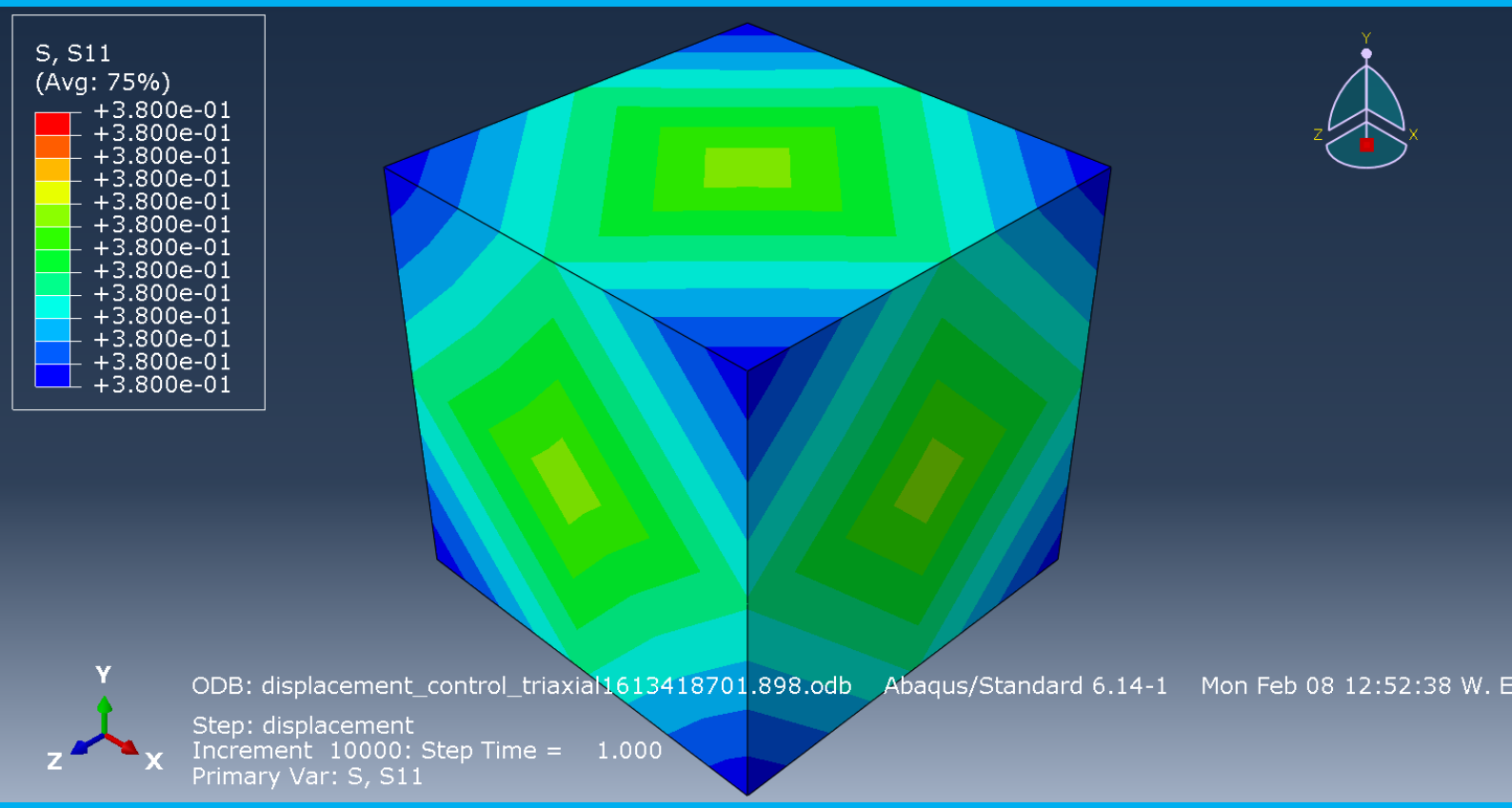
Desai Flow Surface Implementation

A Large Deformation Desai Flow Surface Implementation in Abaqus

H. Hendriks



TU Delft

Desai Flow Surface Implementation

A Large Deformation Desai Flow Surface Implementation in Abaqus

by

H. Hendriks

this thesis is part of the additional research project - course CIE5050-09.
at the Delft University of Technology.

Student number:	4899512	
Project duration:	November 9, 2020 – February 15, 2021	
Thesis committee:	Dr. K. Anupam,	TU Delft, supervisor
	Ir. C. Kasbergen,	TU Delft, supervisor

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Preface

This research project could not be carried out without the help of both my supervisors: Ir. C. Kasbergen and Dr. K. Anupam. I would like to thank both of them for the helpful feedback and great conversations. I'm especially grateful for the weekly meetings on Friday with C. Kasbergen.

H. Hendriks
Delft, February 2021

Contents

1 Introduction

1.1. Background

Investigating the mechanical behaviour of asphaltic materials is an important area of research in road and pavement engineering. Numerous experiments are performed on asphaltic materials to predict their mechanical properties. These mechanical properties are then used in computer models to simulate the behaviour of the material during loading in more complex models, which would be too time consuming to test via experiments.

Most of these computer simulations are finite element method (FEM) analyses. Abaqus is one of the most widely used finite element method programs. Abaqus can for example simulate solids and fluids in static and dynamic analyses. Abaqus implements the modified Drucker-Prager/Cap model, which is a constitutive model with plasticity for static and dynamic simulations of asphaltic materials. This constitutive model has two shortcoming, namely that the plastic flow surface is discontinues and it is not suitable for large deformations.

Lots of advancements are made in this area of research, for example there is a plastic flow surface called the Hierarchical Single Surface model (HiSS) [3]; further called the Desai flow surface. This plastic flow surface is a continuous surface with a cap built into its definition. This plastic flow surface is implemented in Abaqus in [7] and [8], but all implementations up till now assume small deformations. A solution for this would be to use a multiplicative model in a continuum mechanics framework instead of using an additive model.

1.2. Mathematical Notation

The following mathematical notation for tensors is used in this report:

- Matrices and second order tensors are denoted with upper case letters or greek or latin symbols: F, σ;
- Fourth order tensors are denoted with bold upper case letters: $\mathbb{I}$;

This report uses the Einstein summation notation, which means that repeated indices in index notation are implied to be summed over.

The following second order tensor definitions are used in the text:

- I or δ: second order identity tensor, the latter being called the Kronecker delta;
- F: deformation gradient;
- τ: Kirchoff stress tensor;
- σ: Cauchy stress tensor;
- Σ: Mandel stress tensor;
- C: right Cauchy-Green deformation tensor;

- B: left Cauchy-Green deformation tensor;

The following fourth order tensor definitions are used in the text:

- $\mathbb{I}$: fourth order identity tensor (w.r.t. the double dot product);

The double dot product between two tensors is defined as $A : B = A_{ij}B_{ij}$. Likewise, for fourth order tensors $\mathbf{A} : \mathbf{B} = \mathbf{A}_{ijkl}\mathbf{B}_{klpq}$. The same pattern holds for the double dot product between second, fourth and sixth order tensors.

1.3. Scope and Outline

This research project aims to derive and implement an elasto-plastic multiplicative model in the continuum mechanics framework with a Desai flow surface. Therefore, this bridges the gap of having an elasto-plastic model that has a continuous plastic flow surface and can be used with large deformations for asphaltic materials. The implementation of the model will be in Abaqus and a second implementation will be made in Python, which is used for rapid prototyping and testing of the model before implementing it in Abaqus. Both implementations will be compared to each other, a convergence analysis will be performed and the results will be verified via analytical solutions.

The report is structured as follows:

- Chapter 2: introduces the continuum mechanics large deformation framework and derives the multiplicative elasto-plastic model with it.
- Chapter 3: describes the Neo-Hookean Alpha material model, describes the theory of the plastic Desai flow surface, states the hardening and softening curve, and describes the implementation of the model in the Abaqus and the Python framework.
- Chapter 4: presents the methodology used to verify both implementations of the model.
- Chapter 5: presents all results regarding the verification of both implementations of the model.
- Chapter 6: the main conclusions are given and future research activities are suggested.

2

Literature Review

This chapter starts by giving a brief summary of the derivation of an elastic multiplicative model. The summary is a stepping stone for the elasto-plastic multiplicative model. The derivation of this model starts by defining the deformation relation of the elastic and the plastic component in the model. From this relation a general strain energy density function can be set up and by substituting this function in the Clausius-Planck inequality and by using the Coleman-Noll procedure [2], the definition of the Cauchy stress tensor and the dissipation inequality can be derived.

The definition of the Cauchy stress tensor and the dissipation inequality can not be implemented in a FEM program, because the elastic and plastic deformation, and the plastic internal variable are unknown. These can be obtained by performing the Newton-Raphson procedure, which can be derived using the principle of maximum dissipation in combination with the plastic flow surface condition. This condition places a constraint on the possible stress states.

2.1. Elastic Multiplicative Model

This section gives a brief summary of how an elastic multiplicative model is obtained through the continuum mechanics framework.

2.1.1. Summary Derivation

An elastic multiplicative model is defined as a diagram which contains only one single spring. This means that the deformation relation is straightforward, namely $F = F_\infty$.

The following steps must be undertaken to derive an elastic multiplicative model:

1. Define the strain energy density function: $\Psi(F)$;
2. Calculate the time derivative of the strain energy density function: $\dot{\Psi}(F)$;
3. Substitute the time derivative of the strain energy density function into the Clausius-Planck inequality: $D = P : \dot{F} - \dot{\Psi} \geq 0$;
4. Perform the Coleman-Noll procedure to derive the definition of the Kirchoff stress tensor: $D = \left(P - \frac{\partial \Psi}{\partial F}\right) : \dot{F} \geq 0 \xrightarrow{yields} P = \frac{\partial \Psi}{\partial F}$.
5. Determine the Kirchoff stress tensor by taking the derivative of the strain energy density function with respect to the deformation gradient: $P = \frac{\partial \Psi}{\partial F}$.

2.2. Elasto-Plastic Multiplicative Model

This section derives the elasto-plastic multiplicative model in a continuum mechanics framework, which results in the definition of the Cauchy stress tensor and the definition of the dissipation inequality. The dissipation inequality with a plastic flow surface defines the plastic behaviour of the model. To implement this fully in a FEM program a Newton-Raphson procedure must be used. The algorithm and all derivations needed for implementing this procedure are shown.

2.2.1. Potato Diagram

This project aims to implement an elasto-plastic material. First we need to define the elasto-plastic multiplicative model in the continuum mechanics framework. This is defined as a spring and a slider in component in series as shown in figure 2.1. The spring represents the elastic and the slider the plastic deformation.

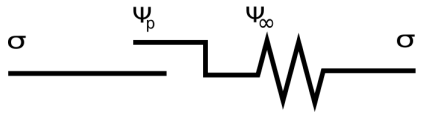


Figure 2.1: Elasto-plastic multiplicative diagram

This model has the following deformation relation, which is shown in figure 2.2

$$F = F_{\infty} \cdot F_p \tag{2.1}$$

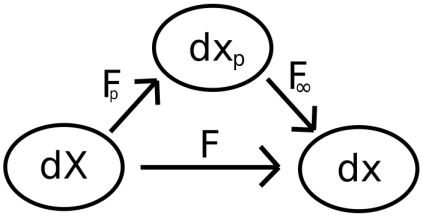


Figure 2.2: Elasto-plastic multiplicative potato diagram

Each multiplicative model must specify a strain energy density function for every elastic component, which the continuum mechanics framework uses to derive for example the Cauchy stress tensor. The elasto-plastic multiplicative model has a strain energy density function in terms of:

$$\Psi\left(F_{\infty}, \xi\right) \tag{2.2}$$

where F_{∞} is the elastic deformation gradient and ξ is the internal plastic variable.

2.2.2. Clausius-Planck Inequality

The Clausius-Planck inequality is defined as:

$$P : \dot{F} - \dot{\Psi} \geq 0 \tag{2.3}$$

The goal of this section is to rewrite the Clausius-Planck inequality such that it can be used to perform the Coleman-Noll procedure. The first step is to substitute the definition of the strain energy density function equation 2.2 in the Clausius-Planck inequality equation 2.3. This step starts by first rewriting the Clausius-Planck inequality in terms of the Cauchy stress tensor instead of the first Piola-Kirchoff stress tensor. The reason for this is that most FEM programs require the calculation of the Cauchy stress tensor. The definition of the first Piola-Korchoff stress tensor in terms of the Cauchy stress tensor is:

$$P = J\sigma F^{-T} \tag{2.4}$$

Substituting this into the Clausius-Planck inequality gives:

$$\begin{aligned} J\sigma F^{-T} : \dot{F} - \dot{\Psi} &\geq 0 \\ J\sigma : \dot{F} \cdot F^{-1} - \dot{\Psi} &\geq 0 \\ J\sigma : L - \dot{\Psi} &\geq 0 \end{aligned} \tag{2.5}$$

$\dot{\Psi}\left(F_{\infty}, \xi\right)$ will be determined and rewritten in terms of L, L_p and $\dot{\xi}$, before substituting it into equation 2.5:

$$\dot{\Psi}\left(F_{\infty}, \xi\right) = \frac{\partial \Psi}{\partial F_{\infty}} : \dot{F}_{\infty} + \frac{\partial \Psi}{\partial \xi}\dot{\xi} \tag{2.6}$$

Rewriting $\frac{\partial \Psi}{\partial F_\infty} : \dot{F}_\infty$ gives:

$$\begin{aligned}\frac{\partial \Psi}{\partial F_\infty} : \dot{F}_\infty &= \frac{\partial \Psi}{\partial F_\infty} : \left(\dot{F} \cdot F_p^{-1} + F \cdot \dot{F}_p^{-1}\right) \\ &= \frac{\partial \Psi}{\partial F_\infty} : \left(L \cdot F_\infty + F_\infty \cdot L_p^{-1}\right) \\ &= \frac{\partial \Psi}{\partial F_\infty} : (L \cdot F_\infty) - \frac{\partial \Psi}{\partial F_\infty} : \left(F_\infty \cdot L_p\right) \\ &= \frac{\partial \Psi}{\partial F_\infty} \cdot F_\infty^T : L - F_\infty^T \cdot \frac{\partial \Psi}{\partial F_\infty} : L_p \end{aligned} \tag{2.7}$$

Introducing the three variables τ_∞, Σ_∞ and q results in the following expression for the time derivative of the strain energy density function:

$$\begin{aligned}\dot{\Psi} &= \underbrace{\frac{\partial \Psi}{\partial F_\infty} \cdot F_\infty^T}_{\tau_\infty} : L - \underbrace{F_\infty^T \cdot \frac{\partial \Psi}{\partial F_\infty}}_{\Sigma_\infty} : L_p + \underbrace{\frac{\partial \Psi}{\partial \xi}}_{-q} \dot{\xi} \\ &= \tau_\infty : L - \Sigma_\infty : L_p - q\dot{\xi} \end{aligned} \tag{2.8}$$

Substituting equation 2.8 in equation 2.5 results in:

$$D = (J\sigma - \tau_\infty) : L + \Sigma_\infty : L_p + q\dot{\xi} \geq 0 \tag{2.9}$$

The goal of rewriting the Clausius-Planck equation such that we can perform the Coleman-Noll procedure and the substitution of the strain energy density function in it are achieved. Performing the Coleman-Noll procedure on equation 2.9 results in:

Non-dissipative case, $L \neq 0$, $L_p = 0$, $q = 0$ and $D = 0$:

$$\sigma = \frac{1}{J}\tau_\infty \tag{2.10}$$

Dissipative case, $L \neq 0$, $L_p \neq 0$, $q \neq 0$ and $D > 0$:

$$\Sigma_\infty : L_p + q\dot{\xi} > 0 \tag{2.11}$$

Concluding the Coleman-Noll procedure, we have an expression for the Cauchy stress tensor and a dissipation inequality, which facilitates the plastic component of the model.

2.2.3. Newton-Raphson Procedure Solved for Plasticity

The previous section derived the results of the Coleman-Noll procedure on a elasto-plastic multiplicative model. This procedure resulted in the definition for the Cauchy stress tensor and the definition of the dissipation inequality. There is still a missing piece in the derivation and that is: how to determine the elastic and the plastic deformation gradients and the internal plastic variable when given a total deformation gradient. Well first this depends on the previous state of the plastic deformation gradient and the internal variable ξ. This problem is solved through the use of a Newton-Raphson procedure, which solves for F_{infty} and ξ given the dissipation inequality and a plastic flow surface; the latter determines which stress states are allowed in the material. The starting point is the principle of maximum plastic dissipation. The principle states that the Mandel stress tensor Σ_∞ and the value of the hardening function q must be chosen such that the dissipation inequality is maximized:

$$\Sigma_\infty : L_p + q\dot{\xi} > 0 \tag{2.12}$$

while subjected to the following criteria, f is called the plastic flow surface:

$$f\left(\Sigma_\infty, q\right) \leq 0 \tag{2.13}$$

The problem is translated into a minimization problem by negating the dissipation inequality, because this results in an easier derivation. This new constraint minimization problem can be rewritten into the following four equations:

$$-L_p + \lambda \frac{\partial f\left(\Sigma_\infty, q\right)}{\partial \Sigma_\infty} = 0 \tag{2.14}$$
$$-\dot{\xi} + \lambda = 0 \tag{2.15}$$
$$\lambda f\left(\Sigma_\infty, q\right) = 0 \tag{2.16}$$
$$\lambda \geq 0 \tag{2.17}$$

where the last two equations are called the Kuhn-Tucker equations. The four equations above can be simplified into:

$$R_1 = -L_p + \dot{\xi} \frac{\partial f\left(\Sigma_\infty, q\right)}{\partial \Sigma_\infty} = 0 \tag{2.18}$$
$$R_2 = f\left(\Sigma_\infty, q\right) = 0 \tag{2.19}$$
$$\dot{\xi} \geq 0 \tag{2.20}$$

The three equations above can be solved by using the Newton-Raphson method for a system of non-linear equations. The Newton-Raphson procedure is derived by taking the first order taylor approximation of the first two equatoins, where both functions are redefined in terms of F_∞ and $\Delta\xi$:

$$\frac{\partial R_1}{\partial F_\infty} : \delta F_\infty + \frac{\partial R_1}{\partial \Delta\xi} \delta\Delta\xi = -R_1 \tag{2.21}$$
$$\frac{\partial R_2}{\partial F_\infty} : \delta F_\infty + \frac{\partial R_2}{\partial \Delta\xi} \delta\Delta\xi = -R_2 \tag{2.22}$$

Which can be rewritten as the following matrix equation:

$$\begin{bmatrix} \frac{\partial R_1}{\partial F_\infty} & \frac{\partial R_1}{\partial \Delta\xi} \\ \frac{\partial R_2}{\partial F_\infty} & \frac{\partial R_2}{\partial \Delta\xi} \end{bmatrix} \begin{bmatrix} \delta F_\infty \\ \delta\Delta\xi \end{bmatrix} = \begin{bmatrix} -R_1 \\ -R_2 \end{bmatrix} \tag{2.23}$$

Only problem so far is that R_1 and R_2 must be rewritten to be equations in F_∞ and $\Delta\xi$. R_1 can be rewritten using $F = F_\infty \cdot F_p$:

$$\begin{aligned} -L_p + \dot{\xi} \frac{\partial f\left(\Sigma_\infty, q\right)}{\partial \Sigma_\infty} &= 0 \\ -\dot{F}_p + \dot{\xi} \frac{\partial f\left(\Sigma_\infty, q\right)}{\partial \Sigma_\infty} \cdot F_p &= 0 \\ \dot{F}_p - \dot{\xi} \frac{\partial f\left(\Sigma_\infty, q\right)}{\partial \Sigma_\infty} \cdot F_p &= 0 \end{aligned} \tag{2.24}$$

Equation 2.24 is a differential equation and is solved in incremental form:

$$\begin{aligned} F_p - exp\left(\Delta t \dot{\xi} \frac{\partial f\left(\Sigma_\infty, q\right)}{\partial \Sigma_\infty}\right) \cdot^t F_p &= 0 \\ F_p - exp\left(\Delta \xi \frac{\partial f\left(\Sigma_\infty, q\right)}{\partial \Sigma_\infty}\right) \cdot^t F_p &= 0 \\ F_p -^t F_p^{-1} \cdot exp\left(-\Delta \xi \frac{\partial f\left(\Sigma_\infty, q\right)}{\partial \Sigma_\infty}\right) &= 0 \\ F_\infty -^t F_\infty \cdot exp\left(-\Delta \xi \frac{\partial f\left(\Sigma_\infty, q\right)}{\partial \Sigma_\infty}\right) &= 0 \end{aligned} \tag{2.25}$$

Using the first order Taylor approximation of the exponential of a matrix, results in:

$$F_\infty - {}^t F_\infty \cdot \left(I - \Delta\xi \frac{\partial f\left(\Sigma_\infty, q\right)}{\partial \Sigma_\infty} \right) = 0 \tag{2.26}$$

This ends the rewriting of R_1 in terms of F_∞. R_2 can be rewritten in terms of $\Delta\xi$ by rewriting q, because the Mandel stress tensor already is written in terms of F_∞ see equation 2.8:

$$q = -\frac{\partial \Psi}{\partial \xi} = -\frac{\partial \Psi}{\partial \Delta\xi}\frac{\partial \Delta\xi}{\partial \xi} \tag{2.27}$$

with $\xi = {}^t\xi + \Delta\xi$ and therefore:

$$q = -\frac{\partial \Psi}{\partial \Delta\xi} \tag{2.28}$$

which concludes the rewriting of R_1 and R_2 in the variables F_∞ and $\Delta\xi$:

$$R_1 = F_\infty - {}^t F_\infty \cdot \left(I - \Delta\xi \frac{\partial f}{\partial \Sigma_\infty} \right) \tag{2.29}$$

$$R_2 = f\left(F_\infty, \Delta\xi\right) \tag{2.30}$$

The derivation of the Newton-Raphson procedure is concluded. It only remains to show how to perform the procedure. The procedure can be performed by performing the following algorithm:

- For displacement iteration: i
  - ${}^t F_{\infty,i} \leftarrow F_i \cdot F_{p,i-1}^{-1}$ and ${}^t\xi_i \leftarrow \xi_{i-1}$
  - $F_{\infty,i} \leftarrow {}^t F_{\infty,i}$; $\Delta\xi_i \leftarrow 0$; converged= $.false.$;
  - Do while not converged:
    - Solve equation 2.23 for $\delta F_{\infty,i}$ and $\delta\Delta\xi_i$;
    - $F_{\infty,i} \leftarrow F_{\infty,i} + \delta F_{\infty,i}$; $\Delta\xi_i \leftarrow \Delta\xi_i + \delta\Delta\xi_i$;
    - converged = $\left(\delta F_{\infty,i} : \delta F_{\infty,i} + (\delta\Delta\xi_i)^2\right) \leq \epsilon^2$
  - enddo;
  - $F_{p,i}^{-1} \leftarrow F_i^{-1} \cdot F_{\infty,i}$
  - $\xi_i \leftarrow {}^t\xi_i + \Delta\xi_i$

Substituting R_1 and R_2 into the Newton-Raphson derivatives, given in equation 2.23, results in the following general equations for the derivatives:

$$\frac{R_1}{F_\infty} = \mathbb{I} + {}^t F_\infty \cdot \Delta\xi \frac{\partial \frac{\partial f}{\partial \Sigma_\infty}}{\partial F_\infty} \tag{2.31}$$

$$\frac{R_1}{\Delta\xi} = {}^t F_\infty \cdot \left[\frac{\partial f}{\partial \Sigma_\infty} + \Delta\xi \frac{\partial \frac{\partial f}{\partial \Sigma_\infty}}{\partial \alpha}\frac{\partial \alpha}{\partial \Delta\xi} \right] \tag{2.32}$$

$$\frac{R_2}{F_\infty} = \frac{\partial f}{\partial \sigma} : \frac{\partial \sigma}{\partial F_\infty} \tag{2.33}$$

$$\frac{R_2}{\Delta\xi} = \bar{I}_1^n \frac{\partial \alpha}{\partial \Delta\xi} \tag{2.34}$$

2.2.4. Assumptions

This report uses the following two assumptions:

$$\frac{\partial F_\infty}{\partial F} = \mathbb{I} \tag{2.35}$$

$$\frac{\partial L_\infty}{\partial L} = \mathbb{I} \tag{2.36}$$

Both assumptions are made to simplify the derivation of the Newton-Raphson derivatives and to simplify the derivation of the displacement controlled gradients, see section 3.4. It can be seen that these two assumptions simplify the derivation, because analytically deriving $\frac{\partial F_\infty}{\partial F}$ or $\frac{\partial L_\infty}{\partial L}$ results in an expression with $\frac{\partial F_p}{\partial F}$ or $\frac{\partial L_p}{\partial L}$ respectively, which expanded further result in an expression in with $\frac{\partial F_\infty}{\partial F}$ or $\frac{\partial L_\infty}{\partial L}$ respectively. This problem could be solved by using the dissipation inequality and the flow surface condition, but this further complicates the derivations. There, the option of these two assumptions is chosen. It is easily seen that these two assumptions make sense, namely by assuming that the plastic deformation gradient is equal to the identity tensor; thus assuming no plasticity has occurred.

3

Model Specification

The previous chapter introduced the elasto-plastic multiplicative model. This is not a complete model because the strain energy density function and the plastic flow surface are undefined. This chapter defines the specific strain energy density function, which consists of two parts: the elastic and the hardening and softening part, and defines the plastic flow surface. These definitions are chosen such that they simulate the behaviour of asphaltic materials.

This chapter starts by introducing the elastic term of the strain energy density function and derives the Cauchy stress tensor. The plastic flow surface is introduced and an explanation is given on why this fits the criteria of the model and its symmetry problem is discussed. The second term of the strain energy density function is defined, namely the hardening and softening curve. At the end of the chapter the specific implementation details are discussed regarding the Abaqus and the Python implementation and a summary is given of all error messages that are implemented in both implementations.

3.1. Neo-Hookean Alpha

The strain energy density function is split in two terms, namely in F_∞ and in ξ. The first is discussed in this section and the latter in the second to last section of this chapter.

The elastic part of the strain energy density function is chosen to be the Neo-Hookean Alpha material, which is a variation of the hyperelastic Blatz-Ko material [1]. This material is chosen, because for small deformations the material behaves according to Hooke's law. The Neo-Hookean Alpha material is defined as:

$$\Psi(I_1, I_2, I_3) = \frac{\mu}{2}\left[(I_1 - 3) + \frac{1}{\alpha}\left(I_3^{-\alpha} - 1\right)\right] \tag{3.1}$$

where I_1, I_2 and I_3 are functions of F_∞, therefore satisfying the requirement for this term. α is defined as $\alpha = \frac{\lambda}{2\mu}$ with λ and μ as the Lamé material constants. The Lamé material constants can be expressed as variables of the Youngs modulus E and the Poisson ratio ν:

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)} \tag{3.2}$$

$$\mu = \frac{E}{2(1+\nu)} \tag{3.3}$$

The Cauchy stress tensor is derived with this strain energy density function by substituting it into the definition of the Cauchy stress tensor in the elasto-plastic multiplicative model, equation 2.10:

$$\sigma = \frac{1}{J}\frac{\partial \Psi}{\partial F_\infty} \cdot F_\infty^T = \frac{\mu}{J}\left(B_\infty - I_3^{-\alpha} I\right) \tag{3.4}$$

The Mandel stress tensor is derived with this strain energy density function by substituting it into the definition of the Mandel stress tensor in the elasto-plastic multiplicative model, equation 2.8:

$$\Sigma_\infty = F_\infty^T \cdot \frac{\partial \Psi}{\partial F_\infty} = \mu\left(C_\infty - I_3^{-\alpha} I\right) \tag{3.5}$$

3.2. Desai Flow Surface

The Desai flow surface [3] is chosen as the definition for the plastic flow surface. The reason for this is that the Desai flow surface is a single continuous cap surface, therefore it has no discontinuities, and the surface is defined in parameters that can experimentally be found in asphaltic materials. Therefore, it can safely be assumed that the plastic behaviour of asphalt can be simulated by the Desai flow surface. The Desai flow surface is defined as:

$$f(\sigma, q) = \frac{J_2}{p_a^2} - \left[-\alpha \bar{I}_1^n + \gamma \bar{I}_1^2\right](1 - \beta S_r)^m \tag{3.6}$$

with:

$$\bar{I}_1 = \frac{I_1 - R}{P_a} \tag{3.7}$$

where J_2 is the second deviatoric invariant, I_1 the first invariant of the elastic deformation gradient F_∞, P_a is the atmospheric pressure, R is the triaxial strength in tension, α the hardening and softening parameter, γ, β and m are material response functions associated with the ultimate behaviour, n is the phase change parameter and S_r the stress ratio given by:

$$S_r = \frac{3\sqrt{3}}{2} \frac{J_3}{J_2^{3/2}} \tag{3.8}$$

where J_2 and J_3 are the second and third deviatoric stress invariants.

From the conclusion of chapter 6 of [4], it is known that for asphaltic materials β can be assumed to be zero, which simplifies equation 3.6:

$$f(\sigma, q) = \frac{J_2}{p_a^2} - \left[q + \gamma \bar{I}_1^2\right] = \frac{J_2}{p_a^2} - \left[-\alpha \bar{I}_1^n + \gamma \bar{I}_1^2\right] \tag{3.9}$$

See appendix B.1 for the derivatives of the Desai flow surface regarding the Newton-Raphson procedure.

The condition $f(\Sigma_\infty, q) \leq 0$, equation 2.13, states for a given plastic flow surface all allowable stress states. Setting n in the Desai flow surface to an integer greater than one and looking at a triaxial stress state in which J_2 is zero. It can be concluded that the Desai flow surface, equation 3.9, still returns a negative value when $I_1 > R$, therefore it is an allowable stress state. Physically this should not be possible, because R is the maximum triaxial strength in tension. For this reason caution should be used when looking at the results of triaxial stress states of implementations of the Desai flow surface.

3.3. Hardening and Softening Curve

The Desai flow surface has the parameter q, which is called the back stress. This parameter is used to define the hardening and softening behaviour of the plastic flow surface. q is defined as $-\alpha(\xi)\bar{I}_1^n$, where α is a function of the internal plastic variable ξ and see section 3.2 for an explanation of $\bar{I}_1^n$. The function $\alpha(\xi)$ can freely be defined as long as it satisfies $q = -\frac{\partial \Psi}{\partial \xi} = -\alpha(\xi)\bar{I}_1^n \leq 0$ due to the fact that each term in Ψ must always be positive. The hardening and softening curve is defined such that the hardening and softening terms are independent and satisfy:

- $\alpha(0) = \alpha_0$.
- $\alpha(\xi_1) = \alpha_1$.
- The slope of the hardening term is equal to: $-\beta_0$.
- The slope of the softening term is equal to: β_1.

therefore, the hardening and softening curve is defined as:

$$\alpha(\xi) = \alpha_0 e^{-\frac{\beta_0}{\alpha_0}\xi} + \alpha_1 e^{\frac{\beta_1}{\alpha_1}(\xi - \xi_1)} + c_1\xi + c_2 \tag{3.10}$$

where α_0, α_1, β_0, β_1 and ξ_1 are hardening parameters freely chosen within the following constraint: the function $\alpha(\xi)$ must always be positive for $\xi \geq 0$. The first term defines the hardening part, the second the softening and the last two terms are used to satisfy the two boundary equations $\alpha(0) = \alpha_0$ and $\alpha(\xi_1) = \alpha_1$, which results in the following definition of c_1 and c_2:

$$c_1 = \frac{-\alpha_0 e^{-\frac{\beta_0}{\alpha_0}\xi_1} - c_2}{\xi_1} \tag{3.11}$$

$$c_2 = -\alpha_1 e^{-\frac{\beta_1}{\alpha_1}\xi_1} \tag{3.12}$$

The derivative of $\alpha(\xi)$ with respect to ξ is needed in the Newton-Raphson procedure. This can easily be calculated from the chosen definition of $\alpha(\xi)$:

$$\frac{\partial \alpha(\xi)}{\partial \xi} = -\beta_0 e^{-\frac{\beta_0}{\alpha_0}\xi} + \beta_1 e^{\frac{\beta_1}{\alpha_1}(\xi-\xi_1)} + c_1 \tag{3.13}$$

Looking at the strain energy density function, equation 2.2, it must have a term in ξ that satisfies $q = -\frac{\partial \Psi}{\partial \xi}$. This term can easily be obtained by integrating $\alpha(\xi)$, equation 3.10, and setting the resulting integration constant c_3, such that $\Psi_\xi(\xi = 0) = 0$, therefore:

$$\Psi_\xi = \left[-\frac{\alpha_0^2}{\beta_0} e^{-\frac{\beta_0}{\alpha_0}\xi} + \frac{\alpha_1^2}{\beta_1} e^{\frac{\beta_1}{\alpha_1}(\xi-\xi_1)} + \frac{1}{2} c_1 \xi^2 + c_2 \xi + c_3 \right] \bar{I}_1^n \tag{3.14}$$

where:

$$c_3 = \frac{\alpha_0^2}{\beta_0} - \frac{\alpha_1^2}{\beta_1} e^{-\frac{\beta_1}{\alpha_1}\xi_1} \tag{3.15}$$

The term Ψ_ξ must always be positive. This is true because α_0, α_1, β_0, β_1 and ξ_1 are chosen such that $\alpha(\xi)$ is always positive for $\xi \geq 0$.

3.4. Implementation

The above defined Neo-Hookean Alpha material, Desai flow surface and the hardening and softening curve as specified within an elasto-plastic multiplicative model are implemented in Abaqus and in Python. The following sections discuss the implementation details of both the Abaqus and the Python implementation.

3.4.1. Abaqus Model

The Abaqus model is implemented in the form of a user material (UMAT) in Abaqus which is programmed in Fortran. In the user material a deformation gradient is given as input and the Cauchy stress tensor and a specific gradient for the displacement- and load steps must be returned in the code. The Cauchy stress tensor is determined via the aforementioned methods in chapter 2 and 3. The specifics of the gradient for the displacement- and load steps are given in appendix B.3.1.

3.4.2. Python Model

The Python model is implemented in a framework that is programmed in Python by the author of this report. This framework is a single deformation gradient displacement controlled program. To create a model within this framework the Cauchy stress must be determined from a given deformation gradient by the program and a specific gradient for the displacement control must be returned. The Cauchy stress tensor is determined via the aforementioned methods in chapter 2 and 3. See for the specifics of this single deformation gradient displacement controlled program appendix B.2 and for the specifics of the gradient for the displacement steps appendix B.3.2.

3.4.3. Error Messages

Both implementation feature error messages, which indicate that something has happened within the model that leads to inaccurate or even incorrect results. The following error messages are included in both models:

- The model returns an error message when the maximum number of iterations is reached in the Newton-Raphson procedure. This error message indicates that the solution did not converge and that probably the number of displacement steps or the number of maximum Newton-Raphson iterations should be increased.
- The model returns an error message when $I_1 > R$, where I_1 is the first invariant of the Cauchy stress tensor and R the triaxial strength in tension. When this condition is met the model is not calculating plasticity correctly and therefore the solution is incorrect. See the end of section 3.2 for an explanation of this error message.

4

Methodology Verification

The previous chapters explained and derived the elasto-plastic multiplicative model, introduced the specific elastic model, plastic flow surface and the hardening and softening curve to be implemented in this project. This model is going to be implemented in Abaqus and in a Python environment. An important part of implementations is its verification. This verification is necessary to know if the implementation behaves according to the theory presented. The verification of the implementation in Abaqus and Python is done by comparing both implementations and by comparing the implementations with analytical solutions. No verification is performed with experimental data, due to the fact that the hardening and softening curve must be fitted to the experimental data. The reason is that the hardening and softening curve uses parameters that are not standard in literature.

This chapter starts by introducing the analyses that are performed on the Abaqus and Python implementation and explains which results are necessary to verify them. It discusses the specific FEM details and lists all material parameters which are used in all analyses. At last the verification of the convergence and the verification with the analytical solutions is discussed.

4.1. Analysis

Six analyses per implementation are being performed for the verification. These are the following analyses: uni-, bi- and triaxial tests in compression and in tension. Each analysis uses displacement control, where each analysis in compression and each in tension use the same prescribed displacement. The compressions variant uses a prescribed displacement such that the elements of the deformation gradient on the diagonal of the prescribed degrees of freedom are equal to 0.98 and likewise in tension till these elements are equal to 1.003. All analyses are performed with a total of 10000 displacement steps and have a maximum of 20 Newton-Raphson iterations per displacement step iteration.

Each analysis returns the Cauchy stress tensor σ, the total deformation gradient F and the internal plastic variable ξ per displacement step. These results are used to construct the stress-strain, the ξ-strain and the stress-ξ curves, which are used to verify the results of the model. The stress-strain curves compare both implementations, the ξ-strain curves also compare both implementations, show if the plastic internal variable behaves as expected and the convergence behaviour, and the stress-ξ curves are used in the verification with the analytical solutions.

All analyses of the Abaqus implementation are performed on a unit cube which uses symmetry in each axis. The unit cube consists of a single C3D20 element, which has 27 integration points and uses quadratic shape functions.

All analyses of the Python implementation are performed on a single deformation gradient, which has one integration point.

4.2. Material Parameters

All analyses use the following material parameters:
Neo Hookean Alpha material parameters

- E = 100.0 N/mm^2
- ν = 0.25

Desai flow surface parameters

- R = 0.03 N/mm^2
- P_a = -0.1 N/mm^2
- n = 4.0
- γ = 0.6

Hardening parameters

- α_0 = 0.005
- α_1 = 0.0051
- β_0 = 40.0
- β_1 = 30.0
- ξ_1 = 1.e-4

4.3. Convergence

The verification of the convergence tests how dependent the hardening and softening behaviour is on the total number of displacement steps. This verification is performed by first computing the ξ-strain curves for different number of total number of displacement steps, namely: 10, 100, 1000 and 10000. This comparison is made only for the Python implementation and only for the uni- and triaxial compression test, because it is assumed that the Abaqus and Python implementation behave exactly the same and the uni- and the triaxial tests are the two most extreme and complex cases of the three. Only compression is tested because most applications are most likely always in compression.

4.4. Analytical Solutions

The verification with the analytical solutions, tests if the Cauchy stress of the implementations stay within the bounds of the Desai flow surface. This is verified by analytically calculating the lower and upper bounds of the Cauchy stress for each ξ for each test: uni-, bi- and triaxial, plotting the ξ-stress curves of the implementations, and looking if the curves are between the bounds.

The lower and upper bounds are calculated by substituting the Cauchy stress conditions of the uni-, bi- and triaxial tests into the Desai flow surface, equation 3.9, and solving for the yield stress by setting the equations to zero.

The Cauchy stress conditions for each test are:

- Uniaxial: $\sigma_1 = \sigma_y$ and $\sigma_2 = \sigma_3 = 0$;
- Biaxial: $\sigma_1 = \sigma_2 = \sigma_y$ and $\sigma_3 = 0$;
- Triaxial: $\sigma_1 = \sigma_2 = \sigma_3 = \sigma_y$

Filling in these conditions results in the following equations:

- Uniaxial: $\frac{\sigma^2}{3P_a^2} - \left[-\alpha(\xi)\left[\frac{\sigma-R}{P_a}\right]^n + \gamma\left[\frac{\sigma-R}{P_a}\right]^2\right] = 0$

- Biaxial: $\frac{\sigma^2}{3P_a^2} - \left[-\alpha(\xi)\left[\frac{2\sigma-R}{P_a}\right]^n + \gamma\left[\frac{2\sigma-R}{P_a}\right]^2\right] = 0$
- Triaxial: $\alpha(\xi)\left[\frac{3\sigma-R}{P_a}\right]^n - \gamma\left[\frac{3\sigma-R}{P_a}\right]^2 = 0$

The resulting equations show that only the yield stress of the triaxial case can be calculated analytically. Therefore, the yield stress of the uni- and the biaxial tests are calculated via a root finder. The analytical solution of the triaxial case is equal to:

$$\sigma = \frac{1}{3}\left[P_a \sqrt[n-2]{\frac{\gamma}{\alpha(\xi)}} + R\right] \tag{4.1}$$

5

Results and Discussion

Chapter 4 introduces the methodology for the verification of the implementation in both Abaqus and in Python of the elasto-plastic multiplicative model as defined in chapter 2 and specified in chapter 3. This chapter shows the results of the analyses and verifications and discusses them. First two general observations are discussed, then the stress-strain and the ξ-strain curves are shown and discussed, after which the hardening and softening convergence dependence is discussed and at last the verification with the analytical results is shown and discussed.

5.1. General Observations

In this section two general observations of the results are discussed. The first being that the uniaxial compression analysis of the Abaqus implementation diverges at $F_{0,0} = 0.98763$, see figures 5.1 and 5.4, while this is not the case for the Python implementation. This could be explained by the fact that ξ is quite high, which means that the first order Taylor approximation in the Newton-Raphson procedure, equation 2.26, becomes quite far off from the real solution. Although, this explanation seems reasonable this does not explain why the Python implementation does not have this divergence, since the Python implementation has the same ξ-strain curve. Another explanation for this divergence could be the assumption $\frac{\partial L_\infty}{\partial L} = \mathbb{I}$, because figure 5.1 shows that the slope of the curve slopes upwards due to softening. This makes the elastic assumption of the gradient a bad approximation. The difference between the Abaqus and the Python implementation could be explained by the fact that both use a different gradient for the displacement control update.

The second general observation is that the biaxial tension analysis of both the Abaqus and Python implementation diverges at $F_{0,0} = 1.002057$, see figures 5.2, 5.5 and 5.9. This observation can be explained by the explanations of the first general observation, although different priorities should be given for this observation because both implementations diverge. Therefore, it is more likely that the cause of the divergence is the first order Taylor approximation, equation 2.26. The value of ξ in the biaxial tension analysis compared to the uniaxial tension analysis supports this, because ξ is about 1.5 times higher in the biaxial tension analysis.

5.2. Stress-Strain Curves

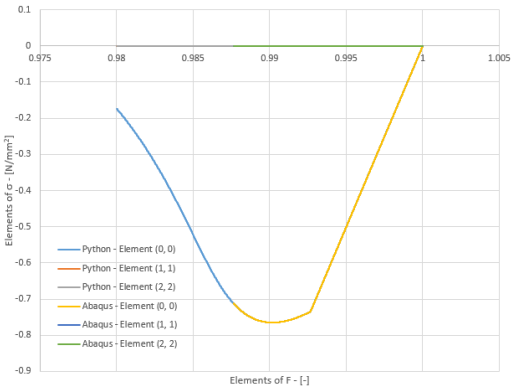
Figure 5.1, 5.2 and 5.3 show respectively the Cauchy stress versus the deformation gradient for the uni-, bi- and triaxial analyses in both compression and tension for the Abaqus and Python implementation.

Figures 5.1, 5.2 and 5.3 show that the Abaqus and the Python models have the same stress-strain curves for the uni-, bi- and triaxial compression and tension analyses, apart from the divergence of the Abaqus implementation in the uniaxial compression analysis as discussed in section 5.1.

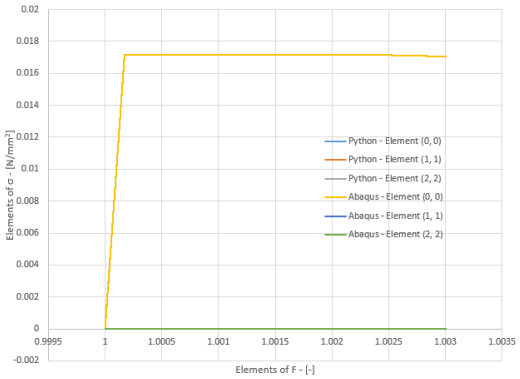
5.3. ξ-Strain Curves

Figure 5.4, 5.5 and 5.6 show respectively the ξ versus the deformation gradient for the uni-, bi- and triaxial analyses in both compression and tension for the Abaqus and the Python implementation.

Figures 5.4, 5.5 and 5.6 show that the Abaqus and the Python models have the ξ-strain curves for uni-, bi- and triaxial compression and tension analyses, apart from the divergence of the Abaqus implementation in the uniaxial compression analysis as discussed in section 5.1.

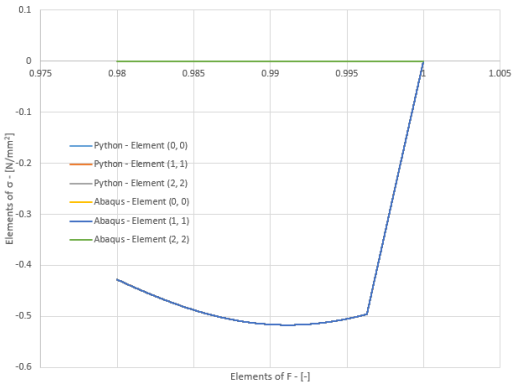


(a) Uniaxial in compression

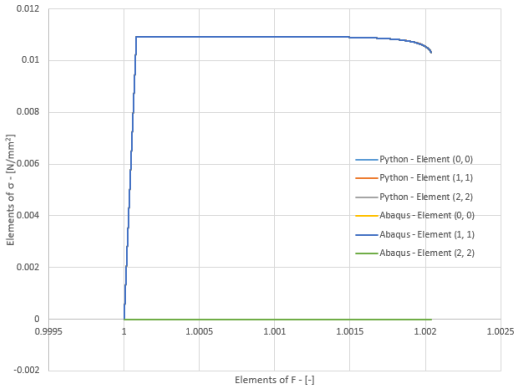


(b) Uniaxial in tension

Figure 5.1: Stress-strain relation under uniaxial displacement control

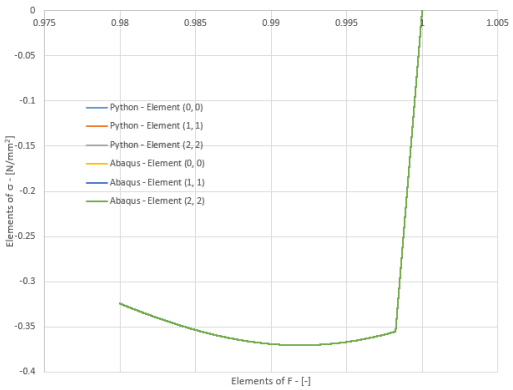


(a) Biaxial in compression

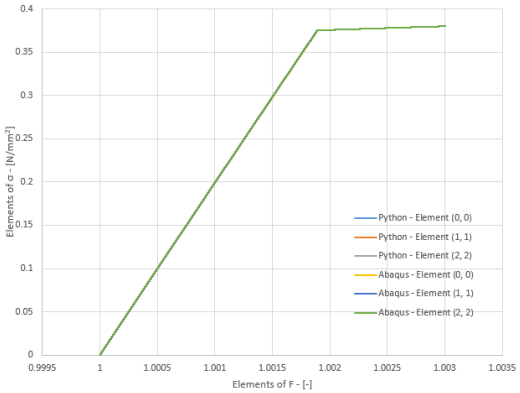


(b) Biaxial in tension

Figure 5.2: Stress-strain relation under biaxial displacement control



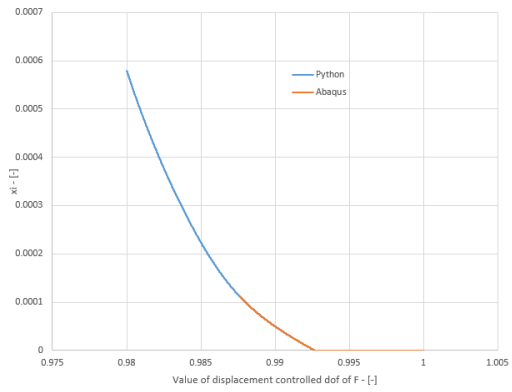
(a) Triaxial in compression



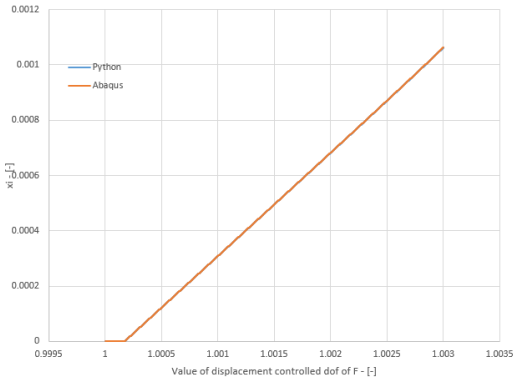
(b) Triaxial in tension

Figure 5.3: Stress-strain relation under triaxial displacement control

Another verification of the ξ-strain curves is that ξ can not decrease, because it follows from the Clausius-Planck inequality and the principle of maximum plastic dissipation that $\dot{\xi} \geq 0$, see equation 2.20. All figures show that this holds true for each analysis in both compression and tension.

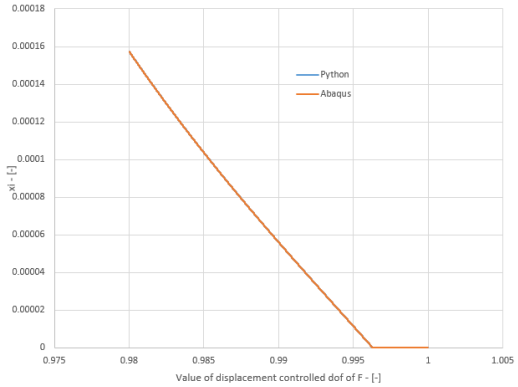


(a) Uniaxial in compression

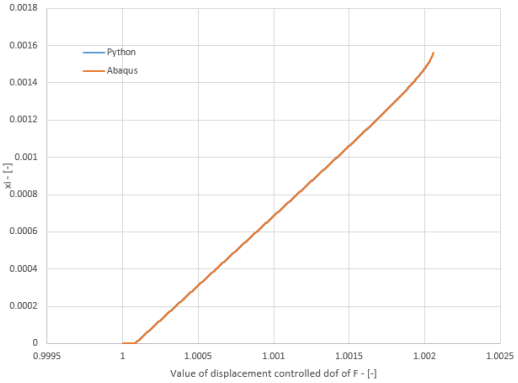


(b) Uniaxial in tension

Figure 5.4: ξ-strain relation under uniaxial displacement control

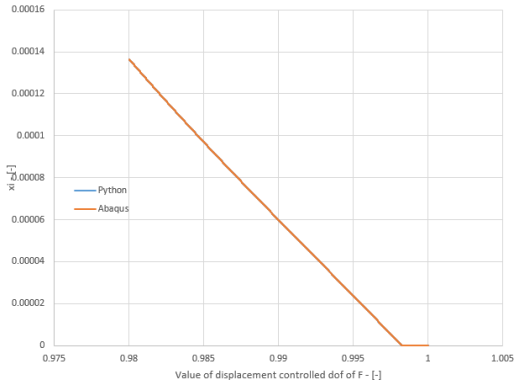


(a) Biaxial in compression

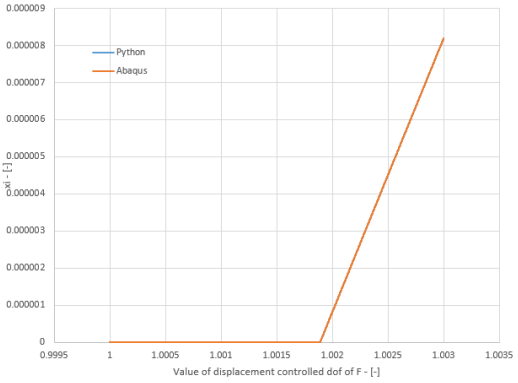


(b) Biaxial in tension

Figure 5.5: ξ-strain relation under biaxial displacement control



(a) Triaxial in compression



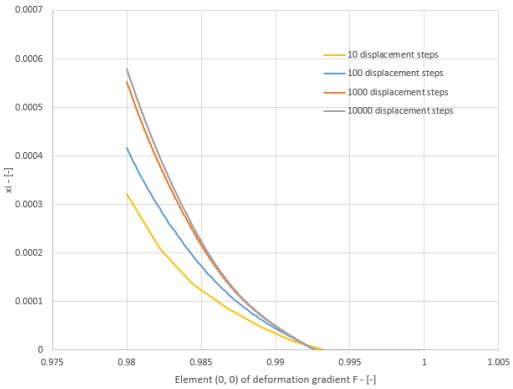
(b) Triaxial in tension

Figure 5.6: ξ-strain relation under triaxial displacement control

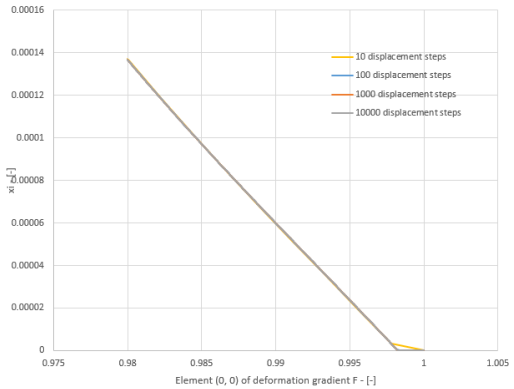
5.4. Hardening and Softening Convergence Dependence

Figure 5.7 shows the ξ-strain curves of the uni- and triaxial analyses in compression of the Python implementation for a total number of displacement steps of 10, 100, 1000 and 10000:

Figure 5.7 shows that the triaxial analysis in compression shows the same ξ-strain curve for each total



(a) Uniaxial in compression



(b) Triaxial in compression

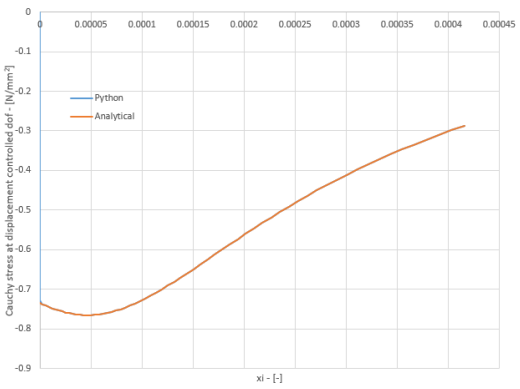
Figure 5.7: ξ-strain curves for uniaxial and triaxial convergence test on Python model

number of displacement steps. The uniaxial compression analysis shows a different behaviour, namely that the curve convergences towards the real curve as the total number of displacement steps increases. This means that the uniaxial compression analysis is not accurate at a lower total number of displacement steps.

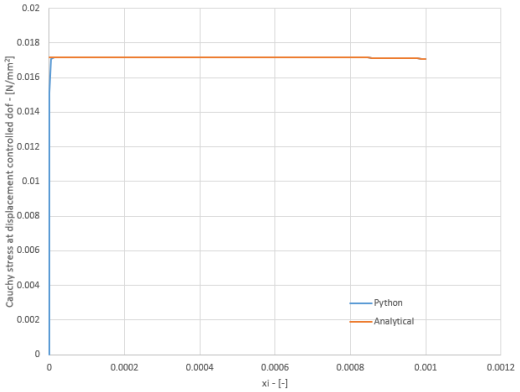
This could be explained by the curvature of the ξ-strain curve in the uniaxial compression analysis, which is not present in the triaxial compression analysis. This would mean that the gradients of the Newton-Raphson procedure, equations 2.31, 2.32, 2.33 and 2.34, become more inaccurate at lower total number of displacement steps especially when combining this with the fact that the first order Taylor approximation is taken, see equation 2.26.

5.5. Analytical Results

Figure 5.8, 5.9 and 5.10 show respectively the stress-ξ curves for the uni-, bi- and triaxial analysis in compression and tension from the Python implementation and the analytical yield stress at which the Desai flow surface equation is equal to zero for the given analysis for each ξ.



(a) Uniaxial in compression

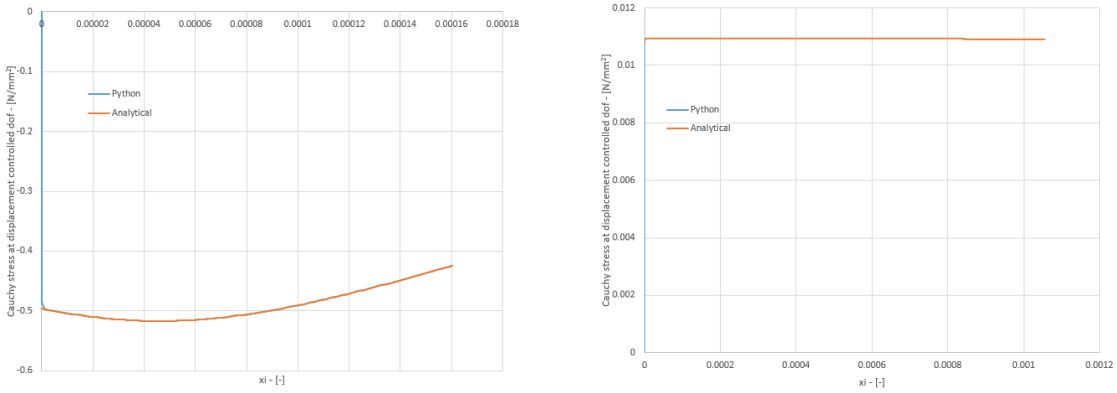


(b) Uniaxial in tension

Figure 5.8: stress-ξ relation under uniaxial displacement control

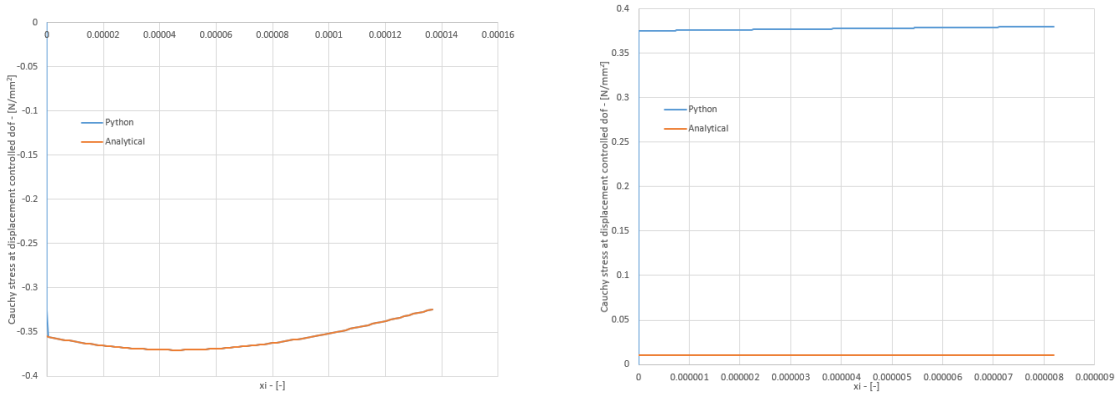
In compression the analytical solution is a lower bound and in tension the analytically solution is an upper bound for the Cauchy stress. The figures show that all curves satisfy these bounds except for the triaxial analysis in tension. This is can be explained from figure 5.11 and using the information at the end of section 3.2. The problem is that the Desai flow surface is symmetric over the line $I_1 = R$, which in turn means that in the triaxial tension analysis the stress state escapes to the right side of the figure. Therefore, the analysis yields when it reaches the right most line, but it should already be yielding at $I_1 = R$.

The same conclusion can be reached for the Abaqus implementation, although the results of the Abaqus



(a) Biaxial in compression (b) Biaxial in tension

Figure 5.9: stress-ξ relation under biaxial displacement control



(a) Triaxial in compression (b) Triaxial in tension

Figure 5.10: stress-ξ relation under triaxial displacement control

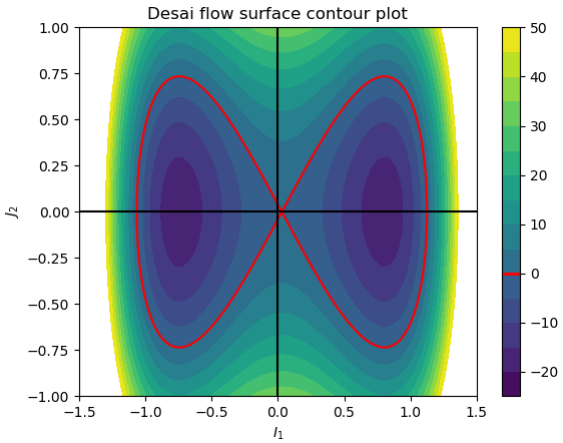


Figure 5.11: Desai flow surface - contour plot

implementation are not shown in this section, because the stress-strain and the ξ-strain curves of all analyses in compression and in tension are the same for the Abaqus and the Python implementation.

6

Conclusions and Future Work

6.1. Conclusions

The goal of this project is to implement a Neo-Hookean Alpha material model with Desai flow surface plasticity with hardening and softening in Abaqus and to verify this implementation for basic displacement controlled cases. The following list summarizes the conclusions that can be made from the results and discussions in chapter 5:

- The stress-strain and the ξ-strain curves of both the Abaqus and the Python implementation are exactly the same except for the uniaxial compression analysis. In this analysis the Abaqus implementation diverges at $F_{0,0} = 0.98763$, which can be explained by the occurrence of large softening behaviour in combination with the assumption on the Abaqus displacement controlled gradient and the first order Taylor approximation in the Newton-Raphson procedure. Therefore, the conclusion can be made that the Abaqus and the Python implementation completely correspond to each other, except when large softening behaviour has occurred.
- Both implementations diverge in the biaxial tension analysis. This occurs probably because of two reasons, namely the assumptions on the displacement controlled gradients or the first order Taylor approximation in the Newton-Raphson procedure. The latter is the probable cause of this issue, because the ξ-strain curve shows a high value of ξ at the point at which the solution diverges. This high value of ξ creates a large error in the first order Taylor approximation. Therefore, it can be concluded that both implementations converge if ξ is relatively small.
- The convergence analysis shows that the Python implementation converges towards a solution when the total number of displacement steps is increased. It also shows that the uniaxial compression analysis convergence takes considerably more total number of displacement steps. This is probably caused by the fact that the ξ-strain curve is non-linear, which is not the case for the triaxial compression analysis. Therefore, the only conclusion can be made that for a high number of total displacement steps the solution has converged. More analyses must be performed to exclude that the uniaxial analysis is the real cause of its worse convergence behaviour instead of the ξ-strain curve being non-linear.
- The same conclusions from the convergence analysis can be made for the Abaqus implementation, because the stress-strain and the ξ-strain curves are exactly the same and the cause of the uniaxial analysis issue is probably due to the current implementation.
- The verification with the analytical solutions shows that all stress-ξ curves of the Python implementation stay between the lower and upper bounds except for the triaxial tension analysis. This conclusion also holds for the Abaqus implementation, due to the fact that the stress-strain and the ξ-strain curves are the same for both implementations. The reason for the triaxial tension analysis going out of bounds is because of symmetry issues in the definition of the equation of the Desai flow surface as previously discussed. Therefore, concluding that both implementations perform plasticity correctly, except for cases where an element is in a triaxial state. Luckily this can easily be tested, namely by checking if $I_1 > R + \epsilon$ is true in an integration point after the Newton-Raphson procedure where ϵ is a small error term.

- The implementation of the elastic-plastic multiplicative model has been successful, therefore the goal of this research project is met. The model has a continuous cap plastic flow surface, namely the Desai flow surface, and it is suitable for large deformations. The potential applications for this model are asphaltic materials which undergo large deformations with hardening and softening in the plastic part of the model. This is an improvement over the standard Abaqus materials which is not suitable for large deformations.

6.2. Recommendations for future Work

During this research project various points came up for improvements and recommendations for future work, these are the following:

- In addition to the uni-, bi- and triaxial tests a simple shear test could also be used to further verify this model. Since this is also a standard test for verifying soil, rock and asphalt models.
- It is recommended to test the model on an experiment of an Asphaltic material: How accurate can this model simulate the behaviour of experimental data and is the current definition of the hardening and softening curve sufficient to accurately describe the plastic behaviour of these experiments.
- The convergence of both implementations can be improved by using a spectral decomposition instead of the first order Taylor approximation in the Newton-Raphson procedure, see equation 2.26. This reduces the approximation error and could potentially solve the divergence issues with the uniaxial compression analysis in the Abaqus implementation and the biaxial tension analysis in both implementations. It could also potentially improve the convergence issue as shown in the convergence verification.
- Further improvement must be made on situations in which elements of the model are in a triaxial tension state. At the moment an error message is given in both implementations when $I_1 > R$, but it would be better to alter the Newton-Raphson procedure to make sure that yielding always occurs when $I_1 > R$.
- An uniaxial compression convergence analysis must be performed in which different material parameters are chosen such that the ξ-strain curve is linear. This analysis could give insight if the convergence issue in section 5.4 was caused by the uniaxial compression analysis or by the non-linearity of the ξ-strain.
- It is advised to incorporate the elastic-plastic part into the gradients used for the displacement control updates in the Abaqus and the Python implementation. This brings more stability and a better convergence to both implementations, because the assumptions in section 2.2.4 are more inaccurate the larger the plastic deformation is.
- In future work a visco-elastic part could be added to this model. This should be done via a parallel dashpot component or a parallel generalized Maxwell component in the potato diagram. When setting this up in parallel the visco-elastic component can be calculated independent of the plastic component, therefore not complicating the model and its implementation.

Bibliography

[1] Paul J Blatz and William L Ko. Application of finite elastic theory to the deformation of rubbery materials. *Transactions of the Society of Rheology*, 6(1):223–252, 1962.

[2] Bernard D Coleman and Walter Noll. The thermodynamics of elastic materials with heat conduction and viscosity. In *The Foundations of Mechanics and Thermodynamics*, pages 145–156. Springer, 1974.

[3] C. S. Desai, K. G. Sharma, G. W. Wathugala, and D. B. Rigby. Implementation of hierarchical single surface $\delta 0$ and $\delta 1$ models in finite element procedure. *International Journal for Numerical and Analytical Methods in Geomechanics*, 15(9):649–680, 1991. doi: https://doi.org/10.1002/nag.1610150904. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/nag.1610150904`.

[4] SMJG Erkens. *Asphalt concrete response, determination, modelling and prediction*. PhD thesis, PhD. Thesis Delft University of Technology, ISBN 0-407-2326-5, DUP Science . . . , 2002.

[5] R. Penrose. A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51(3):406–413, 1955. doi: 10.1017/S0305004100030401.

[6] R. Penrose. On best approximate solutions of linear matrix equations. *Mathematical Proceedings of the Cambridge Philosophical Society*, 52(1):17–19, 1956. doi: 10.1017/S0305004100030929.

[7] Mingqiang Wang and Yang Jun. Three dimensional implementation of hiss model in abaqus. *Springer Series in Geomechanics and Geoengineering*, pages 771–781, 01 2013. doi: 10.1007/978-3-642-32814-5_101.

[8] GW Wathugala and S Pal. Comparison of different implementation algorithms for hiss constitutive models in fem. *International journal of solids and structures*, 36(31-32):4941–4962, 1999.

A

Additional Thesis Documents

A.1. Starting Document

This is the starting document for my (H. Hendriks) research project for the course *CIE5051-09 Additional Graduation Work, Research Project*; this research is not part of a master thesis. This starting document states the proposed research activity, its planning and the learning objectives of the student. The supervisors of the project are Dr. K. Anupam and Ir. C. Kasbergen, who are part of Section of Pavement Engineering at the Faculty of Civil Engineering & Geosciences.

A.1.1. Research Activity

This research project will be carried out for the Section of Pavement Engineering. FEM models of asphalt like materials are made in numerous projects of this Section. In most of these projects standard material models for asphalt like materials are used. But these material models are limited in certain parts. For example (cap models, not continuous) and (only small deformations). Therefore, this research aims to implement the HiSS model (Desai flow surface for plasticity) in a continuum mechanics framework. The HiSS model solves the issue of a discontinuous flow surface since this flow surface has a cap built in its function making it continuous over the whole surface. The continuum mechanics framework has the advantage of working with large deformations.

This model will be implemented in Abaqus using Fortran code since that is the main FEM program that is used in this section. To make it easier to test and implement the model a displaced controlled Python program will be made in which the model can be tested. When the model is implemented in both the Abaqus program and the Python program it will be verified by checking in both correspond to each other and verify the results with results found in literature. These verifications will be done via displacement controlled uni-, bi- and triaxial tests. In Abaqus a load controlled test will be carried out to see how this will perform and check if the results are correct, since the model will also be used in a load controlled setting.

The following goals will be set for this research project:

- An Abaqus implementation of a hyper elastic-plastic material model with hardening which uses a Desai flow surface (HiSS model) in a large deformation framework using continuum mechanics;
- A written report which contains the derivation and verification of the Abaqus implementation;
- A pitch for the supervisors and the research group which will be a concise summary of the research results;

A.1.2. Planning

The following planning will be used for the project:

- Week 1: Literature around the topic of the Desai flow surface (HiSS model) and implementations of plasticity in FEM will be found and read. Also a basic Python displacement control FEM program will be set up, which will be used to partly verify the Abaqus code.

- Week 2: The stress and stiffness tensors of a Neo-Hookean Alpha material will be derived and will be implemented in the Python displacement control program;
- Week 3: The Neo-Hookean Alpha material model will be implemented in Abaqus and von-Mises plasticity will be derived and added to the Python program;
- Week 4: The Von-Mises plasticity model will be implemented in Abaqus;
- Week 5: The Desai flow surface will be derived and implemented in the Python program;
- Week 6: The Desai flow surface will be derived and implemented in Abaqus;
- Week 7: The Desai flow surface implementations will be verified and the final report will be written up till this verification;
- Week 8: A softening behaviour will be derived and added to the Desai flow surface and it will be implemented in the Python program and in Abaqus;
- Week 9: The softening behaviour will be verified and the final report will be written up till this verification;
- Week 10: The final report will be finalized, the lessons learned report will be written and a presentation will be given to the supervisors;

A.1.3. Learning Objectives

The following learning objectives are chosen for this research project:

- A deeper understanding on how to implement and apply the theory of continuum mechanics in a finite element method:
  - Plasticity hardening implementation derivations;
  - Implementation of theory in FEM software;
- Letting go of my perfectionism at certain times:
  - Can work more efficiently when not concentrating on the details at certain times;
  - What is urgent and what is important?;
- Improving my academic writing abilities:
  - Improving overall structure and sentence structure;
  - Improving my speed of writing texts;

A.2. Meeting Notes

The following meeting notes will be found in this appendix:

- First meeting notes;
- Mid-term meeting agenda;
- Mid-term meeting notes;
- Final meeting agenda;
- Final meeting notes;

A.2.1. First Meeting Notes

Subject: First meeting of research project of H. Hendriks
Date: 17-09-2020
Time: 15:00-16:00
Supervisors: Dr. K. Anupam & Ir. C. Kasbergen
Student: H. Hendriks (4899512)

We started the meeting by first talking generally about the subject of the research project. The research that my supervisors want me to do is to implement a Desai flow surface material model in Abaqus. This is a hyperelasto-plastic material with hardening, where the Desai flow surface is a complex plasticity surface that can handle soils, rock and asphalt. This is a research topic that I also would like to do. Because it teaches me how to implement the theory of continuum mechanics and I also like subjects that are quite theory heavy and involve programming.

We established that my task in this research is to implement this model in Abaqus using Fortran code and verify the model with results from literature using uni-, bi- and triaxial tests. After the model is verified a literature report must be made which states the general information of the model, an overview of the derivations, an overview of the algorithm and the verification of the model compared to the literature. At the end of my research project I will give a presentation about my research results to the supervisors and other researchers within their Section.

C. Kasbergen gave me the following advice regarding my research project:

- Make a one element model which is load controlled in Abaqus to check if my implementation of the model works;
- My final report should be around 30 pages. It is not the goal to write a large report but to make it concise and clear;
- I should write my errors and mistakes down, because if I encounter them again than I know how to solve them and not be stuck again. This should also be useful for the following students working on a similar project;

We made the following arrangements regarding the research project:

- Every week (preferably on Friday) C. Kasbergen and I will have a meeting to discuss what I've done the last week;
- I will give C. Kasbergen my netid, because then he can give me access to the computer cluster;
- I will check to see if a user subroutine is possible to create within the student license of Abaqus;

A.2.2. Mid-term Meeting Agenda

Subject: Mid-term assessment of research project of H. Hendriks
Date: 11-12-2020
Time: 15:00-16:00
Supervisors: Dr. K. Anupam & Ir. C. Kasbergen
Student: H. Hendriks (4899512)

Introduction

Current research results

- Research activity and its goal;
- General planning;
- Literature review around topic and implementations of plasticity in FEM;
- Displacement control Python program;

- Derivation of stress and stiffness tensors regarding Neo-Hookean Alpha material;
- Implementation of Neo-Hookean Alpha material in Python and Abaqus;
- Von-Mises plasticity with hardening derivation;
- Von-Mises plasticity with hardening implementation in Python and Abaqus;
- Desai flow surface with hardening derivation;
- Desai flow surface with hardening implementation in Python;

Questions about current research results

Planning for the next five weeks

- Desai flow surface implementation in Abaqus;
- Verification of Desai flow surface implementation in Python and Abaqus via literature;
  - Uni-, bi- and triaxial tests under displacement and load control;
- Write first draft of final report of all research results up till this point;
- Derive and implement the Desai flow surface as a function of the Cauchy stress instead of the Mandel stress;
- Verify the implementation of the Cauchy stress;
- Derive and implement the Desai flow surface with hardening and softening;
- Verify the implementation of the softening behaviour;
- Write second draft of final report of all research results;
- Check and verify second draft of final report;
- Write final report;
- Write lessons learned report;

Questions about planning

Mid-term assessment

Arrangements regarding next five weeks and final assessment

End of meeting

A.2.3. Mid-term Meeting Notes

Introduction

The goal of this mid-term meeting was to inform both supervisors of the current research results and the planning of the next five weeks. The next section of this document describes the comments and feedback per agenda section that the supervisors had. At the end of this document the agenda for this mid-term meeting can be found.

Meeting notes per agenda subject

Introduction:
General introduction of the meeting nothing mentioned here of importance.

Learning objectives:
Told my supervisors about my three learning objectives of this additional research project and what my progress is after the first five weeks.

Current research results:
Talked about the problem of my research, what my research activity is and the goals that I want to achieve at the end of the research project. Small comments were given from the supervisors.

After the introduction to the research problem and goals, I gave a general planning of the whole project.

I gave a summary of all literature that I have found regarding the research projects and what kind of information I could get from each paper and report.

Told about my derivation procedure of the displacement-controlled Python program. I got the feedback that my derivation and theory is quite complex and not immediately understandable. Therefore, the supervisors told me to take extra care when writing this in the final report.

My supervisors told me that in the derivation of the stress and stiffness tensors of the Neo-Hookean Alpha material I made the mistake in calling $\mathbf{C}^{dispcont}$ a stiffness tensor; this should be called a gradient instead. The reason for this is that a stiffness tensor is a stress differentiated with respect to a strain, where $\mathbf{C}^{dispcont}$ is a stress differentiated with respect to a deformation.

The supervisors said that the results from the Neo-Hookean Alpha material looked correct at first glance.

Talked about the derivation of the Von-Mises plasticity with hardening and why this was chosen to be implemented. No comments were given on the derivation, because I took them over from the lecture slides of the course Continuum Mechanics.

The research results from the Neo-Hookean Alpha material with Von-Mises plasticity with hardening looked correct according to the supervisors, when comparing the average stress in all integration points in Abaqus with the stress curve in Python. Though, the supervisors commented that the behaviour of the Abaqus implementation at time 0.55 is weird, but we couldn't pinpoint what the problem could be in the meeting. Therefore, we concluded that there probably is an error in the implementation. We agreed that I should investigate this when I finished implementing the Desai flow surface in Abaqus, because it would be possible that this problem does not occur in that implementation. At the end we had a possible explanation for problem. We argued that the first order approximation of L_p in the derivation of the Newton-Raphson procedure could be the problem.

I showed my derivation of the derivatives needed in the Newton-Raphson procedure for the Desai flow surface with hardening. The supervisors thought that at first inspection they seemed alright. C. Kasbergen will verify these derivations at a later stage. We also talked about the second term in the Desai flow surface formula and C. Kasbergen told me that this term should always be positive. Because when this term is negative it gives rise to non-physical situation, which leads to divergence and errors in the FEM programs. The reason that this was brought up is because I had errors with the Desai flow surface in Python when using a pure triaxial test.

According to the supervisors my results from my Python implementation of the Desai flow surface with hardening looked correct.

The supervisors told me that I made a mistake in the 'known assumptions' section of my presentation. Where β should be equal to zero instead of one.

I talked about the lessons that I have learned in the first five weeks of my research project. The supervisors told me that the communication mistake at the beginning of the project didn't matter (the supervisors thought that I started two months earlier than the research project started). But they noted that I should keep in mind to actively give information regarding my research project and that I shouldn't assume that some-

thing is known. C. Kasbergen also told me to contact him when I'm stuck on something, since talking about it with somebody could speed up the process when you're stuck.

I talked about what I currently have achieved regarding my learning objectives. C. Kasbergen commented that with writing it is better to write spontaneously and reiterate the text till it is at an acceptable level.

Questions about current research results:
There were no questions about the current research results, because the questions were already given in the 'current research' section by the supervisors.

Planning for the next five weeks:
Nothing noteworthy is mentioned while talking about the planning for the next five weeks. My planning was in line with what the supervisors thought I would be able to do in the next five weeks and what is important to do for the research project.

Questions about planning:
There were no questions about the planning.

Mid-term assessment:
The supervisor's said that I did fine and that I know what I'm going to do for the next five weeks. We also talked about the future of this project after I finish my part of it. They noted that probably another student will continue this research.

Dr. K. Anupam and Ir. C. Kasbergen will give me more detailed feedback at a later date, since the meeting ran out of time.

Arrangements regarding next five weeks and final assessment:
Meeting ended before we talked about this agenda subject.

End of meeting:
Meeting ended before we talked about this agenda subject.

A.2.4. Final Meeting Agenda

Subject: Final meeting and assessment of research project of H. Hendriks
Date: 04-03-2021
Time: 14:30-15:30
Supervisors: Dr. K. Anupam & Ir. C. Kasbergen
Student: H. Hendriks (4899512)

Introduction

Presentation of additional research project

Assessment and grading of additional research project

End of meeting

A.2.5. Final Meeting Notes

The meeting is fully recorded, see attachment: Final Meeting.mp4.

A.3. Reflection Statement

This reflection statement is provided in two parts, namely the three learning objectives and a reflection statement about my research skills.

A.3.1. Learning Objectives

The following learning objectives were chosen for this research project:

1. A deeper understanding on how to implement and apply the theory of continuum mechanics in a finite element method:
    - Plasticity hardening implementation derivations;
    - Implementation of theory in FEM software;
2. Letting go of my perfectionism at certain times:
    - Can work more efficiently when not concentrating on the details at certain times;
    - What is urgent and what is important?;
3. Improving my academic writing abilities:
    - Improving overall structure and sentence structure;
    - Improving my speed of writing texts;

The following provides my reflection on each learning objective:

1: In this research project I implemented a Neo-Hookean Alpha elasto-plastic material with a Desai flow surface and hardening and softening. I derived all formulas needed for this implementation. I implemented this both in Abaqus as a new user material and in my own displacement controlled Python program which uses more knowledge about FEM algorithms. Therefore, I can conclude that I succeeded in this learning objective.

2: For most of the project my perfection was not a problem. There were not many cases where I wasted a lot of time by focussing my attention on unnecessary details. The reason for this is that I explicitly wrote this issue down as a learning objective, which made me reflect more on my actions when I concentrated to much on small details. The weekly meetings with C. Kasbergen also helped with making the distinction between what was and what was not important to work on. Therefore, I was focussing my attention on where it was needed instead of getting sidetracked. I conclude that I succeeded in this learning objective by reminding my self when perfectionism took over.

3: The writing process of the thesis can be split into two parts. The first part was my first draft of the thesis, which my supervisors commented on that my writing was not sufficient. The overall structure was good, but I used the wrong form of writing and my grammar could be improved. I overhauled the second draft significantly due to these comments. This resulted in an improved version of my thesis, which both of my supervisors told me. They said that the sentence structure improved significantly, which made the text more clear. Although I improved on this learning objective, I think I can still learn a lot more during my Msc. thesis. Especially by focussing more on writing clear and concise.

My speed of writing has improved slightly. The tip from one of my supervisors that really helped was: write everything down fast and iterate over it to bring it to the final version. Therefore, I improved the speed of my writing by not focusing on perfecting my text when I am writing the first draft.

A.3.2. Research Skills

In this research project I have shown numerous research skills, for example: finding literature, deriving derivations, implementing models, and gathering and discussing results. I liked doing all of these skills, where I liked deriving derivations and implementing models the most. In my opinion and from the feedback of my supervisors I did this these tasks successfully in this research project. Therefore, I think I could become a good researcher and I think that I inspire to be one, which was one of the things that I wanted to find out during this project. Although, I am not 100 percent sure. The reason for this is that this project still felt a bit like an assignment, because the method on how to derive and implement this model was already known; I just had to do it and could not bring in my own ideas. Therefore, I want to find an Msc. thesis project in which the method of solving the problem is unknown and therefore more open to my own ideas. Which would give me a better understanding on what being a researcher entails.

Though there are some areas of improvement, if I really want to be a researcher. First I have to get better at finding literature, since in my opinion that is one of the things that lacked in this project. This is due to two facts, first this research did not require much literature to begin with but second I only looked at the literature halfway during the project instead of the beginning. This was not a problem but for future research this could

potentially be a problem, because I could investigate something that is already done in literature. I also need to improve my writing of academic texts. I have shown improvements over the course of this project, but I am still not there yet. My main issues are still sentence structuring, overall word choice and my speed of writing.

B

Derivations

B.1. Desai Flow Surface Plasticity with Hardening

This section derives the Newton-Raphson procedure derivatives, equations 2.31, 2.32, 2.33 and 2.34, of the Neo-Hookean Alpha material with the Desai flow surface as defined in chapter 3. The derivation of these derivatives starts by first deriving the following intermediate derivatives, namely $\frac{\partial \sigma}{\partial F_\infty}$, $\frac{\partial \sigma}{\partial \Sigma_\infty}$, $\frac{\partial f}{\sigma}$ and $\frac{\partial f}{\partial \Sigma_\infty}$:

$$\frac{\partial \sigma}{\partial F_\infty}_{ijkl} = \frac{\mu}{J}\left[\delta_{ik}F_{\infty,jl} + \delta_{jk}F_{\infty,il} + 2\alpha I_3^{-\alpha} F^{-1}_{\infty,lk}\delta_{ij}\right] \tag{B.1}$$

$$\frac{\partial \sigma}{\partial \Sigma_\infty}_{ijkl} = \frac{1}{J}F^{-T}_{\infty,ik}F^{T}_{\infty,lj} \tag{B.2}$$

$$\frac{\partial f}{\partial \sigma} = \frac{dev(\sigma)}{P_a^2} - \frac{1}{P_a}\left(-\alpha n \bar{I}_1^{n-1} + 2\gamma \bar{I}_1\right) I \tag{B.3}$$

$$\frac{\partial f}{\partial \Sigma_\infty} = \frac{\partial f}{\partial \sigma} : \frac{\partial \sigma}{\partial \Sigma_\infty} \tag{B.4}$$

where σ, Σ_∞ and f are equal to, see chapter 2 and 3:

$$\sigma = \frac{\mu}{J}\left(B_\infty - I_3^{-\alpha} I\right) \tag{B.5}$$

$$\Sigma_\infty = \mu\left(C_\infty - I_3^{-\alpha} I\right) \tag{B.6}$$

$$f = \frac{J_2}{p_a^2} - \left[-\alpha \bar{I}_1^n + \gamma \bar{I}_1^2\right] \tag{B.7}$$

The first Newton-Raphson derivative, equation 2.31, is equal to:

$$\frac{R_1}{F_\infty} = \mathbb{I} + {}^tF_\infty \cdot \Delta\xi \frac{\partial \frac{\partial f}{\partial \Sigma_\infty}}{\partial F_\infty} \tag{B.8}$$

where:

$$\frac{\partial \frac{\partial f}{\partial \Sigma_\infty}}{\partial F_\infty} = \frac{\partial \frac{\partial f}{\partial \sigma}}{\partial F_\infty} : \frac{\partial \sigma}{\partial \Sigma_\infty} + \frac{\partial f}{\partial \sigma} : \frac{\partial \frac{\partial \sigma}{\partial \Sigma_\infty}}{\partial F_\infty} \tag{B.9}$$

$$\frac{\partial \frac{\partial f}{\partial \sigma}}{\partial F_\infty} = \frac{1}{P_a^2}\frac{\partial dev(\sigma)}{\partial F_\infty} - \frac{1}{P_a} I \otimes \left[-\alpha n \frac{\partial \bar{I}_1^{n-1}}{\partial F_\infty} + 2\gamma \frac{\partial \bar{I}_1}{\partial F_\infty}\right] \tag{B.10}$$

$$\frac{\partial dev(\sigma)}{\partial F_\infty} = \left(\mathbb{I} - \frac{1}{3} I \otimes I\right) : \frac{\partial \sigma}{\partial F_\infty} \tag{B.11}$$

$$\frac{\partial \bar{I}_1^{n-1}}{\partial F_\infty} = (n-1)\,\bar{I}_1^{n-2}\frac{\partial \bar{I}_1}{\partial F_\infty} \tag{B.12}$$

$$\frac{\partial \bar{I}_1}{\partial F_\infty} = \frac{1}{P_a} I : \frac{\partial \sigma}{\partial F_\infty} \tag{B.13}$$

$$\begin{aligned}\frac{\partial \frac{\partial \sigma}{\partial \Sigma_\infty}}{\partial F_\infty}_{ijklmn} &= \frac{\partial \frac{1}{J} F^{-T}_{\infty,ik} F^{T}_{\infty,lj}}{\partial F_{\infty,mn}} = \frac{1}{J}\frac{\partial F^{-T}_{\infty,ik}}{\partial F_{\infty,mn}} F^{T}_{\infty,lj} + \frac{1}{J} F^{-T}_{\infty,ik}\frac{\partial F^{T}_{\infty,lj}}{\partial F_{\infty,mn}} \\ &= \frac{1}{J}\left[F^{-T}_{\infty,in} F^{-T}_{\infty,mk} F^{T}_{\infty,lj} + F^{-T}_{\infty,ik}\delta_{jm}\delta_{ln}\right]\end{aligned} \tag{B.14}$$

The second Newton-Raphson derivative, equation 2.32, is equal to:

$$\frac{R_1}{\Delta\xi} = {}^t F_\infty \cdot \left[\frac{\partial f}{\partial \Sigma_\infty} + \Delta\xi \frac{\partial \frac{\partial f}{\partial \Sigma_\infty}}{\partial \alpha}\frac{\partial \alpha}{\partial \Delta\xi}\right] = {}^t F_\infty \cdot \left[\frac{\partial f}{\partial \Sigma_\infty} + \frac{\Delta\xi n}{P_a}\bar{I}_1^{n-1}\frac{\partial \alpha}{\partial \Delta\xi} I : \frac{\partial \sigma}{\partial \Sigma_\infty}\right] \tag{B.15}$$

The third Newton-Raphson derivative, equation 2.33, is equal to:

$$\frac{R_2}{F_\infty} = \frac{\partial f}{\partial \sigma} : \frac{\partial \sigma}{\partial F_\infty} \tag{B.16}$$

The fourth Newton-Raphson derivative, equation 2.34, is equal to:

$$\frac{R_2}{\Delta\xi} = \bar{I}_1^n \frac{\partial \alpha}{\partial \Delta\xi} \tag{B.17}$$

In all Newton-Raphson derivatives $\frac{\partial \alpha}{\partial \Delta\xi}$ is equal to equation 3.13, because $\frac{\partial \alpha}{\partial \Delta\xi} = \frac{\partial \alpha}{\partial \xi}$ since $\xi = {}^t\xi + \Delta\xi$.

B.2. Displacement Control - Python

The results from Abaqus will be partly verified by a Python program. This Python program is a FEM program in a continuum mechanics framework which handles one integration point through displacement control.

B.2.1. Degrees of Freedom

A deformation gradient is a second order tensor with dimension three. A degree of freedom is defined as the number of independent motions. In the case of a deformation gradient it can be said that each element, assuming the deformation gradient is non-symmetric, is a degree of freedom, therefore a deformation gradient has nine. In a displacement controlled environment a certain amount of degrees of freedom have a prescribed displacement. Therefore, it is convenient to define a decomposition of the deformation gradient into its prescribed and non-prescribed degrees of freedom:

$$F = F_p + F_{np} \tag{B.18}$$

where F_p has the same value as F at each prescribed degree of freedom element and likewise F_{np} has the same value for each non-prescribed degree of freedom.

The two following fourth order tensors are defined, to calculate F_p and F_{np}:

Definition The fourth order tensor $\mathbf{D}^0$ is defined as the fourth order tensor where each element is equal to zero except for elements $\mathbf{D}^0_{pqpq} = 1$ where (p,q) is a non-prescribed degree of freedom.

Definition The fourth order tensor $\mathbf{D}^1$ is defined as the fourth order tensor where each element is equal to zero except for elements $\mathbf{D}^1_{pqpq} = 1$ where (p,q) is a prescribed degree of freedom.

It is easy to see that the definitions satisfy equation B.18, when using the following definition for F_np and F_p:

$$F_{np} = D^0 : F \tag{B.19}$$

$$F_p = D^1 : F \tag{B.20}$$

B.2.2. Displacement Steps

This section provides the derivation of the displacement control algorithm in the Python program. The displacement control consists of displacement steps where certain degrees of freedom (elements in the deformation gradient) have prescribed deformations. The goal of the displacement control algorithm is to calculate the deformations of the non-prescribed degrees of freedom in each displacement step.
It is assumed that both $\mathbf{D}^0$ and $\mathbf{D}^1$ are constant over all displacement steps.

The equilibrium of forces tells that the Cauchy stress tensor elements in the non-prescribed degrees of freedom must have a stress equal to zero. Therefore, the following relation is defined:

$$\mathbf{D}^0 : \sigma(F) = 0 \tag{B.21}$$

where σ is the Cauchy stress of a given strain energy density function.

The first order Taylor polynomial approximation of $\sigma(F)$ is equal to:

$$\sigma(F_{i+1}) \approx \sigma(F_i) + \frac{\partial \sigma}{\partial F}|_{F_i} : (F_{i+1} - F_i) \tag{B.22}$$

Substituting equation B.22 into B.21 results in:

$$\mathbf{D}^0 : \sigma(F_{i+1}) \approx \mathbf{D}^0 : \sigma(F_i) + \mathbf{D}^0 : \frac{\partial \sigma}{\partial F}|_{F_i} : (F_{i+1} - F_i) = 0 \tag{B.23}$$

Equation B.23 is rewritten in the follow form:

$$\mathbf{D}^0 : \frac{\partial \sigma}{\partial F}|_{F_i} : (F_{i+1} - F_i) = -\mathbf{D}^0 : \sigma(F_i) \tag{B.24}$$

It can be concluded that each theorem of second order tensors also holds for fourth order tensors by the fact that fourth order tensors have a second order representation, where the dot product between the representations is the same as the double dot product between the original fourth order tensors.
The concept of the Moore-Penrose pseudo-inverse A^+ is used [5], which has the property that if $Ax = b$ then for all x it is true that $\| Ax - b \|_2 \geq \| Az - b \|_2$ with $z = A^+ b$ [6]. Therefore, the pseudo-inverse is the best approximation of the solution of $Ax = b$. Since $\mathbf{D}^0 : \frac{\partial \sigma}{\partial F}|_{F_i}$ of equation B.24 can not be inverted, due to it not being invertible, the pseudo-inverse is used to best approximate the solution:

$$F_{i+1} = -\left[\mathbf{D}^0 : \frac{\partial \sigma}{\partial F}|_{F_i}\right]^+ : \mathbf{D}^0 : \sigma(F_i) + F_i \tag{B.25}$$

Solving for the non-prescribed deformation gradient elements can be done by using equation B.25 and B.19, concluding our displacement controlled deformation gradient procedure:

$$F_{np,i+1} = -\mathbf{D}^0 : \left[\mathbf{D}^0 : \frac{\partial \sigma}{\partial F}|_{F_i}\right]^+ : \mathbf{D}^0 : \sigma(F_i) + \mathbf{D}^0 : F_i \tag{B.26}$$

B.3. Displacement Controlled Gradients

The goals of the both proceeding subsections is to derive an expression for the Abaqus and the displacement controlled gradient tensors. In both subsections the convention is that $B = B_\infty$, I_3 is a function of F_∞ and J is a function of F.

B.3.1. Abaqus Gradient Tensor

The Abaqus gradient tensor is defined as, assuming equations 2.35 $\frac{\partial L_\infty}{\partial L} = \mathbb{I}$:

$$\mathbf{C}^{ABAQUS} = \frac{1}{J}\frac{\partial \dot{\tau}}{\partial d} = \frac{1}{J}\frac{\partial \dot{\tau}}{\partial L_\infty} : \frac{\partial L}{\partial d}$$

The Kirchoff stress for the Neo-Hookean Alpha material is, using equation 3.4:

$$\tau = J\sigma = \mu\left(B - I_3^{-\alpha} I\right) \tag{B.27}$$

Writing out the time derivative of the Kirchoff stress gives:

$$\dot{\tau} = \mu\left(\dot{B} - \left(\dot{I_3^{-\alpha}}\right)I\right) \tag{B.28}$$

where:

$$\dot{B} = L_\infty \cdot B + B \cdot L_\infty^T \tag{B.29}$$

$$\left(\dot{I_3^{-\alpha}}\right) = \frac{\partial I_3^{-\alpha}}{\partial I_3}\dot{I}_3 = -\alpha I_3^{-\alpha-1} 2 I_3 tr\left(L_\infty\right) = -2\alpha I_3^{\alpha} tr\left(L_\infty\right) \tag{B.30}$$

Therefore, the time derivative of the Kirchoff stress is:

$$\dot{\tau} = \mu\left(L_\infty \cdot B + B \cdot L_\infty^T + 2\alpha I_3^{\alpha} tr\left(L_\infty\right) I\right) \tag{B.31}$$

First $\frac{\partial \dot{\tau}}{\partial L_\infty}$ is derived to before substituting it in the definition of the Abaqus gradient tensor:

$$\begin{aligned}\frac{\partial \dot{\tau}}{\partial L_\infty}_{ijpq} &= \mu\left(\frac{\partial L_{ik}}{\partial L_{pq}}B_{kj} + B_{ik}\frac{\partial L_{kj}^T}{\partial L_{pq}} + 2\alpha I_3^{-\alpha}\frac{\partial tr\left(L\right)}{\partial L_{pq}}\delta_{ij}\right) \\ &= \mu\left(\delta_{ip}B_{qj} + \delta_{jp}B_{iq} + 2\alpha I_3^{-\alpha}\delta_{ij}\delta_{pq}\right)\end{aligned} \tag{B.32}$$

The same is done for $\frac{\partial L}{\partial d}$:

$$\frac{\partial L}{\partial d}_{pqkl} = \frac{1}{2}\left(\delta_{pk}\delta ql + \delta_{qk}\delta_{pl}\right) \tag{B.33}$$

Therefore, the Abaqus gradient tensor for the Neo-Hookean Alpha material is:

$$\begin{aligned}\mathbf{C}_{ijkl}^{ABAQUS} &= \frac{1}{J}\frac{\partial \dot{\tau}}{\partial L_\infty}_{ijpq}\frac{\partial L}{\partial d}_{pqkl} \\ &= \mu\left(\delta_{ip}B_{qj} + \delta_{jp}B_{iq} + 2\alpha I_3^{-\alpha}\delta_{ij}\delta_{pq}\right)\frac{1}{2}\left(\delta_{pk}\delta ql + \delta_{qk\delta_{pl}}\right) \\ &= \frac{\mu}{2J}\left[\delta_{ik}B_{lj} + \delta_{il}B_{kj} + \delta_{jk}B_{il} + \delta_{jl}B_{ik} + 4\alpha I_3^{-\alpha}\delta_{ij}\delta_{kl}\right]\end{aligned} \tag{B.34}$$

B.3.2. Displacement Control Gradient Tensor

The gradient tensor for the displacement control algorithm is defined as, assuming equations 2.35 $\frac{\partial F_\infty}{\partial F} = \mathbb{I}$:

$$\mathbf{C}_{ijkl}^{DISPCONT} = \frac{\partial \sigma}{\partial F} = \frac{\partial \sigma}{\partial F_\infty}$$

The Cauchy stress for the Neo-Hookean Alpha material is according to equation 3.4:

$$\sigma = \frac{\mu}{J}\left(B - I_3^{-\alpha} I\right) \tag{B.35}$$

The derivative of the Neo-Hookean Alpha Cauchy stress definition with respect to the elastic deformation gradient is:

$$\begin{aligned}\mathbf{C}_{ijkl}^{DISPCONT} &= \frac{\partial \frac{\mu}{J}\left(B - I_3^{-\alpha} I\right)}{\partial F_\infty} \\ &= \frac{\mu}{J}\left(\frac{\partial B}{\partial F_\infty} - I \otimes \frac{\partial I_3^{-\alpha}}{\partial F_\infty}\right)\end{aligned} \tag{B.36}$$

The two derivatives are equal to:

$$\frac{\partial B}{\partial F_\infty}_{ijkl} = \delta_{ik}F_{jl} + \delta_{jk}F_{il} \tag{B.37}$$

$$\frac{\partial I_3^{-\alpha}}{\partial F_\infty} = 2\alpha I_3^{-\alpha} F^{-1} \tag{B.38}$$

Therefore, the displacement controlled gradient tensor for the Neo-Hookean Alpha material is:

$$\mathbf{C}_{ijkl}^{DISPCONT} = \frac{\mu}{J}\left[\delta_{ik}F_{jl} + \delta_{jk}F_{il} + 2\alpha I_3^{-\alpha}F_{kl}^{-1}\delta_{ij}\right] \tag{B.39}$$

C

Additional Derivation

C.1. Multiplicative Continuum Mechanics Models

This section explores the generalization of multiplicative models in a continuum mechanics framework. The starting point is the definition for a general multiplicative model and the ending point is the Clausius-Planck in inequality with the strain energy density substituted. At last an example with the elasto-plastic multiplicative model is given.

C.1.1. Generalization of Multiplicative Models

The energy strain density function is a function of the variables of a set of elastic deformation gradients F_i and a set of internal variables ξ_j to a scalar:

$$\Psi\left(F_1, F_2, ..., F_n, \xi_1, \xi_2, ..., \xi_m\right) \tag{C.1}$$

The Clausius-Planck inequality is defined as:

$$\tau : L - \dot{\Psi} \geq 0 \tag{C.2}$$

Filling in the strain energy density function into the Clausius-Planck inequality results in:

$$\tau : L - \sum_{i=0}^{n} \left[\frac{\partial \Psi}{\partial F_i} : \dot{F}_i \right] - \sum_{j=0}^{m} \left[\frac{\partial \Psi}{\partial \xi_j} \dot{\xi}_j \right] \geq 0 \tag{C.3}$$

$\frac{\partial \Psi}{\partial F_i}$ and $\frac{\partial \Psi}{\partial \xi_j}$ can easily be calculated from the specified strain energy density function. $\dot{F}_i$ is rewritten in terms of the velocity gradients of the total and the non-elastic deformation gradients before substituting it back in the Clausius-Planck inequality. Therefore, the goal is to rewrite $\dot{F}_i$ in a term which can be written as $C_0 : L$ where L is the total velocity gradient and where the other terms can be written as $C_i : L_i$ where L_i is a non-elastic velocity gradient. This can be done by substituting the following factorisation of the elastic deformation gradients, where F is the total deformation gradient:

$$F_i = F_{i,l} \cdot F \cdot F_{i,r}^{-1} \tag{C.4}$$

This expression is substituted in $\dot{F}_i$, which results in:

$$\frac{\partial \Psi}{\partial F_i} : \dot{F}_i = \frac{\partial \Psi}{\partial F_i} : \left(\dot{F}_{i,l} \cdot F \cdot F_{i,r}^{-1} \right) + \frac{\partial \Psi}{\partial F_i} : \left(F_{i,l} \cdot \dot{F} \cdot F_{i,r}^{-1} \right) + \frac{\partial \Psi}{\partial F_i} : \left(F_{i,l} \cdot F \cdot \dot{F}_{i,r}^{-1} \right) \tag{C.5}$$

Expanding the first term:

$$\frac{\partial \Psi}{\partial F_i} : \left(\dot{F}_{i,l} \cdot F \cdot F_{i,r}^{-1} \right) = \frac{\partial \Psi}{\partial F_i} : \left(\dot{F}_{i,l} \cdot F_{i,l}^{-1} \cdot F_{i,l} \cdot F \cdot F_{i,r}^{-1} \right) = \frac{\partial \Psi}{\partial F_i} : \left(L_{i,l} \cdot F_i \right) = \left(\frac{\partial \Psi}{\partial F_i} \cdot F_i^T \right) : L_{i,l} \tag{C.6}$$

Expanding the second term:

$$\frac{\partial \Psi}{\partial F_i} : \left(F_{i,l} \cdot \dot{F} \cdot F_{i,r}^{-1}\right) = \frac{\partial \Psi}{\partial F_i} : \left(F_{i,l} \cdot \dot{F} \cdot F^{-1} \cdot F \cdot F_{i,r}^{-1}\right) = \frac{\partial \Psi}{\partial F_i} : \left(F_{i,l} \cdot L \cdot F \cdot F_{i,r}^{-1}\right) = \left(F_{i,l}^T \cdot \frac{\partial \Psi}{\partial F_i} \cdot F_{i,r}^{-T} \cdot F^T\right) : L \tag{C.7}$$

Expanding the third term and using the fact that $L_{i,r}^{-1} = -L_{i,r}$:

$$\frac{\partial \Psi}{\partial F_i} : \left(F_{i,l} \cdot F \cdot \dot{F}_{i,r}^{-1}\right) = \frac{\partial \Psi}{\partial F_i} : \left(F_{i,l} \cdot F \cdot F_{i,r}^{-1} \cdot F_{i,r} \cdot \dot{F}_{i,r}^{-1}\right) = \frac{\partial \Psi}{\partial F_i} : \left(F_i \cdot L_{i,r}^{-1}\right) = \left(F_i^T \cdot \frac{\partial \Psi}{\partial F_i}\right) : L_{i,r}^{-1} = -\left(F_i^T \cdot \frac{\partial \Psi}{\partial F_i}\right) : L_{i,r} \tag{C.8}$$

Putting it all together:

$$\frac{\partial \Psi}{\partial F_i} : \dot{F}_i = \left(\frac{\partial \Psi}{\partial F_i} \cdot F_i^T\right) : L_{i,l} - \left(F_i^T \cdot \frac{\partial \Psi}{\partial F_i}\right) : L_{i,r} + \left(F_{i,l}^T \cdot \frac{\partial \Psi}{\partial F_i} \cdot F_{i,r}^{-T} \cdot F^T\right) : L \tag{C.9}$$

The following three definitions are defined $\tau_i = \frac{\partial \Psi}{\partial F_i} \cdot F_i^T$, $\Sigma_i = F_i^T \cdot \frac{\partial \Psi}{\partial F_i}$ and $q_j = -\frac{\partial \Psi}{\partial \xi_j}$. By substituting equation C.9 into the Clausius-Planck inequality, equation C.3, and using the two definitions, this results in:

$$D = \left(\tau - \sum_{i=0}^{n}\left[F_{i,l}^T \cdot \frac{\partial \Psi}{\partial F_i} \cdot F_{i,r}^{-T} \cdot F^T\right]\right) : L - \sum_{i=0}^{n}\left[\left(\frac{\partial \Psi}{\partial F_i} \cdot F_i^T\right) : L_{i,l}\right] + \sum_{i=0}^{n}\left[\left(F_i^T \cdot \frac{\partial \Psi}{\partial F_i}\right) : L_{i,r}\right] + \sum_{j=0}^{m}\left[q_j \dot{\xi}_j\right] \geq 0 \tag{C.10}$$

Performing the Coleman-Noll procedure on equation C.10, results in:

Non-dissipative case, $L \neq 0$, for all elastic deformation gradient $L_i \neq 0$, for all non-elastic deformation gradients $L_j = 0$, for all internal variables $q_j = 0$ and $D = 0$:

$$\tau = \sum_{i=0}^{n} F_{i,l}^T \cdot \frac{\partial \Psi}{\partial F_i} \cdot F_{i,r}^{-T} \cdot F^T \tag{C.11}$$

Dissipative case, $L \neq 0$, for all elastic deformation gradient $L_i \neq 0$, for all non-elastic deformation gradients $L_j \neq 0$, for all internal variables $q_j \neq 0$ and $D \geq 0$:

$$\sum_{i=0}^{n}\left[\Sigma_i : L_{i,r} - \tau_i : L_{i,l}\right] + \sum_{j=0}^{m}\left[q_j \dot{\xi}_j\right] \geq 0 \tag{C.12}$$

C.1.2. Generalized Multiplicative Model Example

This section gives an example of the generalized multiplicative model. The example chosen is the elasto-plastic multiplicative model as defined in section 2.2. The deformation relation for this model is:

$$F = F_\infty \cdot F_p \tag{C.13}$$

where the strain energy density is defined as:

$$\Psi\left(F_\infty, \xi\right) \tag{C.14}$$

Substituting the deformation relation in the factorisation, equation C.4, results in:

$$F_\infty = I \cdot F \cdot F_p^{-1} \tag{C.15}$$

Thus, $F_i = F_\infty$, $F_{i,l} = I$ and $F_{i,r} = F_p$ in the factorisation. Substituting this into equation C.11, results in the following definition of τ:

$$\tau = I^T \cdot \frac{\partial \Psi}{\partial F_\infty} \cdot F_p^{-T} \cdot F^T = \frac{\partial \Psi}{\partial F_\infty} \cdot F_\infty^T = \tau_\infty \tag{C.16}$$

Rewriting this into the definition of the Cauchy stress, gives:

$$\sigma = \frac{1}{J}\tau_\infty \tag{C.17}$$

Substituting the factorisation in equation C.12, results in the following dissipation inequality:

$$\Sigma_\infty : L_p - \tau_\infty : I + q\dot{\xi} = \Sigma_\infty : L_p + q\dot{\xi} \geq 0 \tag{C.18}$$

The Cauchy stress definition, equation 2.10, and the dissipation inequality definition, equation 2.11, of the elasto-plastic model in section 2.2 shows that the same definitions are reached with the generalized multiplicative model. Therefore, completing the example.