partial \boldsymbol{C}}$ has been determined through numerical differntiation (see appendix).

B.2.1 Part 1

We assume that the term $\frac{dS^e_{kl}}{dC^e_{uv}}$ is given and here we will focus our interest on the term $\frac{dC^e_{uv}}{dC_{op}}$. The elastic right cauchy green tensor, C^e_{uv} can be rewritten according to:

$$C^e_{uv} = F^{p-1}_{mu}C_{mn}F^{p-1}_{nv} \tag{B.28}$$

Differentiation of this expression and use of the result in part 2 gives

$$\tfrac{dF_{mu}^{p-1}}{dC_{op}}C_{mn}F_{nv}^{p-1} + F_{mu}^{p-1}\tfrac{dC_{mn}}{dC_{op}}F_{nv}^{p-1} + F_{mu}^{p-1}C_{mn}\tfrac{dF_{nv}^{p-1}}{dC_{op}} =$$

$$-F_{mx}^{p-1}\tfrac{dF_{xr}^{p}}{dC_{op}}F_{ru}^{p-1}C_{mn}F_{nv}^{p-1} + F_{ou}^{p-1}F_{pv}^{p-1} - F_{mu}^{p-1}C_{mn}F_{nq}^{p-1}\tfrac{dF_{qx}^{p}}{dC_{op}}F_{xv}^{p-1} =$$

$$-\tfrac{dF_{xr}^{p}}{dC_{op}}F_{ru}^{p-1}C_{xv}^{e} + F_{ou}^{p-1}F_{pv}^{p-1} - C_{uq}^{e}\tfrac{dF_{qx}^{p}}{dC_{op}}F_{xv}^{p-1} = \tag{B.29}$$

Part 1 is thus equal to

$$Part1 = F_{ik}^{p-1}F_{jl}^{p-1}\frac{dS_{kl}^{e}}{dC_{uv}^{e}}F_{ou}^{p-1}F_{pv}^{p-1} - F_{ik}^{p-1}F_{jl}^{p-1}\frac{dS_{kl}^{e}}{dC_{uv}^{e}}\left(\frac{dF_{xr}^{p}}{dC_{op}}F_{ru}^{p-1}C_{xv}^{e} + C_{uq}^{e}\frac{dF_{qx}^{p}}{dC_{op}}F_{xv}^{p-1}\right) \tag{B.30}$$

B.2.2 Part 2

$$\frac{dF_{ik}^{p-1}F_{kj}^{p}}{dC_{op}} = \frac{dF_{ik}^{p-1}}{dC_{op}}F_{kj}^{p} + F_{ik}^{p-1}\frac{dF_{kj}^{p}}{dC_{op}} = 0 \tag{B.31}$$

Multiplying with F_{js}^{p-1} gives us the sought expression for the derivitive to F_{ij}^{p-1}.

$$\frac{dF_{ik}^{p-1}}{dC_{op}}F_{kj}^{p}F_{js}^{p-1} = -F_{ik}^{p-1}\frac{dF_{kj}^{p}}{dC_{op}}F_{js}^{p-1} = \frac{dF_{is}^{p-1}}{dC_{op}} \tag{B.32}$$

$\frac{dF_{kj}^{p}}{dC_{op}}$ can be found out through (B.27). Part 2 is thus equal to:

$$Part2 = -F_{ik}^{p-1}S_{kl}^{e}F_{jq}^{p-1}\tfrac{dF_{qr}^{p}}{dC_{op}}F_{rl}^{p-1} - F_{iq}^{p-1}\tfrac{-}{F}{}_{ik}^{p-1}S_{kl}^{e}F_{rl}^{p-1}F_{jq}^{p-1}\tfrac{dF_{qr}^{p}}{dC_{op}} - F_{sk}^{p-1}S_{kl}^{e}F_{jl}^{p-1}F_{iq}^{p-1}\tfrac{dF_{qs}^{p}}{dC_{op}} =$$
$$-S_{ir}F_{jq}^{p-1}\tfrac{dF_{qr}^{p}}{dC_{op}} - S_{sj}F_{iq}^{p-1}\tfrac{dF_{qs}^{p}}{dC_{op}} \tag{B.33}$$

And part 2 is done.

B.2.3 Part 3

$\frac{d\alpha}{dC_{op}}$ is known directly from (B.27).

Bibliography

[1] Klaus-Jürgen Bathe. *Finite Element Procedures.* Prentice Hall, 1996.

[2] Peter W. Christensen and Anders Klarbing. *An Introduction to Structural optimization.* Springer, 2009.

[3] P. Ellsiepen and S. Hartmann. Remarks on the interpretation of current non-linear finite element analyses as differential-algebraic equations. *International journal for numerical methods in engineering*, 2001.

[4] Weber G and Anand L. Finite deformation constitutive equations and a time integration procedure for isotropic, hyperelastic-viscoplastic solids. *Computer Methods in Applied Mechanics and Engineering*, 1990.

[5] E. Hairer, S.P. Norsett, and G. Wanner. *Solving Ordinary Differential Equations I.* Springer series in computational mathematics, 1993.

[6] E. Hairer and G. Wanner. *Solving Ordinary Differential equations II.* Springer series in computational mechanincs, 1991.

[7] Stefan Hartman, Georg Lührs, and Peter Haupt. An efficient stress algorithm with applications in viscoplasticity and plasticity. *International journal for numerical methods in engineerng, vol 40, 991-1013*, 1997.

[8] Stefan Hartman, Karsten J. Quint, and Martin Arnold. On plastic incompressibility within time-adaptive finite elements combined with projection techniques. *Comput. Methods Appl. Mech Engrg.*, 2008.

[9] Arieh Iserles. *A first course in the Numerical Analysis of DIfferential Equations.* Cambridge universty press, 2009.

[10] Steen Krenk. *Non-linear Modeling and Analysis of Solids and Structures.* Cambridge university press, 2009.

[11] R. A. Radovitzky M. Ortiz and E. A. Repetto. The computation of the exponential and logarithmic mappings. *International journal for numerical methods in engineering*, 2001.

[12] Niels Ottosen and Hans Petersson. *Introduction to the finite element method.* Pearson Prentice Hall, 1992.

[13] Niels Saabye Ottosen and Matti Ristinmaa. *The Mechanics of Constituive Modelleing.* Elsevier Ltd, 2005.

[14] Arne Persson and Lars-Christer Böiers. *Analys i flera variabler*. Studentlitteratur, 2009.

[15] Matti Ristinmaa and Christer Ljung. *An Introduciton to Stability Analysis*. Lund University, 2002.

[16] Kenneth Runesson, Matti Ristinmaa, and Lennart Mahler. A comparison of viscoplasticity formats and algorithms. *Mechanics of cohesive-frictional materials*, 1997.

[17] J.C. Simo and T.J.r. Hughes. *Computational Inelasticity*. Springer, 2000.

[18] Mathias Wallin and Matti Ristinmaa. Accurate stress updating algorithm based on constant strain rate assumption. *Computer Methods in Applied Mechanics and Engineering*, 2001.

[19] Mathias Wallin and Matti Ristinmaa. Deformation gradient based kinematic hardening model. *International Journal of Plasticity 21*, 2005.

[20] Mathias Wallin, Matti Ristinmaa, and Niels Saabye Ottosen. Kinematic hardening in large strain plasticity. *European Journal of Mechanics A/Solids*, 2003.