Implementation of the Arruda-Boyce Material Model for Polymers in Abaqus

Vegard Tømmernes

Master of Science in Mechanical Engineering
Submission date: June 2014
Supervisor: Zhiliang Zhang, KT

Norwegian University of Science and Technology
Department of Structural Engineering

Abstract

In this Master's thesis, a user material subroutine for polymers is implemented in the FEA-program, Abaqus. The first part of the thesis provides an overview of large deformation mechanics and the Arruda-Boyce constitutive law for polymers. This is a strain-rate dependent viscoplastic material model that includes strain softening and hardening. The main principles behind user material subroutines for Abaqus were presented, and an explicit approach was chosen due to simplicity. Several challenges regarding the implementation have been analyzed and solutions have been proposed. One particular problem was the calculation of Hencky strain where the traditional Padé approximation did not suffice.

This thesis also explains how to use the user material subroutine in an FEA, as well as explaining weaknesses and how to avoid pitfalls. The implemented model was thoroughly tested, with several strain rates and loading conditions. The material parameters in the model were also examined and discussed.

An analysis of void growth in polycarbonate was done using the implemented model. An unknown error in the user material constrained the analysis to very small deformations. However, the analysis showed that polymers experience void growth, but not as excessive as metals. The reason for this is probably the strain hardening effect imposed by the polymer chains.

It was concluded that several types of polymers may be modeled by the user material subroutine, but that it has to be further developed in order to account for complex geometries with large deformations. This thesis aims to offer all NTNU students useful tips on how to develop user material subroutines for polymers in Abaqus, and thus make a small contribution to the research of polymer behavior.

Sammendrag

Denne masteroppgaven omhandler implementering av Arruda-Boyce materialmodellen for polymerer i Abaqus. Første del av oppgaven gir en oversikt over hvordan man håndterer store deformasjoner, og teorien bak den valgte konstitutive loven. Arruda-Boyce-modellen er avhengig av tøyningsrate, og inkluderer egenskaper som tøyningsmykgjøring og herding. Det ble gjort rede for de to hovedprisnippene bak brukermaterialmodeller i Abaqus, og valget falt på en eksplisitt tilnærming. Flere utfordringer vedrørende implementering er analysert, og da spesielt beregningen av Henckytøyning, der det viste seg at en Padétilnærming ikke var tilstrekkelig.

Denne avhandlingen fungerer også som en bruksanvisning for brukerdefinerte materialmodeller i Abaqus, i tillegg til å forklare svakheter og å gi tips for å unngå fallgruver. Modellen ble grundig testet med flere tøyningshastigheter og lasteforhold. Alle materialparameterne er også undersøkt og diskutert.

En analyse av hulromvekst i polykarbonat ble utført ved hjelp av den implementerte materialmodellen, men en ukjent feil begrenset simuleringene til svært små deformasjoner. Analysen viste at polymerer opplever hulromvekst, men ikke i like stor grad som man finner i metaller. Dette skyldes antageligvis herdingen, som forårsakes av polymerkjedene ved høye tøyningsverdier.

Denne oppgaven viser at materialmodellen kan brukes til å modellere flere forskjellige polymerer, men at den også må videreutvikles for å ta høyde for kompleks geometri og store deformasjoner. Oppgaven har som mål å tilby alle NTNU-studenter nyttige tips om hvordan man utvikler og anvender brukermaterialmodeller for polymerer i Abaqus.

Preface

This Master's thesis is written as a part of the Master's program at department of Mechanical Engineering at NTNU in the Spring of 2014. The thesis involves Implementation of the Arruda-Boyce material model in Abaqus, and finite element analysis on polymers.

I would like to express my appreciation towards supervisor Professor Zhiliang Zhang for his consistent help and valuable discussions. Also, big thanks to my fellow student Kasper Sandal and his problem solving capabilities. Special thanks to the ones closest to me. My parents and my brother because they are always there to back me up, and my girlfriend for all her encouragement and support.

Contents

List of Figures **ix**

1 Introduction **1**
- 1.1 Motivation 1
- 1.2 Polymers 1
- 1.3 Ductile Fracture 3

2 Theory **5**
- 2.1 Large Deformation Mechanics 5
  - 2.1.1 Deformation Gradient 5
  - 2.1.2 Finding U, V and R 6
- 2.2 Arruda Boyce 7
  - 2.2.1 Overview 7
  - 2.2.2 Linear Elastic Spring 9
  - 2.2.3 Rubber Elasticity, "Langevin Spring" 10
  - 2.2.4 Viscoplastic Element 10

3 Implementation **13**
- 3.1 User Subroutines 13
- 3.2 VUMAT 13
- 3.3 Implementation of Arruda Boyce - Introduction 14
- 3.4 Layout of the VUMAT for Arruda-Boyce 15
  - 3.4.1 Midpoint Integration Scheme 15
  - 3.4.2 Flow Rule 17
  - 3.4.3 Elastic Stress 19
  - 3.4.4 Backstress 19
  - 3.4.5 Material Parameters in the Material Subroutine 20
  - 3.4.6 Error and Sub Increments 21
  - 3.4.7 Iteration 22
  - 3.4.8 Updating the Stress 23
- 3.5 Some Mathematical Theory and Numerical Approximations 23
  - 3.5.1 Hencky Strain - Padé Approximation 23

3.5.2 Exact Hencky strain . . . 24
3.5.3 Inverse Langevin . . . 25
3.6 Explicit and Implicit . . . 26
3.7 Using Abaqus and the User Subroutine . . . 27
3.8 Limitations with the User Material Subroutine . . . 30

4 Single Element Parameter Study 31
4.1 Introduction . . . 31
4.2 Uniaxial Test Setup . . . 33
4.3 Material Parameters . . . 36
4.3.1 Elastic Stress Response . . . 37
4.3.2 Backstress/Langevin Spring . . . 39
4.3.3 Viscoplastic Element . . . 41
4.4 Strain Rate . . . 44
4.5 Different Load Cases . . . 48
4.5.1 Tension and Compression . . . 48
4.5.2 Hysteresis . . . 49
4.6 Polymethylmethacrylate - PMMA . . . 51

5 Cell Model and Void Growth 53
5.1 Cell Model . . . 53
5.1.1 Introduction . . . 53
5.1.2 Cell Model . . . 53
5.1.3 Strain Ratio . . . 54
5.1.4 Void Analysis Results . . . 57

6 Conclusion and Suggestions for Further Work 65

References 67

Appendices

A Perturbation Method 69

B The User Material Subroutine - VUMAT 71

C Explanation to Subroutines Used in VUMAT 81

D Strain Triaxiality 83

List of Figures

1.1 Illustration of polymer behaviour 2
1.2 Ductile fracture 3

2.1 Illustration of the different elements in the Arruda-Boyce model 8

3.1 Block diagram of the VUMAT 16
3.2 Block diagram of the midpoint scheme 17
3.3 Block diagram of the flow rule 18
3.4 Block diagram of the elastic stress 19
3.5 Block diagram of the backstress 20
3.6 Block diagram of the error check routine 21
3.7 Block diagram of the iteration 23
3.8 The accuracy of the Padé approximation 25
3.9 The padé approximation fails at very high strain values 26
3.10 How to start the analysis 28
3.11 Successfull analysis 29

4.1 The geometry of the test specimen used in this chapter, and an illustration of how the specimen deforms. 31
4.2 An axisymmetric single element, with y symmetry. 32
4.3 Dimensions of the single element, displacement control. It is imposed a displacement E_y in the y-direction. 32
4.4 Stress-strain curve from an uniaxial test of Polycarbonate: Sections of interest: 1: elastic section, 2: yield point, 3: strain softening, 4: strain hardening 34
4.5 A physical view on how the polymer chains affect the material response. 35
4.6 A simple tension test of polycarbonate using axisymmetric elements and the VUMAT user subroutine. 36
4.7 Youngs modulus effect on the elastic section and the yield point. 37
4.8 How changing the Youngs modulus E, affects the material 38
4.9 Poissons ratio: close up at the elastic section with the yield point 38
4.10 The effect of setting $T^b = 0$ in the user material subroutine. 39

4.11 How changing the Rubber modulus, C_r, affects the material. 40
4.12 How the limited chain parameter N, affects the material 41
4.13 How the pre-exponential factor $\dot{\gamma}_0$, affects the material 42
4.14 How the Zero level activation energy A, affects the material 42
4.15 Illustration of how the preferred shear stress state ratio, $\frac{s_{ss}}{s_0}$, affects the material 43
4.16 Illustration of how the softening slope h affects the material 44
4.17 Illustration that shows the viscosity effects of the material 45
4.18 Close up at the yield point showing that it depends on the displacement rate. 45
4.19 Evolution of the yield stress at different displacement rates 46
4.20 Evolution of the yield stress at different values for $\dot{\gamma}_0$ 47
4.21 Strain history of the specimen. 48
4.22 Stress and strain history of polycarbonate at $1mm/s$. 49
4.23 Stress-strain history of polycarbonate at $1mm/s$. 50
4.24 Stress and strain history of the specimen. 50
4.25 Close up of the yield point at the stress-strain curve of a uniaxial test of Plexiglas 52
4.26 Stress-strain curve of Plexiglas at large deformations 52

5.1 The material can be approximated by a series of Hexagon prisms 53
5.2 Approximating the hexagon prism with a cylinder for symmetric reasons. 54
5.3 Using axisymmetry, the cell model can be greatly simplified 54
5.4 Displacement on the cell model 55
5.5 Dimensions of the cell model 56
5.6 Undeformed cell model 56
5.7 Deformed cell model with no imposed strain ratio. The force traction of the edge of the cylinder is zero. The color plot shows the most stressed areas using Mises stress. 57
5.8 Deformed cell model with strain ratio: $m = -15\%$ 58
5.9 Deformed cell model with strain ratio: $m = 0\%$ 58
5.10 Deformed cell model with strain ratio: $m = 10\%$ 59
5.11 Calculating the current void volume fraction using small rectangles and the cylindrical shell method. 59
5.12 It is clear that higher values of m means more rapid rise in the void volume fraction f 60
5.13 The stress in the y-direction (S22) is plotted against the stress in the r-direction (S11) 61
5.14 A purely elastic plastic material has slightly more void growth than polycarbonate in an uniaxial test 62
5.15 A purely elastic plastic material has significantly higher void growth than polycarbonate 63

5.16 Stress distribution and void shape of the Mises material at 20% strain and $m = 10\%$ 63

D.1 Stress in the y-direction vs strain in the y-direction at the given strain ratios 83
D.2 This figure illustrates the Mises stress vs strain in the y-direction 84
D.3 This figure illustrates the stress in the radial direction vs strain in the y-direction 84
D.4 This figure illustrates the strain at the given displacement rates 85
D.5 This figure illustrates the stress ratio at the given displacement rates . . 85

Chapter 1 Introduction

1.1 Motivation

The use of polymers in engineering application has grown considerably over the last few decades. It is important to find good methodologies of analysis in order to find the materials capabilities to withstand complex loads. Most polymers are ductile at room temperature and will fracture somewhat in the same manner as ductile metals. These stages are: nucleation, void growth and coalescence. The difference in ductile fracture between metals and polymers is the matrix material surrounding the impurities and voids.

In order to study how different geometries on the micro scale affect the material properties, a material model for the pure material must be developed. Several models for characterizing polymers have been proposed, and include the Arruda-Boyce Model, Bergstrom Boyce Model, Three Network Model and several others. None of these material models for polymers are yet built in to Abaqus (v6.12), which makes it hard to study the behavior of polymers. In order to analyse such materials, it is custom to utilize a user defined material subroutine. These subroutines are commercially available, but quite expensive.

The purpose of this paper is to present the Arruda-Boyce model and to show how to implement it as a user material subroutine for the commercial FEA-software Abaqus. The user material is also verified by doing some simple finite element analysis using different loading conditions and material properties. The paper will also show how to do simulations using the user material subroutines.

1.2 Polymers

The definition of polymers is a combination of two or more compounds called *mers*. The number of *mers* in a given molecule is called the degree of polymerization, and is for engineering polymers typically in the order of several thousands. These molecules are called polymer chains. Because of the molecular structure, polymers experience rate dependent viscoplastic deformation. In an undeformed polymer specimen, two neighboring polymer chains or different segments of a polymer chain folded back upon itself, encounter weak attractive Van Der Waals forces between the chain segments.

When a polymer specimen is deformed, these forces will try to prevent the separation of the polymer chains. The elastic resistance in polymers, or Youngs modulus, is generally several orders of magnitude lower than the elastic resistance of metals or ceramics. This is because the weak forces between the polymer chains are significantly weaker than primary bonds. When deforming a polymer specimen, the molecule chains will stretch and change orientation. At lower strain rates, the molecules have enough time to move, and the material becomes relatively easy to deform. However, at high strain rates, the molecules will experience excessive friction between both themselves and the neighboring molecules. Thus, highest stress is required to deform the material.

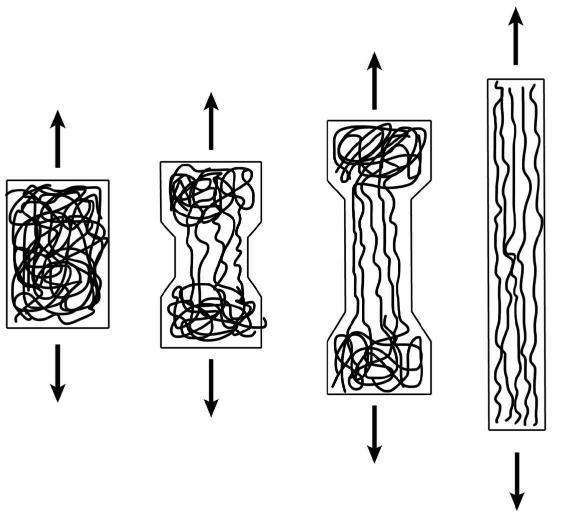


Figure 1.1: Illustration of the polymer deformation. The material yields when the polymer chains start to align, and the stress will increase significantly when most of the molecules are aligned.

When most of the polymer chains are stretched, the strength of the polymer rises significantly. This is because the polymer chains then are aligned and the reason for the resistance in the material is the primary bonds in the polymer chains. These forces are much higher than the weak Van Der Waals forces. The effect of the weak and strong forces is illustrated in figure 1.1

1.3 Ductile Fracture

All materials contain impurities such as particles or inclusions. In ductile materials these impurities are highly significant in the deformation process. The ductile fracture process involves three stages: void nucleation, void growth and coalescence. These stages are illustrated in figure 1.2. When a ductile material undergoes deformation, the matrix surrounding the impurities will debond, and the sphere that is created is then called a void. This process is the void nucleation. The mechanism depends on both the material properties of the surrounding matrix and the material properties of the particle. The next stage in the ductile fracture process is void growth, and studies show that the void growth is highly dependent on the hydrostatic stress in the material and in the initial shape of the void.

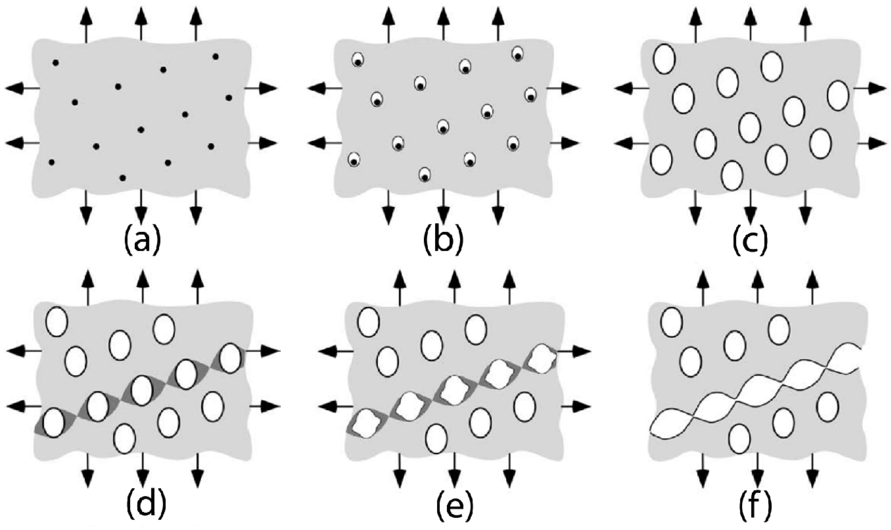


Figure 1.2: The illustration of the different mechanisms involved in ductile fracture. a: Impurities or inclusions in the material matrix. b: Void nucleation. c: Void growth. d: Strain localization between voids. e: Necking between voids and coalescence. f: Fracture. (Picture taken from "Fracture Mechanics, Fundamentals and Applications", T.L. Anderson[4])

In 1975, A.L Gurson [7] developed an analytical micro mechanical model for ductile void growth. It is by now understood that, deformation in combination with high hydrostatic pressure leads to a significant void growth in a ductile metal, and the Gurson model can, with high accuracy, predict the void growth. Voids reaching a critical size will experience necking between themselves and neighboring voids. This is when the original Gurson model breaks down, and the strength of the material drops significantly. This stage in the ductile fracture mechanism, is called void coalescence, and is the last stage before fracture. In 2000, Z.L. Zhang, C. Thaulow, and J, Ødegård [15] completed the Gurson model to account for void coalescence by including the Thomason-criteria.

In the recent years, there has been given a significant attention to the fracture mechanics of ductile polymers, and several attempts have been made on developing a "Gurson model for polymers". In order to accomplish this, it is important to understand the nucleation process, void growth and coalescence of polymers. The main goal of this paper has been to develop the material model for the surrounding polymer matrix. However, when the material model was completed, some simple finite element analyses have been done on a cell model containing an initial void. Only the void growth has been studied.

Chapter 2
Theory

2.1 Large Deformation Mechanics

2.1.1 Deformation Gradient

The mechanics of fully three-dimensional large strain deformation involve the use of the deformation gradient tensor, F. The Deformation gradient is a fundamental measure of large deformations in continuum mechanics. When an object moves from initial position to a deformed position, two parameters can change. The position might change and the object may be deformed. Consider an infinitesimal line segment dX. The initial position of the line segment is called X, and after the deformation the same line segment is called dx at the new position x. The deformation gradient describes the transition between dX to dx such that:

$$dx = F \cdot dX \tag{2.1}$$

Thus, the definition of F is given by:

$$F = \frac{dx}{dX} \tag{2.2}$$

or in Cartesian coordinates:

$$F_{ij} = \frac{\delta x_i}{\delta X_j} = \begin{bmatrix} \frac{\delta x_1}{\delta X_1} & \frac{\delta x_1}{\delta X_2} & \frac{\delta x_1}{\delta X_3} \\ \frac{\delta x_2}{\delta X_1} & \frac{\delta x_2}{\delta X_2} & \frac{\delta x_2}{\delta X_3} \\ \frac{\delta x_3}{\delta X_1} & \frac{\delta x_3}{\delta X_2} & \frac{\delta x_3}{\delta X_3} \end{bmatrix} \tag{2.3}$$

In strain-based constitutive laws, you can separate the strain increments in both elastic strain and inelastic strain like the following: $d\epsilon = d\epsilon + d\epsilon_i$. The same goes for the deformation gradient, but in a different way.

$$F_{total} = F_{elastic} \cdot F_{inelastic} \tag{2.4}$$

Equation 2.4 is based on the simple assumption, that for each increment, the first part of the deformation is purely inelastic, and the last part is purely elastic. Note that even for isotropic materials, F does not have to be symmetric. In several cases, we need to decompose the deformation gradient F into the rotation tensor R in combination with the right stretch tensor U, or the left stretch tensor V. This is called polar decomposition.

$$F = RU = VR \tag{2.5}$$

The rotation tensor from equation 2.5 is proper orthogonal, i.e $R^T R = det(R) = I$. Where I is the identity tensor. R is a measure of the local rotation of X. The initial line segment is either first stretched by U and then rotated by R, or it is first rotated by R and then stretched by V. The Rotation tensor R is the same for each case, but the right and left stretch tensors V and U are different. In chapter 2.1.2 it is explained how to find R and U. Equation 2.6 shows the decomposition of the total deformation gradient, F, in the Arruda-Boyce theory.

$$F = F^e F^i \tag{2.6}$$

F^e is the elastic deformation gradient and F^i is the inelastic deformation gradient, given by the relaxed configuration obtained by an elastic unloading to a stress-free state. The "rate of deformation gradient change", $\dot{F}$, is given by the velocity gradient, L, defined in equation 2.7.

$$\dot{F} = (\dot{F}F^{-1})F = LF \qquad \text{thus} \qquad L = \dot{F}F^{-1} \tag{2.7}$$

L is decomposed into:

$$L = D + W \tag{2.8}$$

where D is the rate of deformation, and W is the spin tensor. The elastic deformation gradient is restricted to being only the result of stretching. L can then be decomposed into:

$$L = L^e + F^e L^i {F^e}^{-1} \tag{2.9}$$

where

$$L^i = \dot{F}^i {F^i}^{-1} = D^i + W^i \tag{2.10}$$

D^i is the plastic rate of deformation and W^i is the plastic spin tensor. A standard assumption for isotropic materials is that $W^i = 0$ [8]. Thus, for an isotropic material:

$$\dot{F}^i = D^i F^i \tag{2.11}$$

The constitutive relationship used to find D^i is the heart of the Aruda-Boyce Theory, and is described in chapter 2.2.4.

2.1.2 Finding U, V and R

In order to find U, V and R it is necessary to calculate some eigenvalues. Calculations of eigenvalues are relatively easy for a symmetric matrix. It is custom to first obtain a symmetric matrix that contains all the information from the deformation gradient. The Cauchy-Green tensor given in equation 2.12 is by definition a symmetric matrix.

$$C = U \cdot U = F^T F \tag{2.12}$$

This also imposes that:

$$U = \sqrt{C} \tag{2.13}$$

Thus, we need to find the matrix square root of C. In order to do this, Cauchy-Green tensor has to be rotated until it only contains the principal values. The square root can then be found using the relationship in equation 2.14

$$\sqrt{C'} = \begin{bmatrix} \sqrt{c'_{11}} & 0 & 0 \\ 0 & \sqrt{c'_{22}} & 0 \\ 0 & 0 & \sqrt{c'_{33}} \end{bmatrix} \quad \text{where} \quad C' = \begin{bmatrix} c'_{11} & 0 & 0 \\ 0 & c'_{22} & 0 \\ 0 & 0 & c'_{33} \end{bmatrix} \tag{2.14}$$

for a matrix C' that only has values on the diagonal. Since C is a symmetric, positive finite tensor, it has a set of positive eigenvalues. Obtain the eigenvalues and the eigenvectors and make the eigenmatrix C' and the directionmatrix Q. The relationship is given by:

$$C = Q^T C' Q \tag{2.15}$$

where C' has the eigenvalues in the diagonal and zeros elsewhere. Q has the three eigenvectors in each column.

$$C' = \begin{bmatrix} eig_1 & 0 & 0 \\ 0 & eig_2 & 0 \\ 0 & 0 & eig_3 \end{bmatrix} \quad \text{and} \quad Q = \begin{bmatrix} \vdots & \vdots & \vdots \\ N_1 & N_2 & N_3 \\ \vdots & \vdots & \vdots \end{bmatrix} \tag{2.16}$$

We can now find U:

$$U = \sqrt{C} = Q^T \sqrt{C'} Q \tag{2.17}$$

and the rotation tensor is given by:

$$R = FU^{-1} \tag{2.18}$$

When R is found, V can easily be extracted from the deformation gradient:

$$V = FR^{-1} \tag{2.19}$$

2.2 Arruda Boyce

2.2.1 Overview

The Arruda-Boyce model [5] is a hyperelastic and viscoplastic constitutive model that is used to describe the mechanical properties of rubber and polymers. The model highly differs from constitutive models for metals on the fact that it does not have a clearly defined transition point between elastic and inelastic response. The yield point can change for different strain rates. The rheological model is described in figure 2.1. The Arruda-Boyce constitutive law is made from three elements:

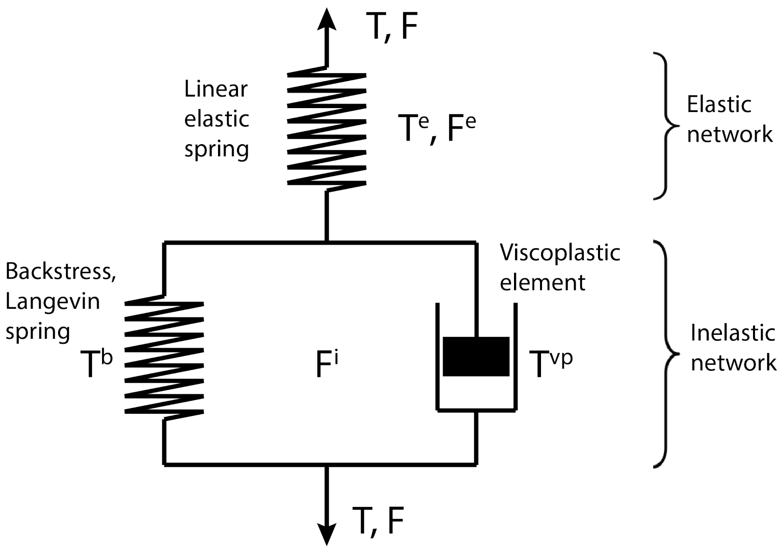


Figure 2.1: Illustration of the different elements in the Arruda-Boyce model

- A Linear elastic spring, in series with the two other elements in the inelastic network. This gives the initial linear elastic strain response.

- A viscoplastic element (Dampening). This gives rate dependence in all the loading stages.

- Rubber elasticity based on the eight chain model called the Langevin Spring. This element deals with hardening.

The Arruda-Boyce Model differs highly from most metallic models because the main variable used is the deformation gradient, F, rather than the strain increment, $d\epsilon$.

2.2.2 Linear Elastic Spring

The linear elastic spring characterizes the initial response as elastic using Hooks law to calculate the elastic strain. The stress in the linear elastic spring is the total stress. Since the material might experience excessive deformation, large strain theory and finite hyperelasticity is applied. The common Hooks stress-strain relation for isotropic infinitesimal deformation is described in eq 2.20

$$\sigma = \mathscr{L}(\varepsilon) \tag{2.20}$$

$\mathscr{L}$ is the fourth order elasticity tensor defined in equation 2.21.

$$\mathscr{L}_{ijkl} = \Lambda\delta_{ij}\delta_{kl} + G(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}) \tag{2.21}$$

With two elastic material constants, Λ and G, named after Gabriel Lamé:

$$\Lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} \tag{2.22}$$

and

$$G = \frac{E}{2(1+\nu)} \tag{2.23}$$

Where E is Youngs modulus and ν is Poissons ratio. For an isotropic material, considering symmetry in the stress tensor and the absence of initial stresses and strains [10] , Hooks law is defined as

$$\sigma_{ij} = \Lambda\varepsilon_{kk}\delta_{ij} + 2G\varepsilon_{ij} \qquad \Leftrightarrow \qquad \sigma = \Lambda tr(\varepsilon)I + 2G\varepsilon \tag{2.24}$$

Polymers experience large deformations, and ince ε is a measure for infinitesimal strain, it has to be replaced with the Hencky strain tensor, (a logarithmic strain tensor). The Cauchy stress tensor, σ, has to be replaced with the Kirchoff stress tensor τ. The new modified Hooks law for large deformations is then defined as:

$$\tau_{ij} = \Lambda h_{kk}\delta_{ij} + 2Gh_{ij} \qquad \Leftrightarrow \qquad \tau = \Lambda tr(h)I + 2Gh \tag{2.25}$$

The Hencky strain[14], h, can be calculated using equation 2.26

$$h = \frac{1}{2}ln(B) = ln(\sqrt{B}) \tag{2.26}$$

Where $B = FF^T$ is the Cauchy-Green tensor. The relationship between the Kirchoff stress tensor and the Cauchy stress tensor is given by: $\tau = J\sigma$, where $J = det[F]$. The general equation for Cauchy stress with large deformations for a purely elastic material is given by equation 2.27

$$\sigma = \frac{1}{J}[\Lambda tr(h)I + 2Gh] \tag{2.27}$$

Since the Arruda-boyce material is experiencing plastic deformation, only the elastic deformation gradient, F^e, is used in the calculation of the Cauchy elastic stress tensor T^e.

$$T^e = \frac{1}{det[F^e]}[\Lambda tr(ln(\sqrt{B^e}))I + 2Gln(\sqrt{B^e})] \tag{2.28}$$

Where B^e is the elastic Cauchy-Green tensor, and F^e is the elastic deformation gradient. See chapter 2.1.1 for more information.
Note: evaluating the Hencky strain requires the matrix logarithm in combination with the square root of a matrix or the decomposition of the elastic deformation gradient $F^e = V^e R^e$. None of these operations are easy, and will be explored in chapter 3.5.1 and 3.5.2

2.2.3 Rubber Elasticity, "Langevin Spring"

The nonlinear rubber elastic spring element deals with an anisotropic resistance to chain alignment. In this Master's thesis, the rubber elasticity spring, or Langevin spring, is based on the eight chain model of rubber elasticity [5].

The stress in the Langevin spring, T^b depends on the inelastic deformation, and is given by equation 2.29. The b stands for backstress.

$$T^b = \frac{1}{3}C_r\sqrt{N}\mathscr{L}^{-1}\left[\frac{\Lambda^p_{chain}}{\sqrt{N}}\right]\frac{\Lambda_i^{p2} - \frac{1}{3}I_1}{\Lambda^p_{chain}} \tag{2.29}$$

C_r is the rubber modulus term, a material constant which is a combination of chains per volume, n, the Boltzmann constant, κ and the absolute temperature, θ. C_r is defined in equation 2.30

$$C_r = n\kappa\theta \tag{2.30}$$

N is a statistical parameter related to the limiting value of chain stretch. $\mathscr{L}^{-1}$ is the inverse Langevin function, see chapter 3.5.3. Λ_i^p are the principal applied stretches given by the eigenvalues of the inelastic stretch tensor V^i. (see chapter 2.1.1)

$$I_1 = \Lambda_1^{p2} + \Lambda_2^{p2} + \Lambda_3^{p2} \tag{2.31}$$

Λ^p_{chain} is the stretch on any individual chain in the network given by equation 2.32

$$\Lambda^p_{chain} = \sqrt{\frac{trace(F^i F^{i^T})}{3}} \tag{2.32}$$

2.2.4 Viscoplastic Element

When the linear elastic stress T^e (chapter 2.2.2) and the backstress T^b (chapter 2.2.3) are obtained, it is possible to calculate the rate of shape change in the relaxed configuration, D^i. The viscoplastic stress, T^{vp} is given by the tensorial difference between the total stress T^e and the convected back stress on the nonlinear spring element.

$$T^{vp} = T^e - \frac{1}{J^e} F^e T^b F^{eT} \tag{2.33}$$

T^{vp} is the plastic driving stress state, which continues to activate plastic flow. Only the deviatoric part of T^{vp} leads to plastic deformation, and it is defined as:

$$T^{vp}{}_{dev} = T^{vp} - \frac{1}{3} tr[T^{vp}] \tag{2.34}$$

The effective equivalent shear strength, τ, is defined as

$$\tau = \sqrt{\frac{T^{vp}{}_{dev} : T^{vp}{}_{dev}}{2}} \tag{2.35}$$

The flow rule in the Arruda-Boyce theory is defined as

$$D^i = \dot{\gamma}^p N \tag{2.36}$$

N is a normalized tensor aligned with the deviatoric driving stress state. See equation 2.37

$$N = \frac{T^{vp}{}_{dev}}{\sqrt{2}\tau} \tag{2.37}$$

And $\dot{\gamma}^p$ is the plastic shear strain rate given by equation 2.38

$$\dot{\gamma}^p = \dot{\gamma}_0 exp\Big[- \frac{As}{\kappa\theta} \Big(1 - \Big(\frac{\tau}{s}\Big)^{\frac{5}{6}} \Big) \Big] \tag{2.38}$$

where $\dot{\gamma}_0$ and A is material constants. θ is the absolute temperature and κ the Boltzmann constant. Strain softening is modeled by taking the "athermal shear stress" s to evolve to a "preferred" state, s_{ss}. The rate of s is described in the next equation.

$$\dot{s} = h\Big(1 - \frac{s}{s_{ss}}\Big)\dot{\gamma}^p \tag{2.39}$$

Where h is the softening slope, a material constant.

Chapter 3 Implementation

3.1 User Subroutines

In Abaqus, a number of material models are already built in, and it keeps increasing its library every year, traditionally with a main focus on metals. Every model embedded in the Abaqus material library has to be highly stable, fast and accurate. Material models for polymers with large strain deformation and viscoplastic effects are not yet included in the library, thus they have to be implemented manually. Abaqus supports both explicit and implicit user material models: VUMAT and UMAT. It is always recommended to write an explicit algorithm (VUMAT) before trying to make an implicit algorithm (UMAT). This is because VUMAT does not require the calculation of the consistent tangent stiffness matrix (see appendix A).

3.2 VUMAT

Every VUMAT subroutine has to begin and end as shown [3]:

```
      subroutine vumat(
C Read only (unmodifiable)variables -
     1  nblock, ndir, nshr, nstatev, nfieldv, nprops, lanneal,
     2  stepTime, totalTime, dt, cmname, coordMp, charLength,
     3  props, density, strainInc, relSpinInc,
     4  tempOld, stretchOld, defgradOld, fieldOld,
     5  stressOld, stateOld, enerInternOld, enerInelasOld,
     6  tempNew, stretchNew, defgradNew, fieldNew,
C Write only (modifiable) variables -
     7  stressNew, stateNew, enerInternNew, enerInelasNew )
C
      include 'vaba_param.inc'
C
      dimension props(nprops), density(nblock), coordMp(nblock,*),
     1  charLength(nblock), strainInc(nblock,ndir+nshr),
     2  relSpinInc(nblock,nshr), tempOld(nblock),
     3  stretchOld(nblock,ndir+nshr),
     4  defgradOld(nblock,ndir+nshr+nshr),
     5  fieldOld(nblock,nfieldv), stressOld(nblock,ndir+nshr),
     6  stateOld(nblock,nstatev), enerInternOld(nblock),
     7  enerInelasOld(nblock), tempNew(nblock),
     8  stretchNew(nblock,ndir+nshr),
     8  defgradNew(nblock,ndir+nshr+nshr),
     9  fieldNew(nblock,nfieldv),
     1  stressNew(nblock,ndir+nshr), stateNew(nblock,nstatev),
     2  enerInternNew(nblock), enerInelasNew(nblock),
C
      character*80 cmname
C
      do 100 km = 1,nblock

         user coding

  100 continue
      return
      end
```

VUMAT, is a Fortran subroutine which implies that every variable used needs to be given a type and a dimension. Since VUMAT is an explicit user material, it contains a do loop, which calculates the stress in every integration point in the model. This means that for every time VUMAT is executed, the stress is updated for the whole model. All variables that are passed in for information are defined at the VUMAT by default. The meaning of each variable can be found in the documentation for user material subroutines[3].

All user coding has to be done where it says: "user coding". It is crucial that the VUMAT is built exactly this way, or the VUMAT will not work correctly. Exceptions are user defined subroutines (inside VUMAT) such as: transpose of a matrix, inverse of a matrix or calculations of eigenvalues. These small subroutines can be written at the end of the VUMAT code. This saves a lot of space in the VUMAT, because operations that are needed several times can then be called in the user coding.

3.3 Implementation of Arruda Boyce - Introduction

The principle with the explicit user material subroutine VUMAT is quite simple. Abaqus provides an initial strain state and a strain increment. VUMAT is a user defined explicit constitutive law that updates the stress to the correct value. For a simple theory like purely elastic, the VUMAT code can be written in about 10 lines. This Arruda-Boyce VUMAT however, is a code at roughly 1000 lines. Since the Arruda-Boyce Theory involves large deformations, deformation gradients are used instead of strain increments.

3.4 Layout of the VUMAT for Arruda-Boyce

The only information needed at the start of the increment, is the total deformation gradient, F_t, the inelastic deformation gradient, ${F^i}_t$, and the softening factor s_t. The only other information that is known about the increment is the time step, dt, and the deformation gradient at the end of the increment, F_{t+dt}. VUMAT is used to update the stress state at the end of the increment, T_{t+dt}. It is also necessary to store the updated inelastic deformation gradient ${F^e}_{t+dt}$ and the updated strain softening parameter s_{t+dt}. These will be used at the start of the next increment. See equation 3.1

$$(F_t, F_{t+dt}, {F^i}_t, s_t, dt) \quad \Rightarrow \quad (T_{t+dt}, {F^i}_{t+dt}, s_{t+dt}) \tag{3.1}$$

The elastic strain increment is not stored because it can easily be extracted from the total deformation gradient and the inelastic deformation gradient. $F^e = F{F^i}^{-1}$. The stress at the end of the increment, T_{t+dt}, is based on the elastic deformation gradient at the end of the increment, ${F^e}_{t+dt}$. Finding ${F^e}_{t+dt}$ is not an easy task, because the rate of F^e is not known. In order to find ${F^e}_{t+dt}$, it is necessary to first find the rate of the inelastic deformation gradient, $\dot{F^i}$, calculate the inelastic deformation gradient at the end of the increment, ${F^i}_{t+dt}$, and use that information to extract ${F^e}_{t+dt}$ from F_{t+dt}. $\dot{F^i}$ is found using the flow rule (chapter: 3.4.2).

The integration method used in this VUMAT is the midpoint Runge-Kutta method described in chapter: 3.4.1. The accuracy of the midpoint scheme is checked every time. If the accuracy is not sufficient, the increment must be divided into several smaller sub increments. The accuracy of the sub incrementation is continuously tested, and if the error gets too big, the sub incrementation process starts over again with even smaller increments. The outline of the VUMAT user subroutine is described in figure 3.1.

3.4.1 Midpoint Integration Scheme

This midpoint Runge Kutta scheme[9] is approximately a second order accuracy scheme. It provides an approximation for the inelastic deformation gradient and the softening parameter at the end of the increment. The midpoint scheme should generally give a more accurate result than a simple forward Euler scheme[8], which

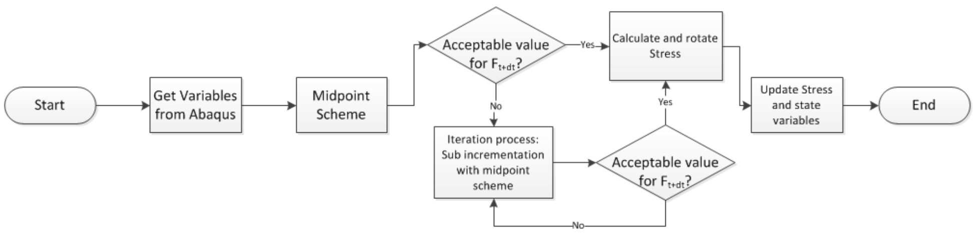

Start
Get Variables from Abaqus
Midpoint Scheme
Acceptable value for F_{t+dt}?
Yes
Calculate and rotate Stress
No
Iteration process: Sub incrementation with midpoint scheme
Acceptable value for F_{t+dt}?
Yes
No
Update Stress and state variables
End

Figure 3.1: Block diagram of the VUMAT

is a first order scheme. The principle of the Midpoint scheme for the Arruda-Boyce theory is explained in the equations 3.2 to 3.7.

$$(\dot{F^i}_t, \dot{s}_t) = f(F^e{}_t, F^i{}_t, s_t) \tag{3.2}$$

$$F^i{}_{t+\frac{dt}{2}} = F^i{}_t + \frac{dt}{2}\dot{F^i}_t \tag{3.3}$$

$$s_{t+\frac{dt}{2}} = s_t + \frac{dt}{2}\dot{s}_t \tag{3.4}$$

$$(\dot{F^i}_{t+\frac{dt}{2}}, \dot{s}_{t+\frac{dt}{2}}) = f(F^e{}_{t+\frac{dt}{2}}, F^i{}_{t+\frac{dt}{2}}, s_{t+\frac{dt}{2}}) \tag{3.5}$$

$$F^i{}_{t+dt} = F^i{}_t + dt\dot{F^i}_{t+\frac{dt}{2}} \tag{3.6}$$

$$s_{t+dt} = s_t + dt\dot{s}_{t+\frac{dt}{2}} \tag{3.7}$$

In order to get the elastic deformation gradient at the midpoint, it is necessary to find the total deformation gradient at the midpoint. This is done by taking the average of the deformation gradient at the start of the increment and at the end of the increment. See eq 3.8.

$$F_{t+\frac{dt}{2}} = \frac{F_t + F_{t+dt}}{2} \tag{3.8}$$

The flow rule is utilized two times for each run of the midpoint scheme. This is because it needs the inelastic deformation rate and the strain softening parameter rate at the start and in the middle. Consequently, this makes the midpoint scheme twice as computationally expensive as a Forward Euler scheme. The outline of the Midpoint scheme is explained in figure 3.2

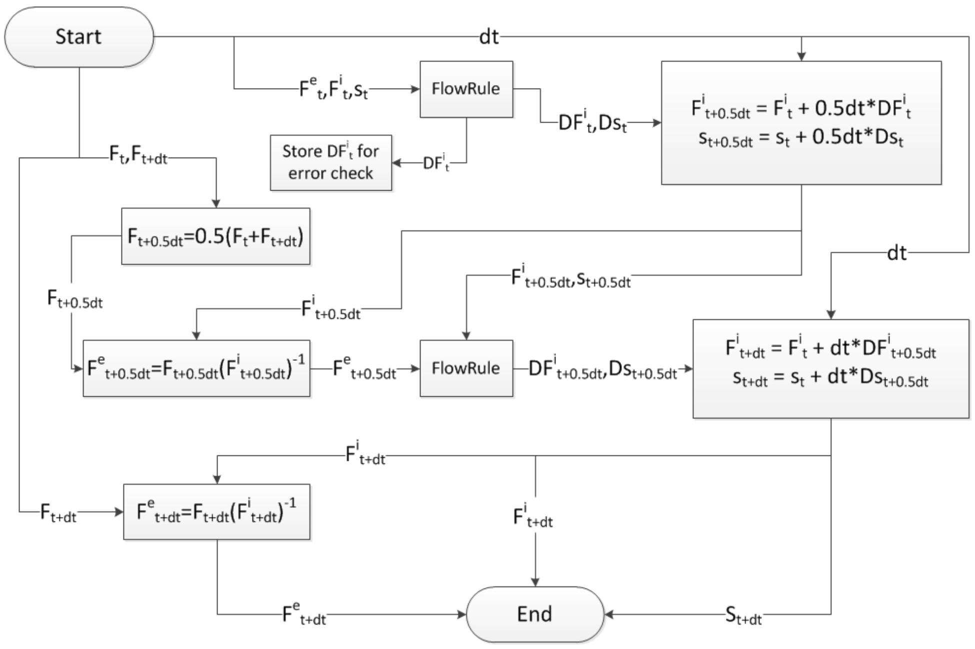


Figure 3.2: Block diagram of the midpoint scheme

3.4.2 Flow Rule

$$(\dot{F^i}_t, \dot{s}_t) = f({F^e}_t, {F^i}_t, s_t) \tag{3.9}$$

The flow rule is the most complicated operation in this subroutine and includes roughly all of the Arruda-Boyce theory. The equations used in the Flow Rule are described in chapter: 2.2.3, 2.2.2 and 2.2.4. When ${D^i}_t$ is found from equation 2.36, $\dot{F^i}$ is calculated using equation 3.10.

$$\dot{F^i} = D^i F^i \tag{3.10}$$

At the same time, it is necessary to find $\dot{s}$ using equation 2.39. The flow rule in combination with an integration scheme, provides an approximation for the updated elastic deformation gradient. The simplest form is the forward Euler method. An approximation for ${F^i}_{t+dt}$ and s_{t+dt} is given using equation: 3.11 and 3.12.

$${F^i}_{t+dt} = {F^i}_t + dt\dot{F^i}_t \tag{3.11}$$

$$s_{t+dt} = s_t + dt\dot{s}_t \tag{3.12}$$

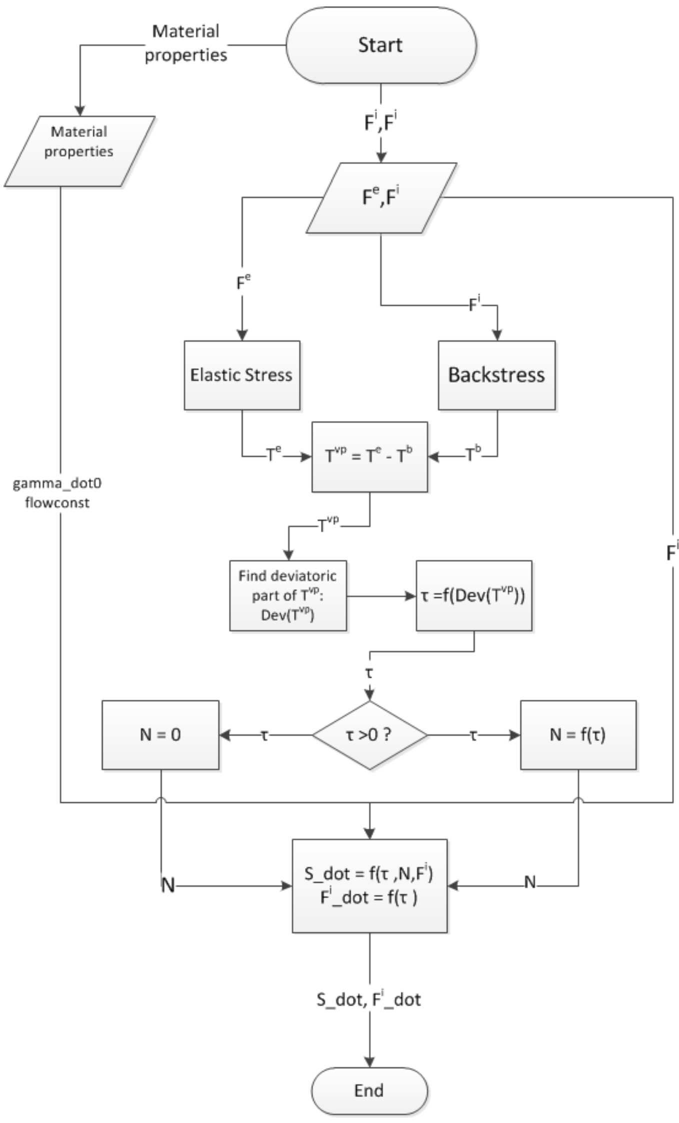


Figure 3.3: Block diagram of the flow rule

Since the flow rule contains some numerical approximations, it can produce some errors. But the main source of convergence errors in the VUMAT comes from the integration scheme. The flow rule is illustrated with a flowchart in figure 3.3. Notice that a decision has to be made on whether τ is equal to zero. If τ is equal to zero, the material response is purely elastic and the calculation of the direction vector N (see eq: 2.37) produces a singularity problem. The solution to this problem is simple: $\tau = 0 \quad \Rightarrow \quad N = 0_{3x3}$. "When there is no plastic deformation, the direction of the plastic deformation does not exist". The calculation of elastic stress and the backstress is embedded in the flow rule and will be further explored in chapter 3.4.3 and 3.4.4.

3.4.3 Elastic Stress

Calculation of the elastic stress or the total stress is done using only the elastic deformation gradient. The tricky part of the calculation is to extract the Hencky strain, h, from the deformation gradient. In other implementations of the Arruda-Boyce theory, this is normally done using a second order Padé approximation ([9],[6]). It is fast and accurate for deformations up to 100% strain. However, for more excessive deformations, the Padé approximation breaks down, and a better method must be used. Proposed in this thesis is an exact calculation of the Hencky strain using eigenvalues and eigenvectors of the elastic Cauchy-Green tensor B^e. It turns out that the extra computational power needed, does not effect the efficiency of the VUMAT. The method is explored in chapter 3.5.2. When the Hencky strain, h, is found, the calculation of the stress is done using Hooks law for large deformation. A flow chart explaining the calculation is illustrated in figure 3.4.

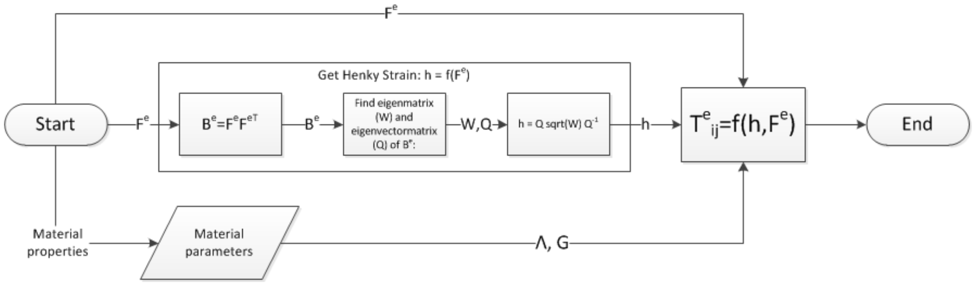


Figure 3.4: Block diagram of the elastic stress

Where the equation for $T^e{}_{ij}$ is given by eq: 2.28.

3.4.4 Backstress

The calculation of the backstress is based on the eight chain theory and the Langevin spring[5]. The equations used in this routine are explained in chapter 2.2.3. Similar to the calculation of the elastic stress, it is also required to calculate the eigenvalues of B^i. Moreover, the elastic deformation gradient is also used because the backstress is taken to be coaxial with the elastic deformation. A flow chart explaining the calculation is given in figure 3.5.

Where the equation for calculating T^b is given by eq 2.29

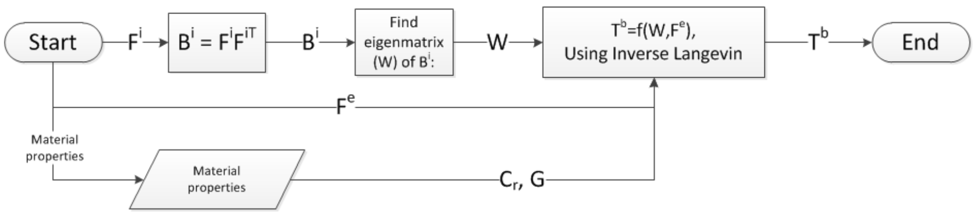


Figure 3.5: Block diagram of the backstress

3.4.5 Material Parameters in the Material Subroutine

The material properties needed in the user material subroutine are defined in the next table:

E	Youngs Modulus
ν	Poissons ratio
$\dot{\gamma}_0$	Pre-exponential factor
h	Softening slope
$\frac{s_{ss}}{s_0}$	Preferred shear stress state ratio
A	Zero stress level activation energy
n	Number of chains per unit volume
N	Limiting chain stretch parameter
θ	Absolute temperature

Many of the material parameters are combined as shown in the following table:

Combination of constants	Values	Description
G	$\frac{E}{2(1+\nu)}$	Elastic shear modulus
Λ	$\frac{E\nu}{(1+\nu)(1-\nu)}$	Cross modulus
C_r	$n\kappa\theta$	Rubber modulus term
s_0	$0.077\frac{G}{1-\nu}$	Initial athermal shear strength
s_{ss}	$\frac{s_{ss}}{s_0} * s_0$	Preferred shear stress state
$Flowconst$	$\frac{A}{\theta\kappa}$	*For simplicity

To save computational power and simplify the user material subroutine, all parameters and combination of parameters needed for the calculations are combined in an $1x8$ vector *mprop*.

$$mprop = [G, \Lambda, C_r, N, \dot{\gamma}_0, h, s_{ss}, flowconst] \tag{3.13}$$

3.4.6 Error and Sub Increments

Since the correct analytical solution to the problem is not known, it can not be used to check the accuracy of the midpoint integration scheme. The method used in this VUMAT is to see if the time step is small enough to give a good approximation. This is done by calculating the "weighted difference" between the midpoint scheme and a simple forward Euler scheme:

$$(\dot{F^i}_t, \dot{s}_t) = f(F^e{}_t, F^i{}_t, s_t) \tag{3.14}$$
$$F^i{}_{t+dt,Euler} = F^i{}_t + dt\dot{F^i}_t \tag{3.15}$$
$$\tag{3.16}$$

The difference between $F^i{}_{t+dt}$ and $F^i{}_{t+dt,Euler}$ scales with dt^2 [9]

$$Error = F^i{}_{t+dt} - F^i{}_{t+dt,Euler} \approx 0_2 dt^2 \tag{3.17}$$

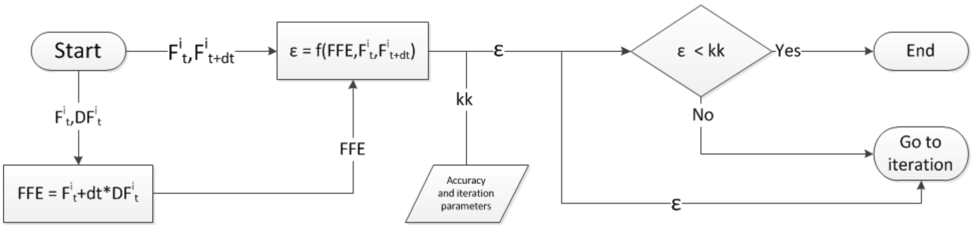


Figure 3.6: Block diagram of the error check routine

The difference between the midpoint integration solution and the initial deformation gradient, or step length scales with dt

$$Step = F^i{}_{t+dt} - F^i{}_t \approx 0_1 dt \tag{3.18}$$

The weighted error ϵ can then be be calculated using equation 3.19

$$\epsilon = \frac{Error}{Step} = \frac{||F^i{}_{t+dt} - F^i{}_{t+dt,Euler}||_2}{||F^i{}_{t+dt} - F^i{}_t||_2} = \alpha dt \tag{3.19}$$

where $|| * ||_2$ is the Frobenius norm. $\alpha = \frac{\epsilon}{dt}$ is a constant that is used in order to find a suitable sub time step.

3.4.7 Iteration

When the error ϵ has been found, it is possible to propose a new time step. In order to keep the solution stable, it is preferable to keep ϵ less than a prescribed constant c. This VUMAT has used $c = 0.01$. The new time step dt_{new} is proposed in equation 3.23 using the original time step, dt, the accuracy constant, c, and the weighted error, ϵ.

$$\begin{aligned} \epsilon_{new} &< c && (3.20)\\ \alpha * dt_{new} &< c && (3.21)\\ dt_{new} &< dt * \frac{c}{\epsilon} && (3.22)\\ dt_{new} &= b * dt\frac{c}{\epsilon} && (3.23) \end{aligned}$$

b can be chosen to have any value in the domain: $0 < b < 1$. The default value in the user material is: $b = 0.95$, in order to make sure that the new time step is lower than $dt * \frac{k}{\epsilon}$. When the new time step is calculated, the number of iterations is given by equation 3.24

$$iter = ceil(\frac{dt}{dt_{new}}) \tag{3.24}$$

Where $ceil(*)$ means rounding up to the next integer. This means that the new time step has to be slightly changed in order to reach dt after $iter$ iterations. $dt_{new} = \frac{dt}{iter}$. The deformation gradients F^e and F^i have to be updated at the start of each sub increment. In the first sub increment, they get the same value as the deformation gradients at the start of the main increment. The updated deformation gradients after each sub increment are kept, and used as the start of the next sub increment. The same goes for the softening parameter s. The exception is the total deformation gradient at the end of each sub increment. It is always calculated using the total deformation gradients in the main increment, the number of iterations, $iter$, and the iteration counter k. This is more easily explained in figure 3.7.

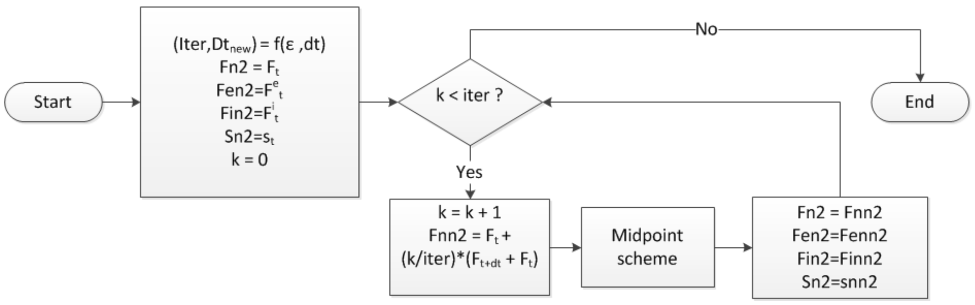


Figure 3.7: Block diagram of the iteration

3.4.8 Updating the Stress

When the approximation for F^i_{t+dt} is acceptable, the stress can be calculated:

$$F^e_{t+dt} = F_{t+dt} {F^i_{t+dt}}^{-1} \tag{3.25}$$

$$T_{t+dt} = T^e_{t+dt} = f(F^e_{t+dt}) \tag{3.26}$$

Moreover, the stress has to be rotated using the rotation tensor, R, shown in 2.1.2. The rotation is given by equation 3.27. Since this is a VUMAT and not a UMAT, the stretch tensor, U, is available at the start of the increment, and R can easily be calculated by: $R = FU^{-1}$. Then, F^i_{t+dt} and s_{t+dt} are stored in the state variables.

$$T_{rotated} = RTR^T \tag{3.27}$$

3.5 Some Mathematical Theory and Numerical Approximations

Since user subroutines for Abaqus have to be written in Fortran, routines for common tasks like an eigenvalue problems or matrix logarithm have to be developed.

3.5.1 Hencky Strain - Padé Approximation

Evaluating the Hencky strain is necessary to find the elastic stress T^e. The most fundamental definition of the Hencky strain is given by $h = ln(V^e)$, but because of the relationship, $F^e = V^e R^e$, it is possible to avoid the polar decomposition. See equation 3.28

$$B^e = F^e F^{eT} = V^e R^e (V^e R^e)^T = V^e (R^e R^{eT}) V^{eT} = V^e V^{eT} = V^{e2} \tag{3.28}$$

Consequently:

$$\begin{aligned} V^e &= \sqrt{B^e} && (3.29)\\ ln(V^e) &= ln(\sqrt{B^e}) && (3.30)\\ ln(V^e) &= \frac{1}{2}ln(B^e) && (3.31) \end{aligned}$$

In this paper, the Hencky strain is always calculated using the elastic Cauchy-Green tensor B^e. Calculation of the matrix logarithm analytically will always require the eigenvalues and eigenvectors. Since that calculation is not built in to the Fortran environment, they are often avoided for simplicity and to keep computational costs down. Traditionally, this is done by using a higher order Padé approximation ([9],[8],[10]). The Padé approximation is explained in equation 3.32 and equation 3.33.

$$\epsilon = ln(\sqrt{B}) = \frac{1}{2}ln(B) = Pad\acute{e}(B) \tag{3.32}$$

The approximation is given by:

$$Pad\acute{e}(X) = 3(X+I)(X-I)\cdot(2X^2+8X+2I)^{-1} \tag{3.33}$$

Where X is a $3x3$ matrix and I is the $3x3$ identity matrix. A stress strain curve is shown in figure 3.8 where the padè approximation is compared to an analytical solution of the Hencky strain (chapter 3.5.2). The two curves are approximately identical for strains even up to 100%. This Padè approximation is very accurate for small deviations from the identity matrix.

3.5.2 Exact Hencky strain

The Padé approximation works fine for deformations up to 100% strain. However, when trying to analyze a cell model with a spherical void, the analysis crashed, and the error messages given from Abaqus were either "massive distortion" or "floating point error". The problem seemed to come from strange behavior in the linear elastic spring. Much effort in this master thesis has been on finding a solution for this problem. It turned out that some elements in the cell model did experience extremely large deformations, and the Padè approximation did not give a correct value for the Hencky strain. It was then decided to try calculating the Hencky strain analytically. If B^e is a diagonalizable matrix, the logarithm and the square root of B^e can be calculated equation 3.34.

$$h = Q\begin{bmatrix} 0.5\sqrt{eig(B^i)_1} & 0 & 0\\ 0 & 0.5\sqrt{eig(B^i)_2} & 0\\ 0 & 0 & 0.5\sqrt{eig(B^i)_3} \end{bmatrix}Q^{-1} \tag{3.34}$$

Where $eig(B^e)_k$ are the eigenvalues of B^e and Q is the eigenvector matrix of B^e. Se chapter: 2.1.2 for more information regarding the eigenvectors. The routine used

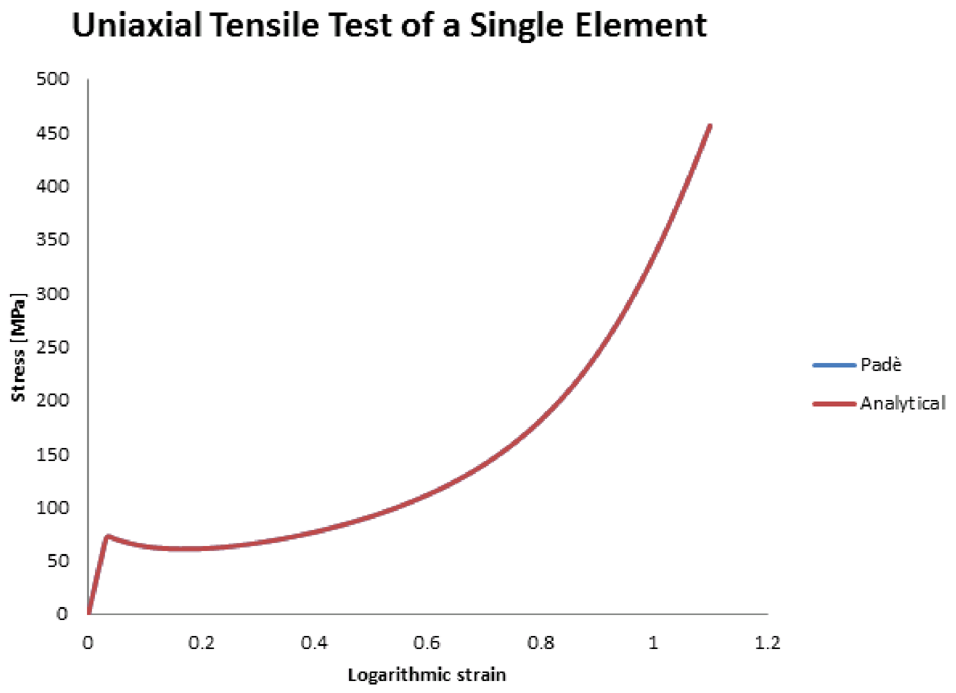


Figure 3.8: The accuracy of the Padé approximation

to calculate the eigenvalues and eigenvectors is taken from Joachim Kopp 2008 [12] and [11]. Equation 3.34 requires that the sum of dimensions of its eigenspace must be equal to 3. This might not always be the case, but it is still a more accurate calculation of the Hencky strain than the Padé approximation. Notice that the simple relationship $0.5 * ln(X) = log(\sqrt{X})$ is used to avoid taking the square root. The effect of calculating the Hencky strain in this manner is illustrated in figure 3.9.

Calculating the Hencky strain using this method has proven to give better results at very high strains, and the extra computational time needed compared to the Padé approximation is actually negligible. This could be due to the subroutine, optimized for $3x3$ matrices, used for calculating the eigenvalues. However, after even more extreme strain values, this method also fails and the slope of the stress-strain curve becomes negative. It might be that B^i is not always a diagonalizable matrix.

3.5.3 Inverse Langevin

Another approximation that is used in Arruda-Boyce is the inverse Langevin function. The Langevin function is quite simple: $\mathscr{L}(x) = coth(x) - \frac{1}{x}$, but the inverse $\mathscr{L}^{-1}(x)$

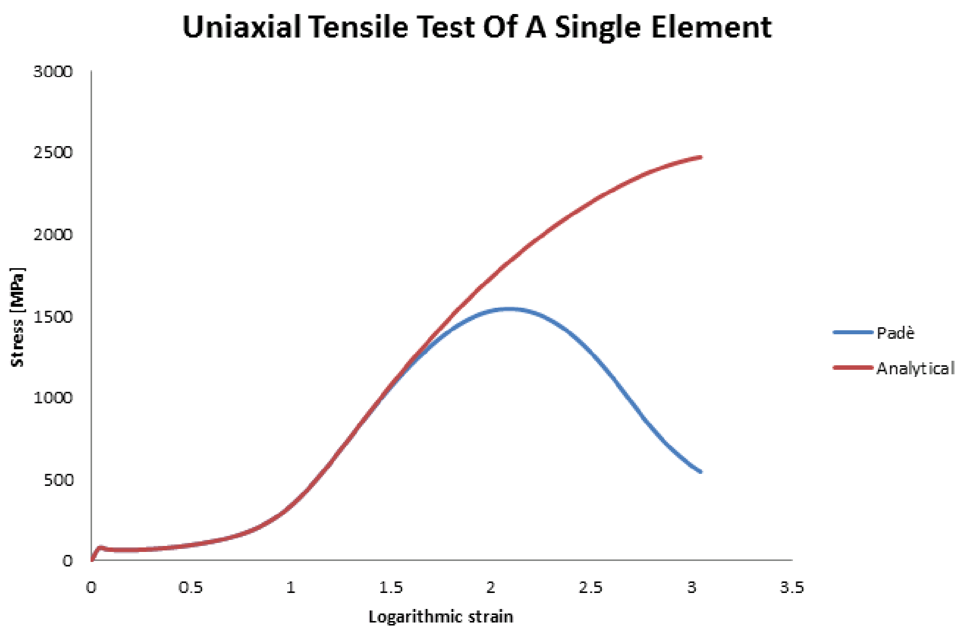


Figure 3.9: The padé approximation fails at very high strain values

must be approximated and is given by equation 3.35.

$$\mathscr{L}^{-1}(x) = x\frac{a + bx^2}{1 - x^2} \tag{3.35}$$

where

$$a = 2.99248834685337 \tag{3.36}$$

and

$$b = -1.14365108190676 \tag{3.37}$$

3.6 Explicit and Implicit

It is custom to write an explicit subroutine before trying to make an implicit, simply because the implicit subroutine is more complex. In an implicit user material subroutine, each node in the model depends on the neighboring nodes. In an explicit user material subroutine, each node is independent. In principle, the only extra computations needed when making an implicit user material is the calculation of the consistent tangent stiffness matrix: $\frac{\delta\Delta\sigma}{\delta\Delta\epsilon}$. Because of the complexity of the Arruda-Boyce Model, an analytical solution is not an option. If the stress is calculated correctly, it is always possible to calculate the consistent tangent stiffness matrix using a perturbation method. This method is explored in appendix A.

3.7 Using Abaqus and the User Subroutine

In order to run an Abaqus analysis with a user material subroutine, a functional combination of Abaqus, Intel Fortran and Microsoft Visual studio has to be installed on the computer. Running the analysis from within the Abaqus interface has proven to be troublesome. The procedure used during this master project is the following section:

Make the model in the Abaqus interface, and define a random material. Be sure to specify some kind of density, or else a dynamic explicit analysis will not run. Choose "dynamic explicit" in the step module. When the model is finished with an appropriate mesh, create a job and check that the analysis is successful with the simple material definition. If it is successful, make an input file. Put the input file, for example, "model.inp", and the user subroutine file, "ABoyce.for", in the same folder. The input file has to be modified manually in order for the user material to work. Open the input file with Microsoft visual studio or a simple text editor. Scroll down to the "ASSEMBLY" header, and change the material definition to:

```
material=ABoyce
```

Where "ABoyce" is the name of the user material subroutine. Then scroll down to the "MATERIALS" header and change the whole section to the following:

```
** MATERIALS
**
*Material, name=ABoyce
*Density
1.2e-9,
*User Material, constants=1
1.,
*DEPVAR
6
**
```

The density of the material has to be specified in the input file. The reason for the apparently low density is that the "ABoyce" subroutine uses mm and not m as standard length dimension (In order to get stress in MPa). This means that the material definition states that $\rho = 1.2 * 10^{-9} kg/m^3 = 1.2 kg/mm^3$. "DEPVAR" needs to have the value 6 or greater, because 5 of the inelastic deformation gradient elements have to be stored, in addition to the strain softening parameter. By standard, Abaqus divides the step into 20 intervals, such that only 20 (x,y) values are available for plotting. In order to get more intervals, go to the "OUTPUT" header in the input file and change the line:

```
"Output,␣variable=preselect"
```

into

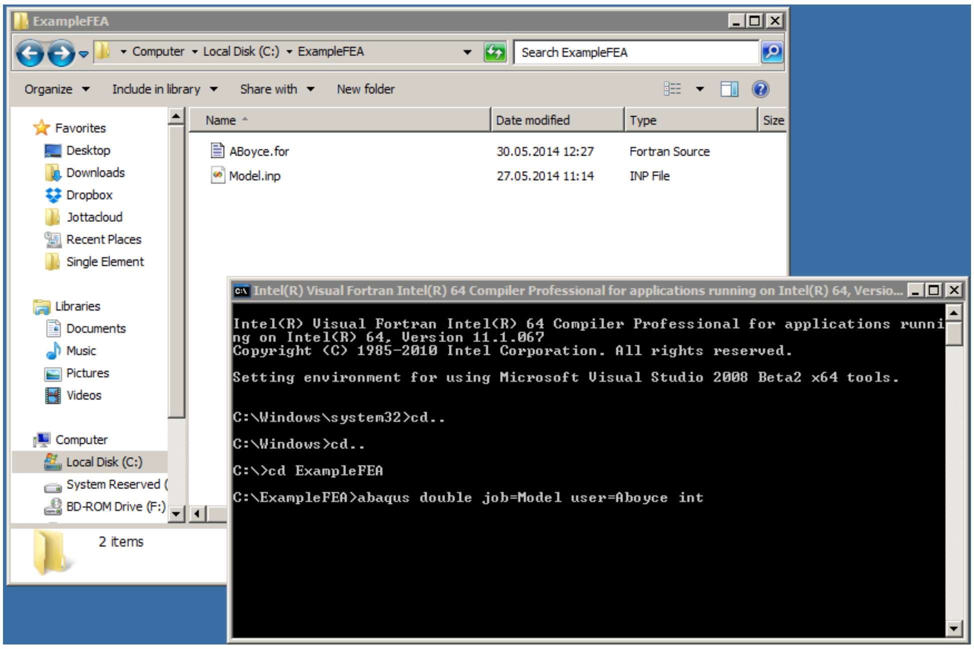
```
"Output,␣number␣interval=100,␣variable=preselect"
```

This will increase the number of intervals to 100. Save the input file.

Normally, running a finite element analysis with the Abaqus engine is done using the "Abaqus command" environment, but then user materials are not supported. In order to use the user material, Open "Fortran Build Environment for applications running on Intel(R) 64". Make sure to be in the same directory as the input file and the user subroutine. (Use simple commands such as "dir" and "cd"). When in the right directory, type the following:

```
"abaqus␣double␣job=Model␣user=Aboyce␣int"
```

see figure 3.10 for more information. The different terms are explained as:



Name	Date modified	Type	Size
ABoyce.for	30.05.2014 12:27	Fortran Source	
Model.inp	27.05.2014 11:14	INP File	

```
Intel(R) Visual Fortran Intel(R) 64 Compiler Professional for applications runni
ng on Intel(R) 64, Version 11.1.067
Copyright (C) 1985-2010 Intel Corporation. All rights reserved.

Setting environment for using Microsoft Visual Studio 2008 Beta2 x64 tools.


C:\Windows\system32>cd..

C:\Windows>cd..

C:\>cd ExampleFEA

C:\ExampleFEA>abaqus double job=Model user=Aboyce int
```

Figure 3.10: How to start the analysis

– "abaqus double" -> start the Abaqus solver with double precision

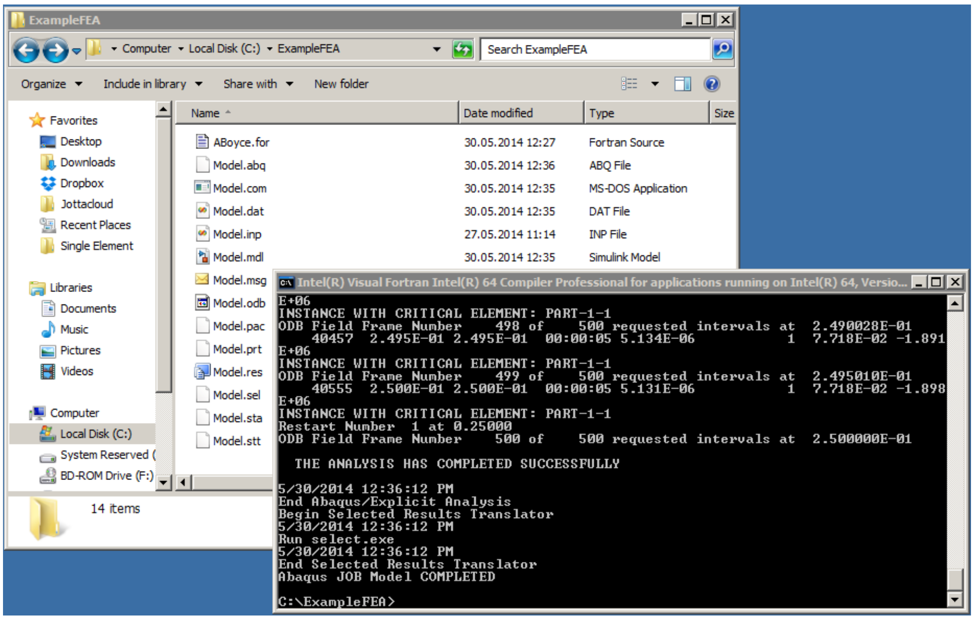
– "job=Model" -> specify the input file. In this case the "Model.inp" file.

– "user=Aboyce" -> specify the user subroutine. In this case the "ABoyce.for" file.

– "int" -> Interactive modus. Monitor the analysis and get eventual error messages.

Press enter to start the analysis.

When the analysis is finished, the results are stored in the odb-file: "Model.odb". Open this file in Abaqus and view the results. See figure 3.11 for more information. For models with high numbers of elements, it is possible to speed up the analysis



Name	Date modified	Type	Size
ABoyce.for	30.05.2014 12:27	Fortran Source	
Model.abq	30.05.2014 12:36	ABQ File	
Model.com	30.05.2014 12:35	MS-DOS Application	
Model.dat	30.05.2014 12:35	DAT File	
Model.inp	27.05.2014 11:14	INP File	
Model.mdl	30.05.2014 12:35	Simulink Model	
Model.msg			
Model.odb			
Model.pac			
Model.prt			
Model.res			
Model.sel			
Model.sta			
Model.stt			

```
E+06
INSTANCE WITH CRITICAL ELEMENT: PART-1-1
ODB Field Frame Number    498 of    500 requested intervals at  2.490028E-01
    40457  2.495E-01 2.495E-01  00:00:05 5.134E-06             1  7.718E-02 -1.891
E+06
INSTANCE WITH CRITICAL ELEMENT: PART-1-1
ODB Field Frame Number    499 of    500 requested intervals at  2.495010E-01
    40555  2.500E-01 2.500E-01  00:00:05 5.131E-06             1  7.718E-02 -1.898
E+06
INSTANCE WITH CRITICAL ELEMENT: PART-1-1
Restart Number  1 at 0.25000
ODB Field Frame Number    500 of    500 requested intervals at  2.500000E-01

  THE ANALYSIS HAS COMPLETED SUCCESSFULLY

5/30/2014 12:36:12 PM
End Abaqus/Explicit Analysis
Begin Selected Results Translator
5/30/2014 12:36:12 PM
Run select.exe
5/30/2014 12:36:12 PM
End Selected Results Translator
Abaqus JOB Model COMPLETED

C:\ExampleFEA>
```

Figure 3.11: Successfull analysis

using multiple processors or CPU-cores. The workstation used in this master project had 8 cores, which allowed the use of 7 cores for the analysis. Notice that the model has to contain at least n elements in order to use n CPU-cores. Using multiple cores is done by changing the command given in the Fortran terminal into:

```
abaqus double job=Model cpus=7 user=Aboyce int
```

This will speed up the analysis by a factor of approximately 7.

3.8 Limitations with the User Material Subroutine

Every analysis of a single element has proven successful using the user material subroutine. However, when analyzing more complex geometries, such as a cell model with a void, the user material subroutine tends to fail at fairly small global deformations. Much of the time spent implementing the Arruda-Boyce theory has been focused on making the user subroutine stable for different geometries. Normal error messages are "excessive distortion in element" and "Floating point error". The first error message can occur even when using the material models built in to Abaqus. When the second error message appears however, its root comes almost always exclusively from the user material subroutine. It means that some variables are not properly defined. Common reasons for this could be dividing by zero or that a variable goes towards infinity for some reason. When analyzing voids in chapter 5, the analysis constantly crashed. The source of this error is still not fully understood, but it seems that problem rises in elements with extremely large deformations, and that the sub incrementation is not working correctly, i.e. smaller time steps do not increase the accuracy of the analysis. This can perhaps be solved using a better iterative integration method.

Chapter 4

Single Element Parameter Study

4.1 Introduction

In order to study the Arruda-Boyce model implemented in the VUMAT user material, a simple tension test of a cylinder specimen has been considered. The cylinder has a radius of $15mm$ and a height of $30mm$. The cylinder is tested for uniaxial load with displacement control as described in figure 4.1

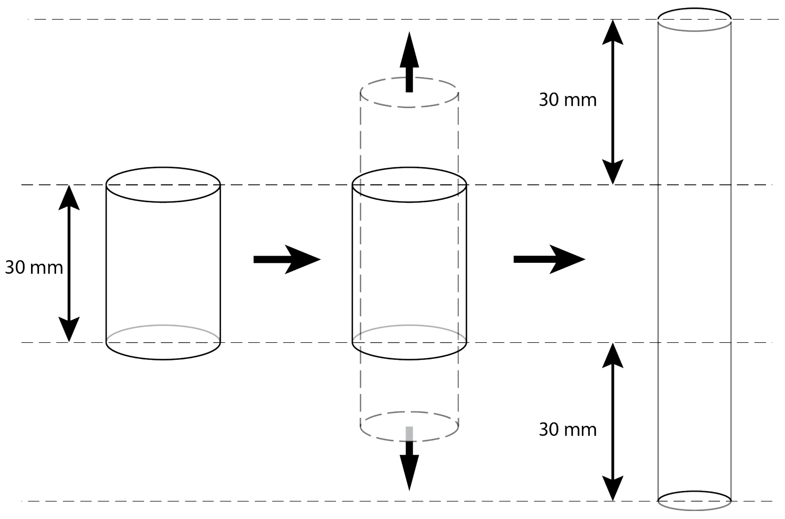


Figure 4.1: The geometry of the test specimen used in this chapter, and an illustration of how the specimen deforms.

For simplicity and due to geometry, the cylinder can be modeled as a single axisymmetric element. It is always recommended to test the user material using a single element before analyzing a more complex problem. The element is described in figure 4.2 and 4.3.

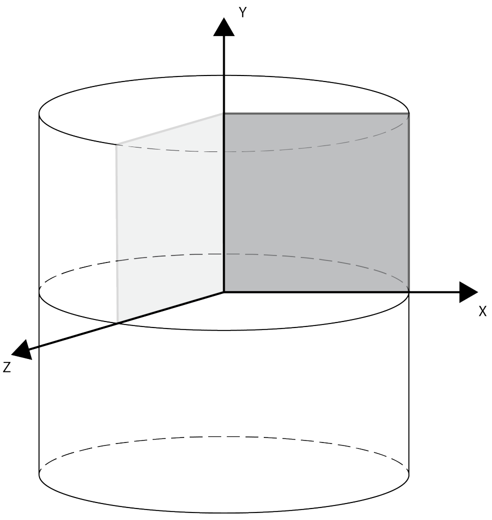


Figure 4.2: An axisymmetric single element, with y symmetry.

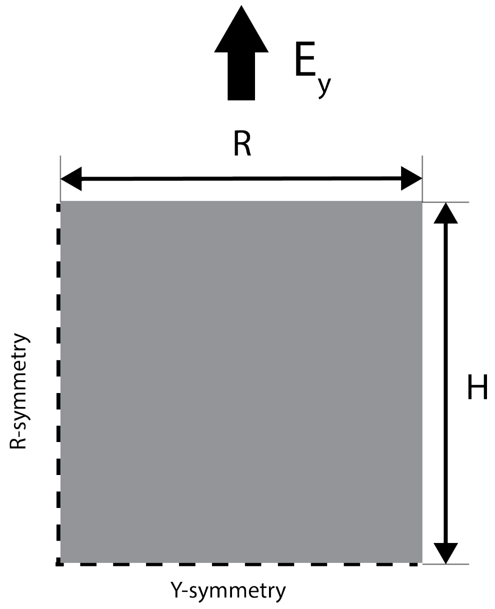


Figure 4.3: Dimensions of the single element, displacement control. It is imposed a displacement E_y in the y-direction.

The single axisymmetric element has dimensions: $H = R = 15mm$ and is imposed a displacement E_y. Since the Arruda-Boyce material is a viscoplastic material, it is of course strain rate dependent, and analysis with constant deformation ratios are preferred. The element is always deformed with displacement control using a linear ramp function. and the rate is given by equation 4.1

$$\dot{u} = \frac{u_{tot}}{dt} \tag{4.1}$$

where u_{tot} is the total displacement, and dt is the total time step for the increment. Analysis of the single element is divided into 3 sections. In the first section, the viscosity effects of the material model will be explored. In the second section, the material properties will be studied. In the last section, different loading conditions will be explored.

4.2 Uniaxial Test Setup

The axisymmetric element is tested with several variations in the material properties, several displacement rates and different types of loading conditions. For every simulation of the single element in this thesis, the magnitude of the displacement rate is constant: $1mm/s$. The only exception is the strain rate study. The maximum displacement is never over $30mm$. For most studies, the total time is then given by $\frac{30mm}{1mm/s} = 30s$. Figure 4.4 shows a simple stress-strain curve using the values for polycarbonate, and three sections and a point of interest have been marked.

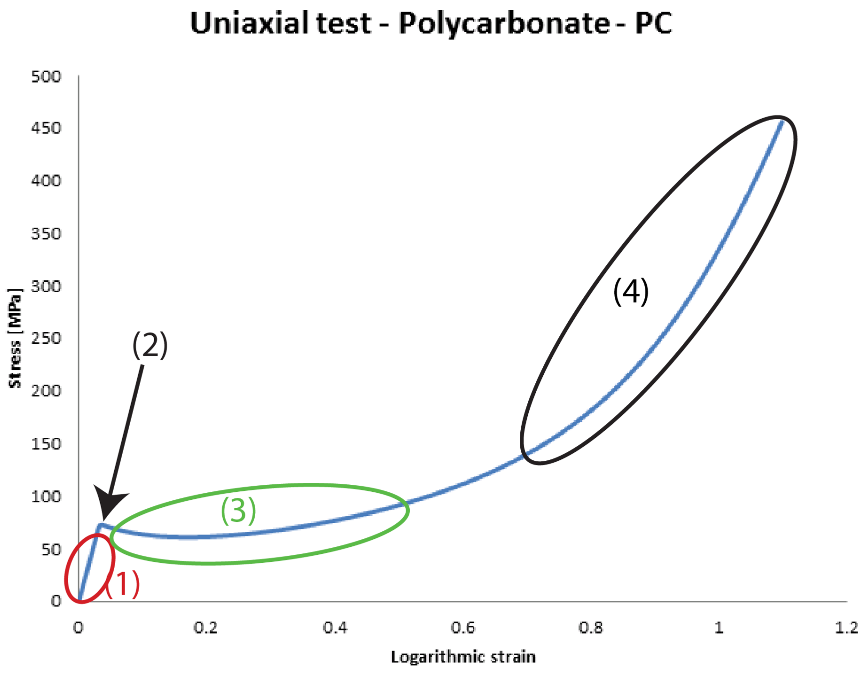


Figure 4.4: Stress-strain curve from an uniaxial test of Polycarbonate: Sections of interest: 1: elastic section, 2: yield point, 3: strain softening, 4: strain hardening

The first section(1), is the elastic stress section. All deformation that takes place in this area is reversible. This means that the total deformation gradient F will be equal to the elastic deformation gradient F^e. The slope of this curve should therefore be highly dependent on the parameters controlling the elastic stress response (chapter 4.3.1). The second section(2) of interest is not really a section, but a point. This is the transition point between the elastic section and the elastic-plastic section, the yield point(3). Much of the deformation that occurs after this point is irreversible. After the yield point, the third section begins and the slope of the curve becomes negative. This phenomenon is called strain softening. The more deformed the specimen gets, the weaker it gets. A material that only experiences strain softening after yield will experience excessive necking in a tension test. However, after some more deformation, the slope of the curve gets positive again, and the material undergoes strain hardening(4). This combination of strain softening and strain hardening has an interesting effect on how a uniaxial test specimen behaves.

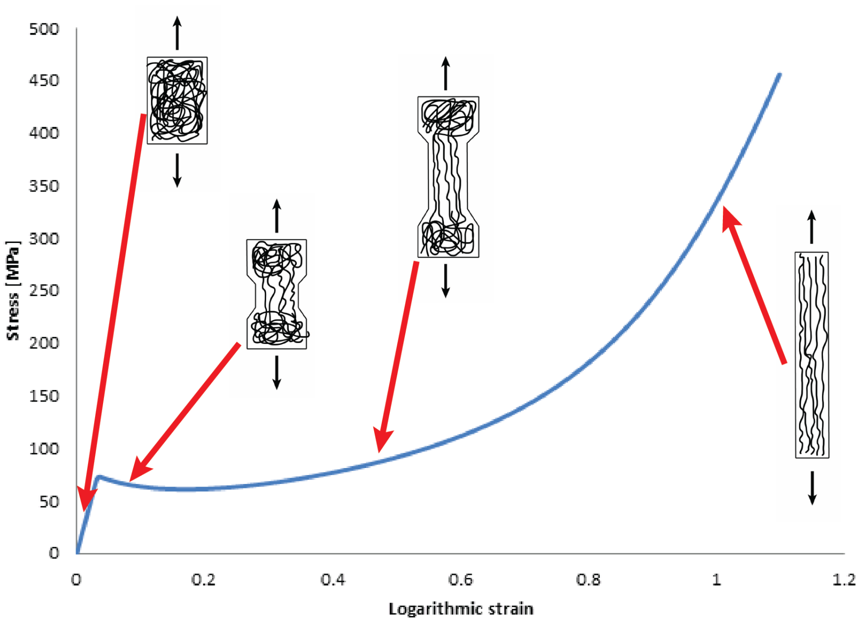


Figure 4.5: A physical view on how the polymer chains affect the material response.

Consider a cylinder with constant cross section. First it will show slight signs of necking, but then the strained part becomes stronger than the surrounding material and the necking "spreads" throughout the specimen. When the whole specimen has reduced its cross section, it will continue its deformation evenly in the whole specimen. This effect is shown in figure 4.5, where it is related to the stress strain curve. In order to show that the Arruda-Boyce material behaves in this way, a simple uniaxial test specimen was stretched with displacement control. This was done using a dynamic explicit finite element analysis with the user material subroutine developed in this master project. A screen shot collage from the FEA is displayed in figure 4.6

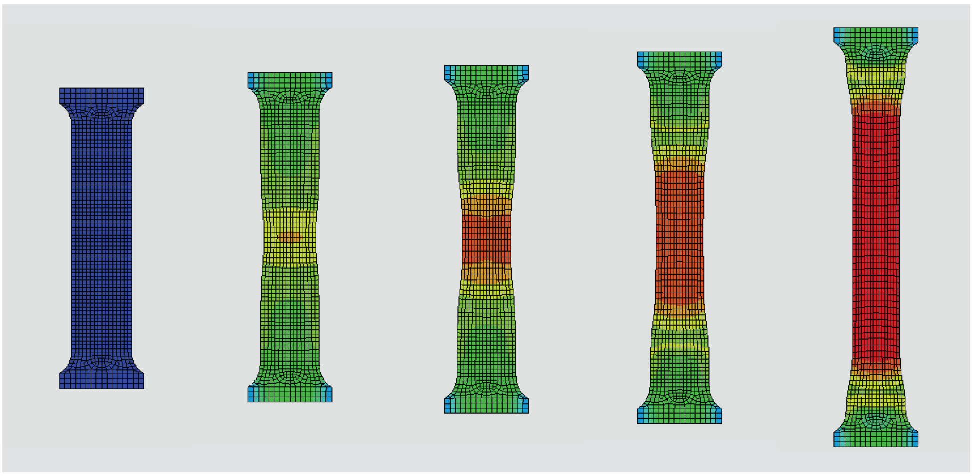


Figure 4.6: A simple tension test of polycarbonate using axisymmetric elements and the VUMAT user subroutine.

4.3 Material Parameters

The idea behind this study is to show that the Arruda-Boyce subroutine can be used to model several types of polymers by changing the material properties. It is also important to understand the meaning of each material parameter and what it does to the material response. One of the most widely used polymers is polycarbonate, and is used as the "benchmark material" in this study. The material parameters for polycarbonate are given in the following table[5]:

Polycarbonate	Values	Description
ρ	$1.12[g/cm^3]$	Material density
E	$2300[MPa]$	Youngs Modulus
ν	0.33	Poissons ratio
$\dot{\gamma}_0$	$2 * 10^{15}$	Pre-exponential factor
h	$500[MPa]$	Softening slope
$\frac{s_{ss}}{s_0}$	0.78	shear stress state ratio
A	$3.31 * 10^{-27}$	Zero stress level activation energy
C_r	$18.0[MPa]$	Rubber modulus
N	2.78	Limeting chain stretch parameter
θ	$273 + 22.5[K]$	Absolute temperature

Notice that the temperature of the material in this study is always $22.5[C]$. Even if it is not explicitly studied, it is still easy to see how the temperature affects the material response. The rubber modulus in chapter 4.3.2, is given by $C_r = n\kappa\theta$, and the zero stress level activation energy in chapter 4.3.3, A, is always divided by the absolute temperature when used: $\frac{A}{\theta}$. This means that, for example, rising the absolute temperature with 10% has the same effects as increasing C_r with 10% and decreasing A with $(100 - \frac{100}{10})\% \approx 9.1\%$.

4.3.1 Elastic Stress Response

The elastic stress response is governed by the Youngs modulus E and the poissons ratio ν. Youngs modulus is normally the slope of the elastic section in the material, but in the Arruda Boyce theory, E has even more effect on the material. In figure 4.7 it is easy to see that increasing E means increasing the slope of the elastic region. Moreover, it also increases both the strain and stress values at the yield point. In

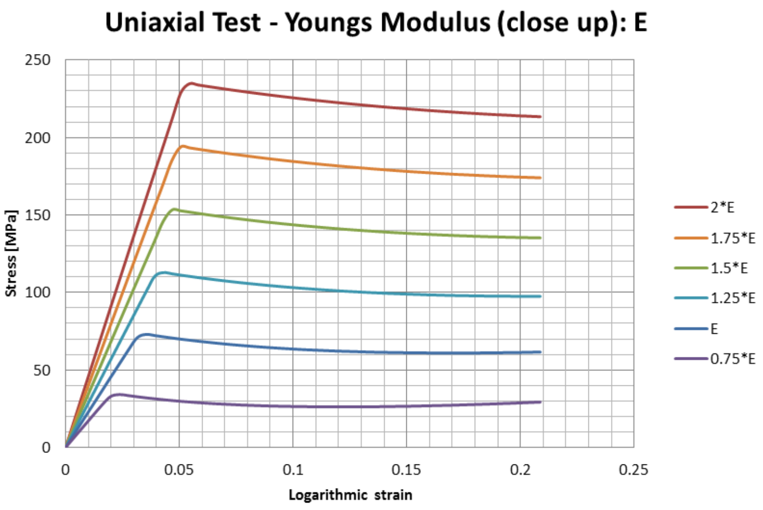


Figure 4.7: Youngs modulus effect on the elastic section and the yield point.

addition, E also increases the slope of the strain hardening section quite drastically. See figure 4.8.

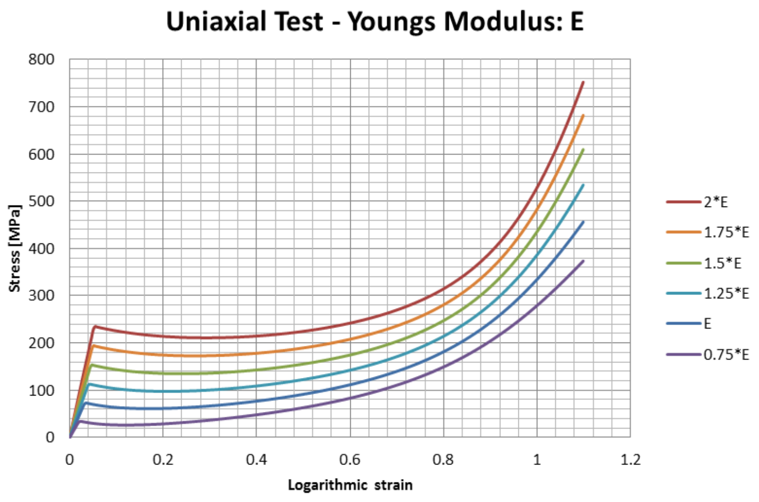


Figure 4.8: How changing the Youngs modulus E, affects the material

Even though the values for ν is around 0.33 for most polymers, it is interesting to see how the value affects the material response. Higher Poissons ratio leads generally to a more drastic change in cross section. From figure 4.9 it is clear that the slope of the elastic section remains constant with different values for Poissons ratio. The yield point occurs earlier for increased Poissons ratio. The reason for a Poisson value

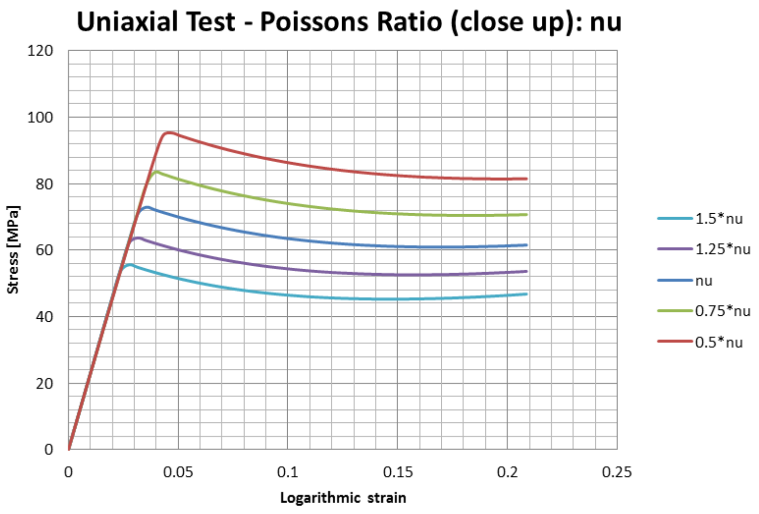


Figure 4.9: Poissons ratio: close up at the elastic section with the yield point

dependent yield point is that the initial strain softening is highly dependent on ν.

$$s_0(E, \nu) = 0.077 \frac{G(E, \nu)}{1 - \nu} \tag{4.2}$$

The strain softening parameter is used in equation 2.38 along with the zero strain level activation energy constant A, which controls the yield. It is clear that changing the strain softening parameter, can give somewhat the same result as changing A. This is further explored in chapter 4.3.3.

4.3.2 Backstress/Langevin Spring

The Langevin spring is characterized by the rubber modulus term: $C_r = n\kappa\theta$, and the limiting chain stretch parameter: N. The stress in the Langevin spring denotes the difference between the stress in the viscoplastic dampening element and the elastic stress: $T^{vp} = T^e - T^b$. Since T^{vp} is the plastic driving stress, more backstress leads to less plastic deformation. The backstress does not affect elastic section or the yield point, in fact, it only becomes active after the material has reached yield. The backstress has no effect on the yield point, as can be observed from figure 4.10.

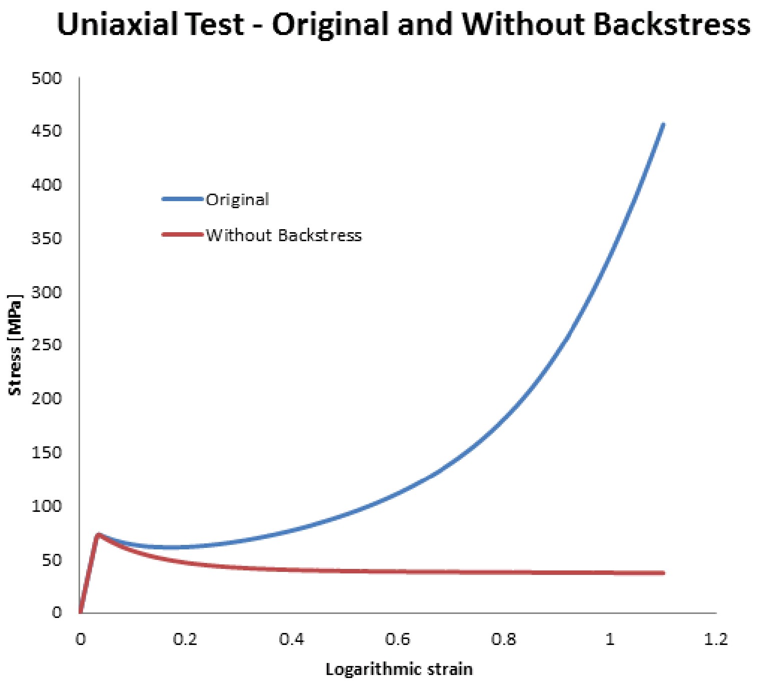


Figure 4.10: The effect of setting $T^b = 0$ in the user material subroutine.

The C_r parameter has somewhat the same meaning for the stress response as Youngs modulus has for the linear elastic spring, and is connected to the number of polymer chains per volume. A higher rubber modulus leads to a stiffer Langevin spring. Figure 4.11 shows what happens if the value of the rubber modulus is changed.

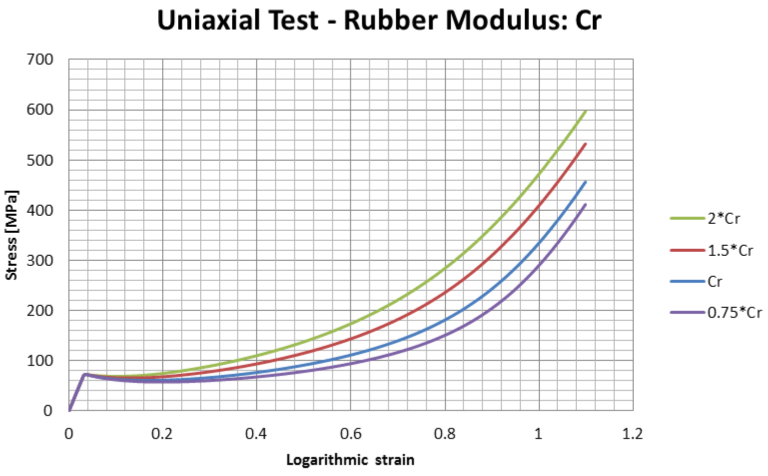


Figure 4.11: How changing the Rubber modulus, C_r, affects the material.

The chain density can, for semi-crystalline polymers, be determined from the degree of cross linking between chains per unit volume. The limited chain stretch parameter $\sqrt{N}$ is the square root of the number of statistical rigid links per chain N. However, for amorphous materials such as polycarbonate, the cross linking does not exist. N is then a measure of the polymer chain lengths. From figure 4.12 it can be observed that shorter polymer chains lead to an earlier and more rapid strain hardening section.

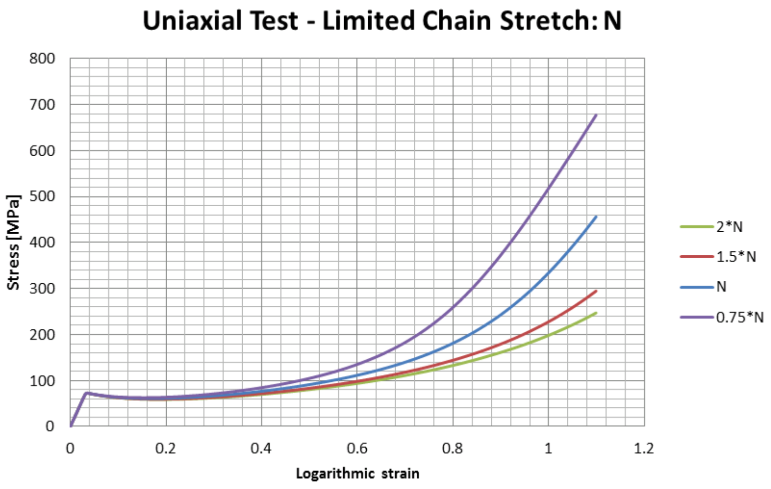


Figure 4.12: How the limited chain parameter N, affects the material

4.3.3 Viscoplastic Element

The viscoplastic element controls both yield and strain softening. It is therefore convenient to devide this section in two subsections.

Viscoplastic Yield

The yield is characterized by two parameters. $\dot{\gamma}_0$, the pre-exponential factor in the flow rule, and the zero stress level activation energy: A. From figure 4.13 it can be observed that changing the value for $\dot{\gamma}_0$ has the same effect as changing the displacement rate or the strain rate. Increasing $\dot{\gamma}_0$ simulates lower displacement rates. The effect of the zero stress level energy, A, that is illustrated in figure 4.14 will, like $\dot{\gamma}_0$, change the yield point. Higher zero stress level activation energy increases the yield point. As the header for this chapter suggests, these material properties only affect the yield point. When the material experiences excessive strain, the effect is not recognizable.

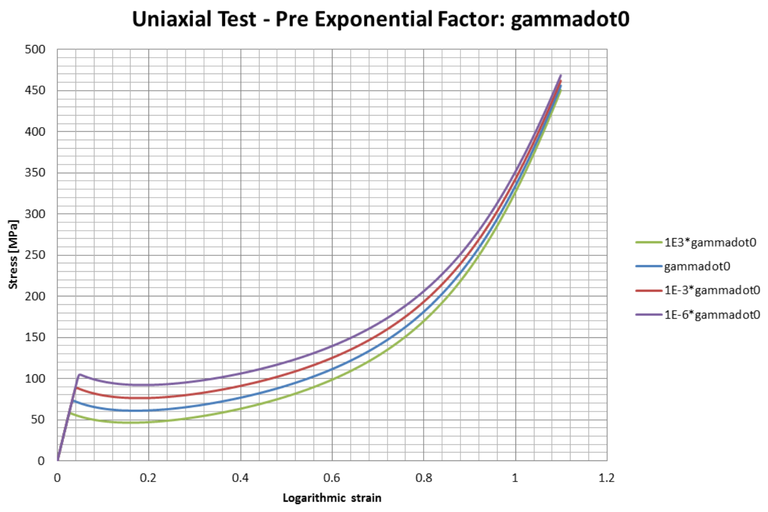


Figure 4.13: How the pre-exponential factor $\dot{\gamma}_0$, affects the material

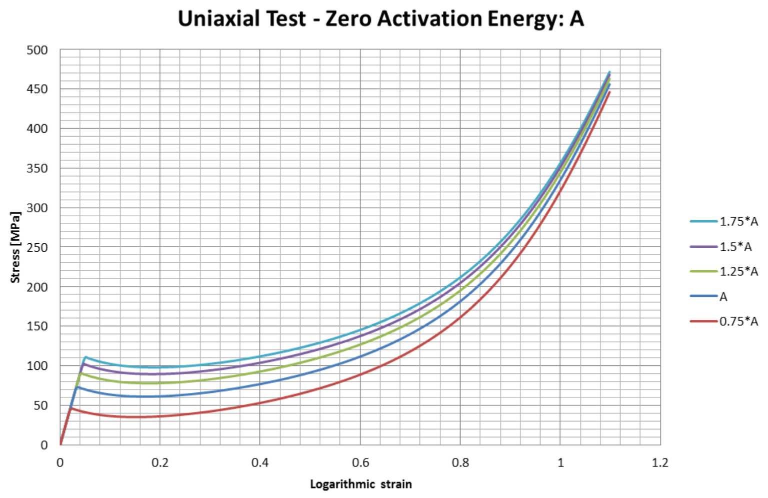


Figure 4.14: How the Zero level activation energy A, affects the material

Strain Softening

Strain softening is controlled by the athermal shear strength s_t. The initial athermal shear strength is given by equation 4.3

$$s_{t=0} = s_0 = 0.077 \frac{G}{1-\nu} \tag{4.3}$$

During the test, s_t goes towards s_{ss} which is generally lower than s_0. This leads to strain softening. The parameters studied in this section are the preferred shear strength fraction $\frac{s_{ss}}{s_0}$ and the softening slope h. Lowering the fraction consequently means lowering s_{ss}, since s_0 is always defined by 4.3. From figure 4.15 it can be observed that a lower fraction means more strain softening.

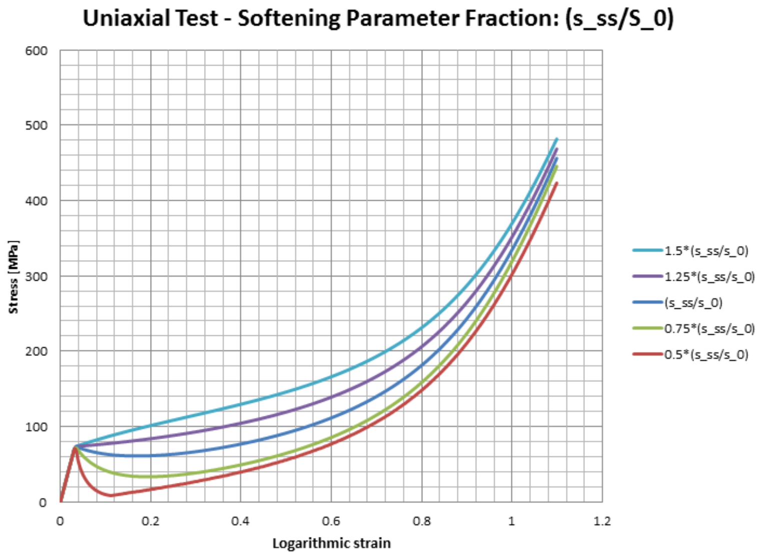


Figure 4.15: Illustration of how the preferred shear stress state ratio, $\frac{s_{ss}}{s_0}$, affects the material

The softening slope h is used in equation 2.39. It is proportional to the rate of the athermal shear strength. A higher softening slope means that the athermal shear strength will reach its preferred state faster. Thus, it leads to more radical strain softening. The effect can be observed in figure 4.16

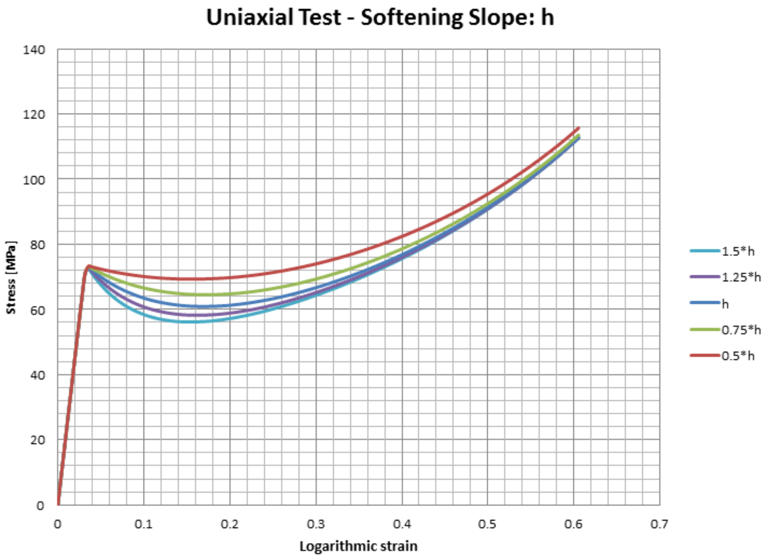


Figure 4.16: Illustration of how the softening slope h affects the material

4.4 Strain Rate

Since the Arruda-Boyce model is a viscoplastic material model, it is important to show how the strain rate affects the result. The same single element used in this chapter is uniaxially tested using displacement control. For simplicity, and because of the limitations of an explicit analysis, the rate of the displacement is constant. This means that the strain rate will vary slightly due to the change in the cross section area. In figure 4.17 the stress-strain curves from the uniaxial tests with displacement rates, ranging from $0.01mm/s$ to $600mm/s$, are illustrated. It can be seen that the strain rate mostly affects the yield point, and the difference between the curves gets smaller when the total strain rises. This is because the stress contribution from the viscoplastic dampening elements gets lower as the strain increases. The most interesting change in the stress strain-curve when analyzing different displacement rates, is the change in the yield point. In figure 4.18 it can be observed that the elastic stiffness is unaffected by the strain rate. However, increasing the displacement rate also increases the elastic zone domain. In figure 4.19, the development of the yield stress is plotted against the displacement rate. It shows a logarithmic relationship between displacement rate and yield point stress.

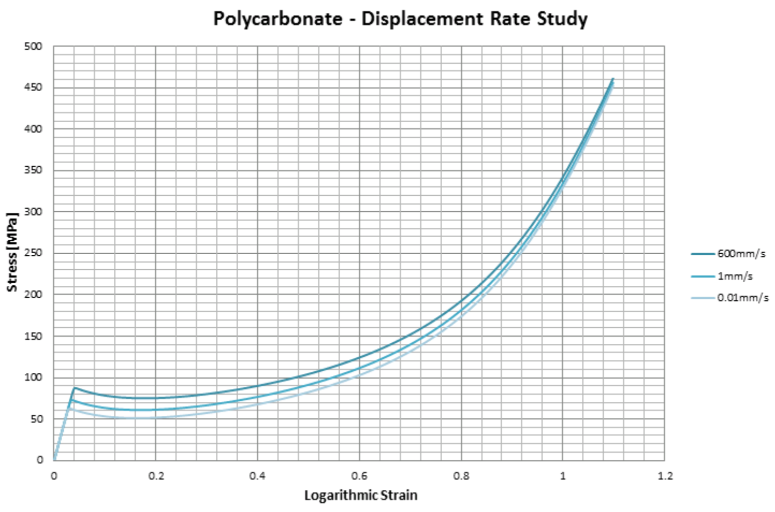


Figure 4.17: Illustration that shows the viscosity effects of the material

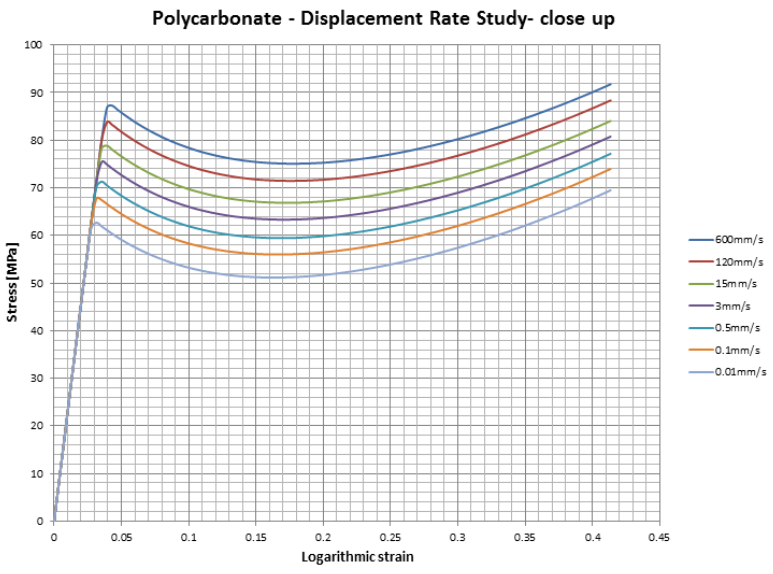


Figure 4.18: Close up at the yield point showing that it depends on the displacement rate.

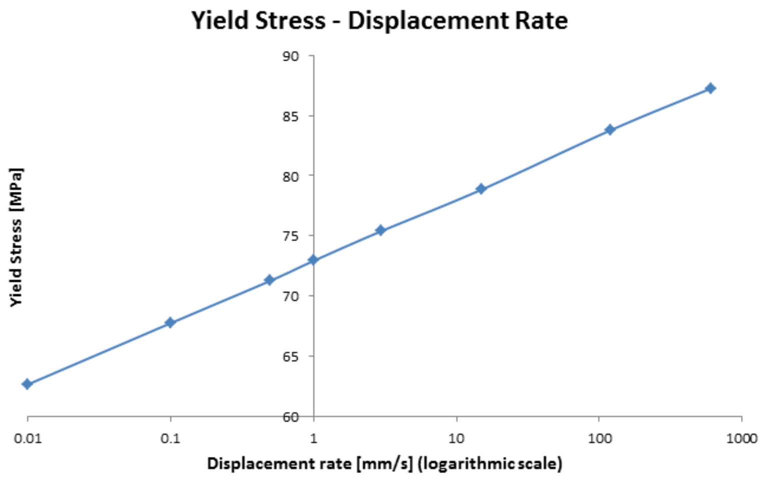


Figure 4.19: Evolution of the yield stress at different displacement rates

The computational time used in an explicit analysis is highly dependent on the strain rate/displacement rate. Analyzing the single element with $600mm/s$ took only a couple of seconds. Analyzing the single element with $0.01mm/s$ however, tok 25 hours. This is because Abaqus does not take the strain rate into consideration when deciding the magnitude of the time increment. The time required to analyze complex problems with slow deformation can easily be in the magnitude of several days.

It is always recommended to let Abaqus decide the time increment used because of stability reasons. The time increments are based on the materials density, and elastic response. Higher density means longer time increments and thus, a shorter analysis time. Changing the density is not recommended, especially for highly dynamic tests. It is also possible to tweak the time increment by changing the elastic stiffness in the first increment of the analysis. A lower stiffness will give a longer time increment. However, this can change the stability of the analysis. In the light of the strain rate study, and the material parameter study, another method for increasing efficiency proves successful.

Recall figure 4.13 where several pre-exponential factors where analyzed. The behavior looks very much like analyzing different displacement rates (figure 4.17). As for different displacement rates, the yield stress was also plotted with the pre-exponential factor $\dot{\gamma}_0$ in figure 4.20. The slopes of the curves in figure 4.20 and 4.19

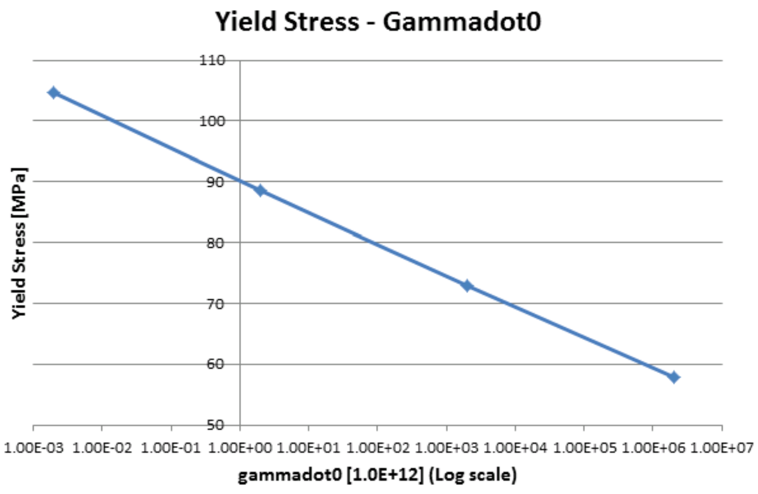


Figure 4.20: Evolution of the yield stress at different values for $\dot{\gamma}_0$

have about the same magnitude ($\frac{slope_1}{slope_2} = -0.988$). In other words: reducing the displacement rate with a factor of 10^3, is roughly equivalent to multiplying the pre exponential factor with 10^3. This proves to be a more efficient way of analyzing problems with slower displacement rates.

4.5 Different Load Cases

4.5.1 Tension and Compression

In this uniaxial test, the specimen was first imposed a displacement of $30m$ with a displacement rate of $1\frac{mm}{s}$. Then, immediately after, it was compressed back to its initial length with $-1\frac{mm}{s}$. The strain history is illustrated in figure 4.21, and the stress-strain curve is shown in figure 4.22.

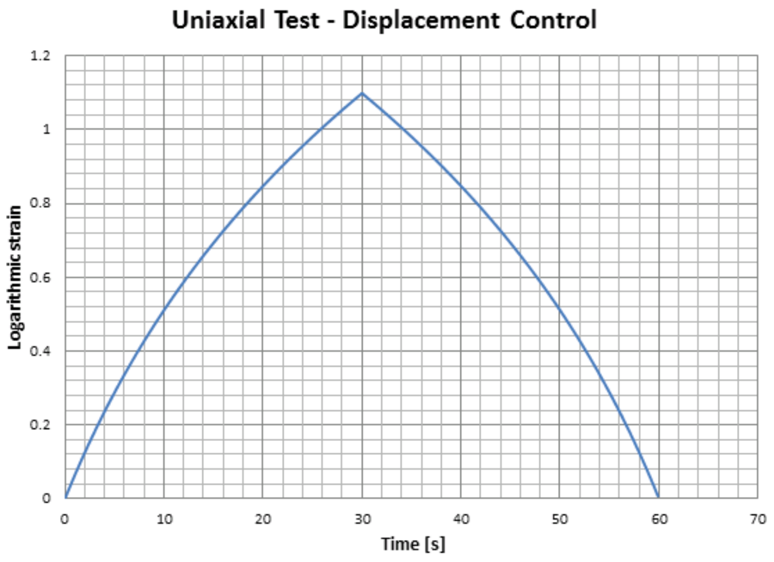


Figure 4.21: Strain history of the specimen.

Notice that the strain is not proportional to time. This is because a constant displacement rate has been chosen in this study, and not a constant strain rate. For small deformations the difference is negligible. At most, the element experienced 115% strain. From figure 4.22, it can be observed that the material experienced elastic deformation at the start of the test. Immediately after the displacement changed direction, it experienced some elastic deformation again before the deformation became plastic. When the element is fully relaxed, it will have a permanent strain of roughly 40%. I.e. if the specimen had been unloaded when it reached 115% strain, the relaxed specimen would then be 40% longer than the original specimen. In order to reach the initial length, the element must be compressed with a pressure of $20MPa$.

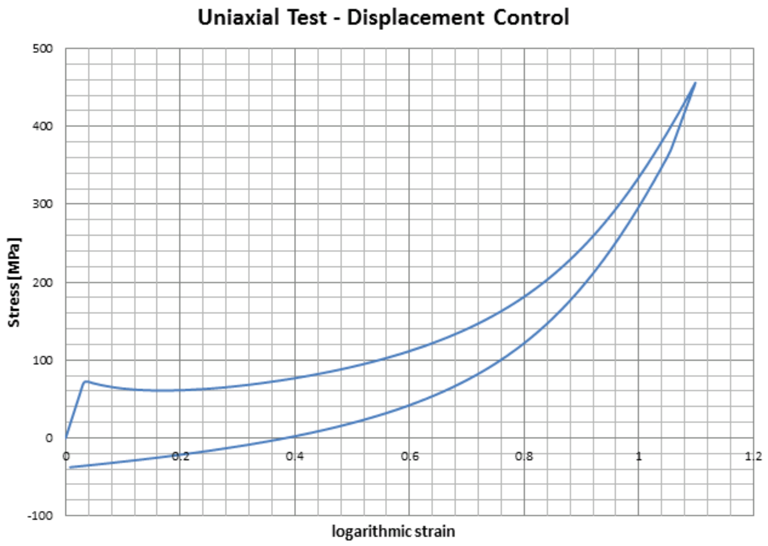


Figure 4.22: Stress and strain history of polycarbonate at $1mm/s$.

4.5.2 Hysteresis

In this uniaxial test, a specimen was imposed a strain of 47% with displacement rate $1mm/s$. Then it was immediately compressed to a strain state of -47% at $-1mm/s$. Afterwards it was imposed with a displacement to get it back to original length. Figure 4.23 shows the strain history of the specimen. As a reference, a purely elastic material would reach its initial state after the strain history given in figure 4.23.

The strain softening is modeled by letting the strain softening parameter start at an initial value s_0, and then, after some deformation, reach s_{ss}. The model is defined in such a way that every form of inelastic deformation will contribute to a change in s towards s_{ss}(see equation 2.39 and 2.38). After the elastic response at the start of the uniaxial test, the material experiences strain softening. At the highest stress state, roughly $85MPa$, the strain softening parameter has reached its preferred state. Because of equation 2.39, it is impossible to change the strain softening parameter when it has reached its preferred state. This is why no strain softening can be observed during the unloading and the compression.

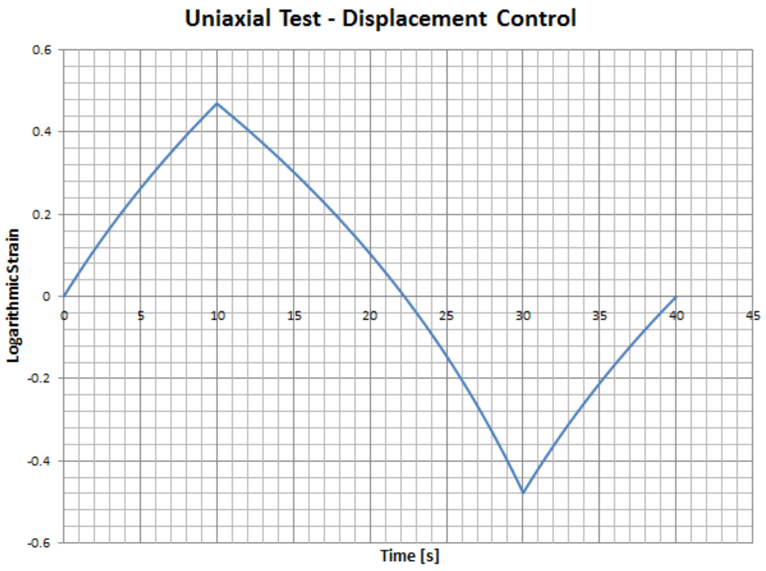


Figure 4.23: Stress-strain history of polycarbonate at $1mm/s$.

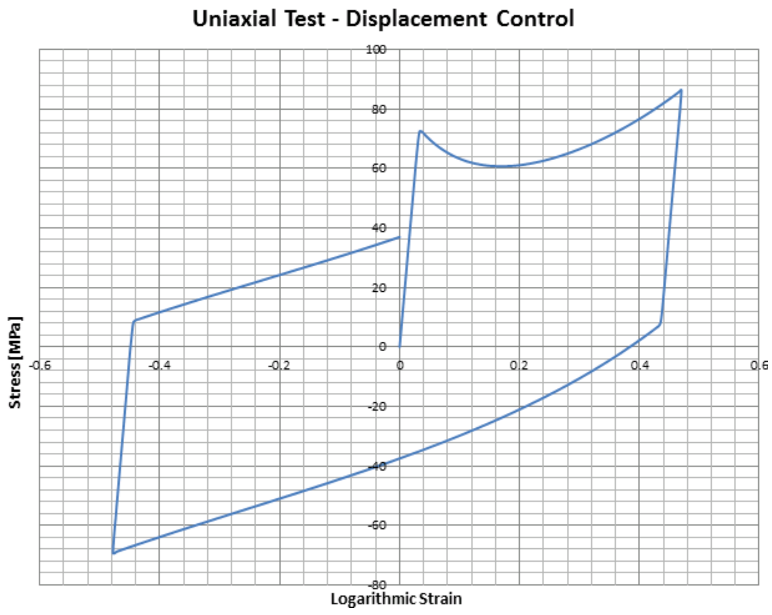


Figure 4.24: Stress and strain history of the specimen.

4.6 Polymethylmethacrylate - PMMA

In order to demonstrate the diversity of the Arruda-Boyce model, it is important to show that it can model other materials than polycarbonate. Polymethylmethacrylate, or PMMA, is a transparent, lightweight and shatter resistant thermoplastic. A more common name is Plexiglas. The material can be modeled in Abaqus using the Arruda-Boyce user material subroutine. It is a much tougher material then polycarbonate with 50% increase in Youngs modulus and very little strain softening. In the next table, the material parameters needed to model Plexiglas are presented.

Polymethylmethacrylate	Values	Description
ρ	$1.18[g/cm^3]$	Material density
E	$3205[MPa]$	Youngs modulus
ν	0.33	Poissons ratio
$\dot{\gamma}_0$	$2.8 * 10^7$	Pre-exponential factor
h	$315[MPa]$	Softening slope
$\frac{s_{ss}}{s_0}$	0.87	Shear stress state ratio
A	$1.39 * 10^{-18}$	Zero stress level activation energy
C_r	$8.0[MPa]$	Rubber modulus
N	2.1	Limiting chain stretch parameter
θ	$273 + 22.5[K]$	Absolute temperature

An uniaxial test was executed on the single element with PMMA material properties. The results are shown in figure 4.25 and 4.26 When comparing this material to polycarbonate, it is expected a stiffer elastic response due to the increase in Youngs modulus. Due to a higher preferred shear stress state ratio and a lower strain slope, there is very little strain softening. The limited chain stretch parameter is smaller, which leads to a more rapid rise of the stress during strain hardening. The results show that the material behaves as expected.

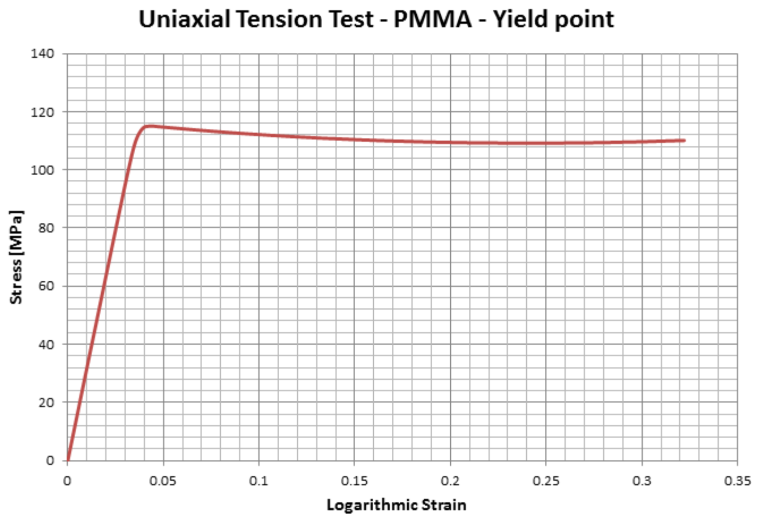


Figure 4.25: Close up of the yield point at the stress-strain curve of a uniaxial test of Plexiglas

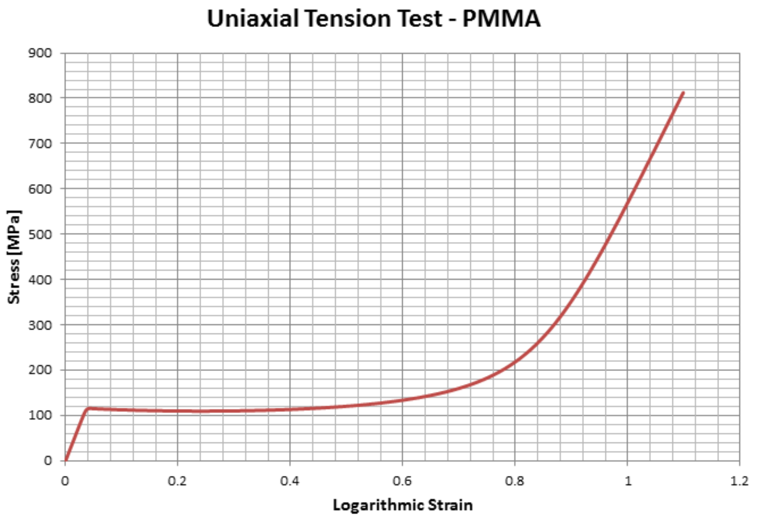


Figure 4.26: Stress-strain curve of Plexiglas at large deformations

Chapter 5

Cell Model and Void Growth

5.1 Cell Model

5.1.1 Introduction

Most polymers are ductile at room temperature, and the fraction of polymers seems to follow the same stages as for ductile metals. The first stage is void nucleation, where an inclusion or a particle debonds from the material matrix and creates a void. The second stage is void growth, where a combination of loading and hydrostatic stress leads to increased void volume fraction. The last stage is void coalescence, where the material between voids experiences excessive strain and necking effects before the material fractures. In this thesis, only void growth will be studied. It is assumed that the void nucleation already took place, and that the material has an initial void volume fraction of 1%.

5.1.2 Cell Model

A material can be approximated as a collection of hexagon prisms. After the nucleation, the particle in the matrix has no more effect, and we consider it as a void.

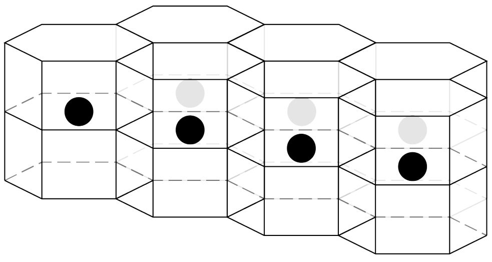


Figure 5.1: The material can be approximated by a series of Hexagon prisms

Assuming that each hexagon prism in the matrix contains a void, the prisms are approximated as cylinders containing voids due to symmetrical benefits. The

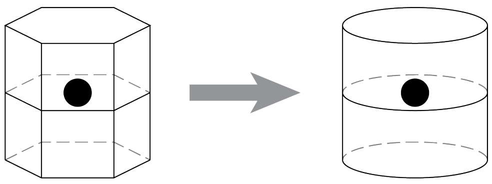


Figure 5.2: Approximating the hexagon prism with a cylinder for symmetric reasons.

cylinders can easily be modeled using the triaxiality function in Abaqus. The final model becomes a rectangle with a cut out quarter circle in the bottom left corner.

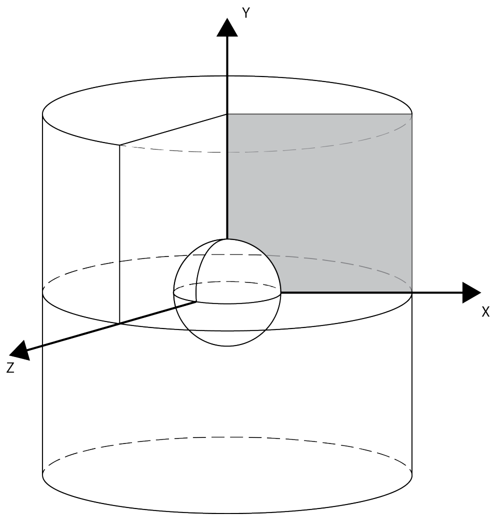


Figure 5.3: Using axisymmetry, the cell model can be greatly simplified

5.1.3 Strain Ratio

Because of the limitations of an explicit analysis, it is not easy to keep a constant stress triaxiality. In 2013, S. A. Reffas, M. Elmeguenni and M. Benguediab did some analysis on void growth in polymers[13]. They used a fairly simple material without strain softening or viscoplastic properties, in addition to power law hardening. The Arruda-Boyce model for polymers is more complex, and thus more challenging to implement in a user material subroutine. Unfortunately, the user subroutine written during this master project is an explicit method, and can only be used in an explicit analysis.

The traditional way of studying void growth, is with a constant stress traixiality. For an implicit analysis with strain softening, this can either be done using the Riks method, or a combination of springs and MPCs(multi point constraints) user subroutines. Since either Riks method[2] or MPC[3] user subroutines are available for an explicit analysis, a different approach was needed. An effort has been made to approximate the same conditions used in the paper from S. A. Reffas, M. Elmeguenni and M. Benguediab [13], using a constant displacement ratio. The stress triaxiality is defined as

$$\beta = \frac{\Sigma_h}{\Sigma_e} = \frac{|\Sigma_r - \Sigma_y|}{\frac{1}{3}(\Sigma_r - 2\Sigma_y)} = \frac{1}{3}\frac{1+2\alpha}{|1-\alpha|} \tag{5.1}$$

Where Σ_h is the hydrostatic stress, Σ_e is the equivalent mises stress and $\alpha = \frac{\Sigma_r}{\Sigma_y}$ is the stress ratio.

The displacement ratio used in this thesis is given by equation 5.2

$$m = \frac{u_y}{u_r} \tag{5.2}$$

The different strains are explained in figure 5.4, where the side of the element is constrained to remain vertical, and the top is constrained to remain horizontal.

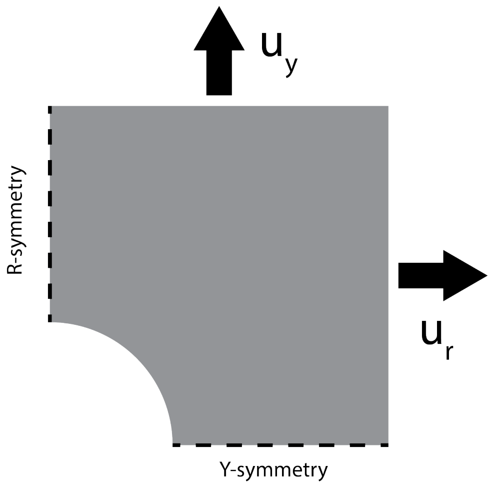


Figure 5.4: Displacement on the cell model

In order to show how the different displacement ratios affect the material, 3 different ratios have been analyzed using the initial void volume fraction: $f_0 = 1\%$. The results are compared with an uniaxial test where the side of the element can move freely. The cell has the dimensions given in figure 5.5, where $R = H = 15mm$ and $r_y = r_x = 1.7438668mm$ which leads to the initial void volume fraction being 1%.

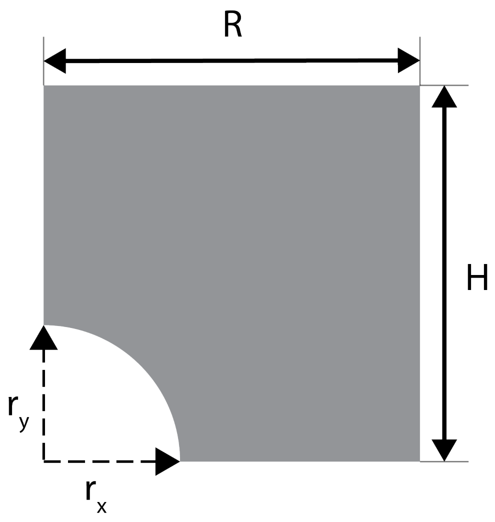


Figure 5.5: Dimensions of the cell model

441 elements, of type CAX4R, were used in a structured mesh, with bias towards the void. The undeformed cell model is illustrated in figure 5.6. The three different strain ratios are $m = -15\%$, $m = 0$ and $m = 10\%$.

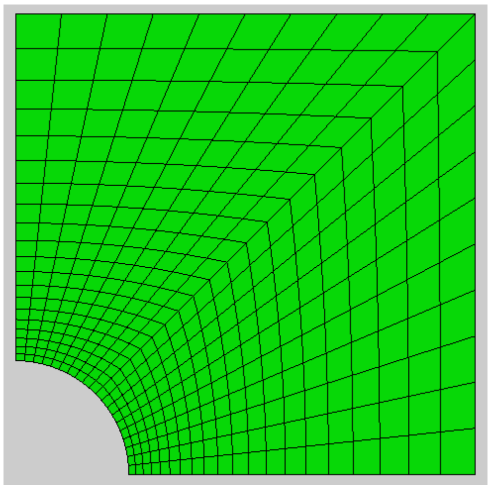


Figure 5.6: Undeformed cell model

Unfortunately, the user material subroutine has a tendency to crash even at moderate global displacements when analyzing a void problem, especially when there are forces in both r direction and y direction. Only the uniaxial analysis could withstand up to 50% strain.

5.1.4 Void Analysis Results

The deformation of the cell model with no imposed strain on the side is illustrated in figure 5.7. For the imposed strain ratios, only results for ϵ_y up to approximately 10%

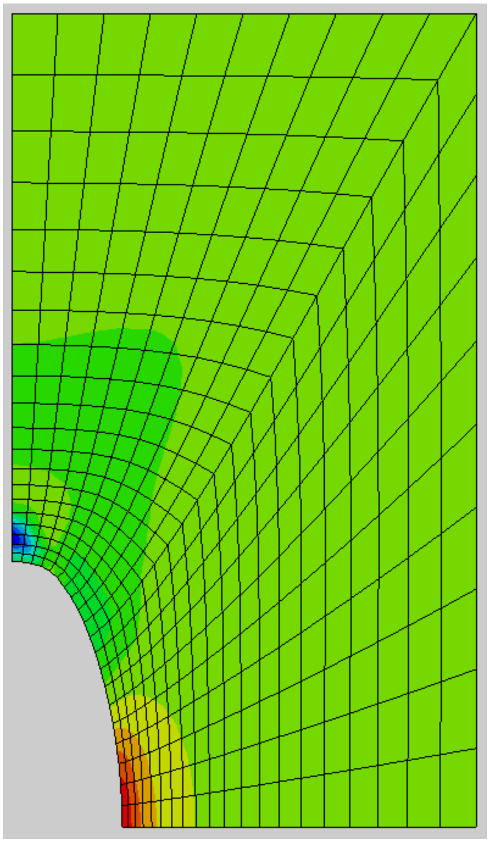


Figure 5.7: Deformed cell model with no imposed strain ratio. The force traction of the edge of the cylinder is zero. The color plot shows the most stressed areas using Mises stress.

are available. Since polymers can withstand much higher strains than metals, this means that no coalescence effects could be observed from the analysis. The deformed shapes at maximum strain before the crash are shown in figure 5.8, 5.9 and 5.10 The color gradient in the figures illustrates the distribution of the Mises stress in the cell model, where red is high and blue is low. The void volume fractions were calculated using Matlab by finding the current area in the quarter circle and calculating the

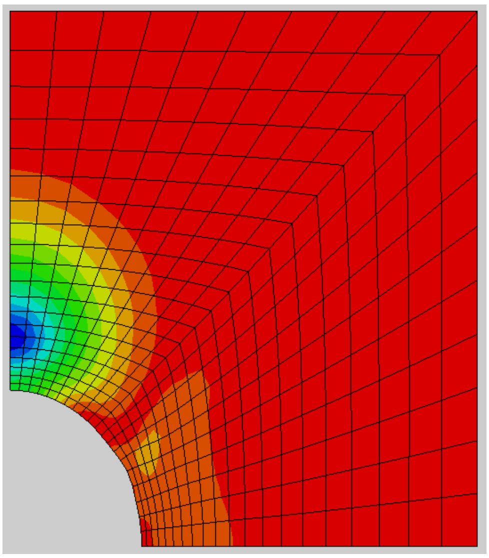


Figure 5.8: Deformed cell model with strain ratio: $m = -15\%$

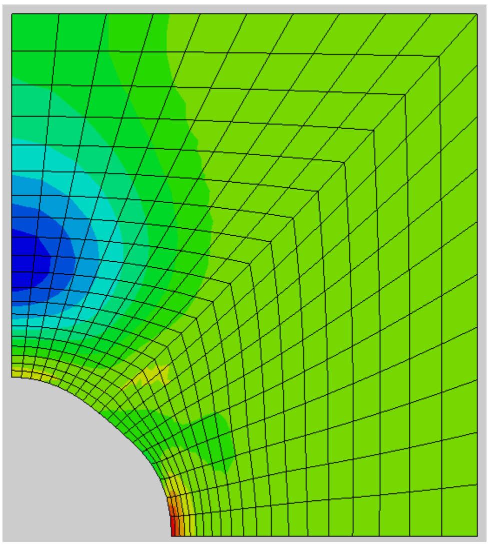


Figure 5.9: Deformed cell model with strain ratio: $m = 0\%$

current void volume. Then, the fraction was found by dividing the current void volume by the current total cylinder volume. The area in the quarter circle was calculated using the sum of small triangles and integrating using the cylindrical shell method(see figure 5.11). The development of the void volume fraction is given in figure 5.12. It is important to emphasize that the total cell growth is taken into

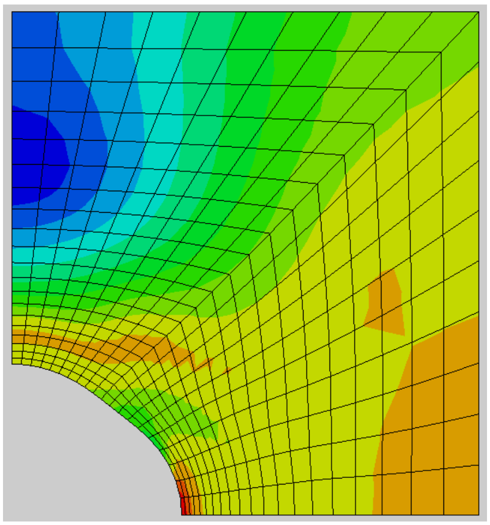


Figure 5.10: Deformed cell model with strain ratio: $m = 10\%$

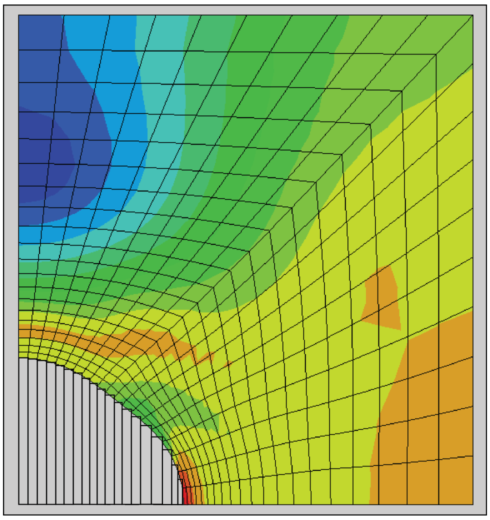


Figure 5.11: Calculating the current void volume fraction using small rectangles and the cylindrical shell method.

consideration when calculating void volume fraction. The "free side" analysis shows that the void growth is very small when there is no stress triaxiality or no imposed strain in the radial direction. In the three other analysis, the cell model is either constrained in the radial direction, or imposed with a positive or negative strain in the r-direction. All of these scenarios produce a positive hydrostatic stress(see figure

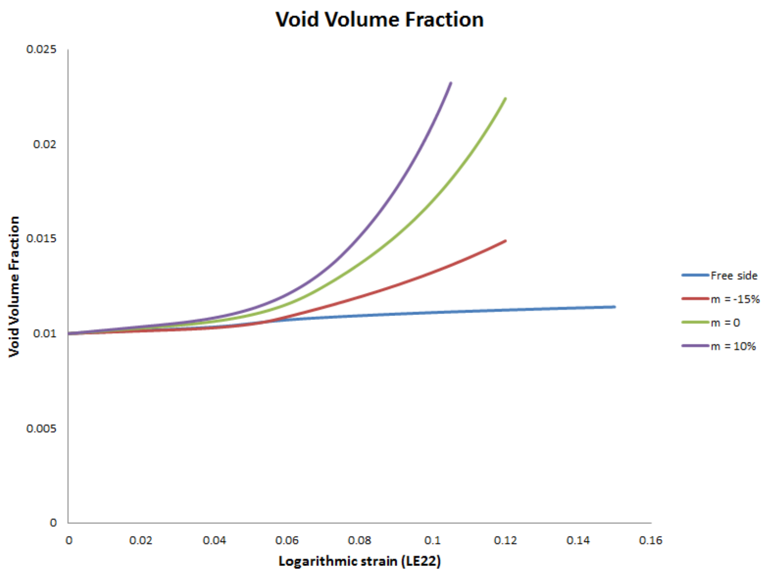


Figure 5.12: It is clear that higher values of m means more rapid rise in the void volume fraction f

5.13). Each of these analysis showed some kind of void growth.

It is important to show how this analysis differs from traditional cell model studies with constant stress triaxiality [13], [1], which is equivalent to a constant stress ratio(see equation 5.1). In figure 5.13 a stress-stress curve is represented by showing how the ratio between the radial stress and the y-stress develops. For a constant stress rate, these curves would have been linear. The curves seem to be linear for smaller strains, but when the deformation gets bigger and out of the elastic domain, the stress ratio $\alpha = \frac{\Sigma_y}{\Sigma_r}$ grows rapidly. Since the material model breaks down at high deformations around the void, no coalescence effect could be seen. However, these results suggest that the rate of the void growth depends on the strain ratio.

In order to see how the Arruda-Boyce material definition affects the void growth, an elastic-plastic material(pure Mises), was tested with the same strain conditions as the polycarbonate material. The Mises material had the same density, elastic response and Poissons ratio as polycarbonate. The material yield stress was constant at $72[MPa]$, which gives the material the same yield point as polycarbonate at $1mm/s$ displacement rate. First, it is interesting to show how the material affects the void growth with no hydrostatic stress. In figure 5.14 it can be concluded that overall, when there is no hydrostatic stress, the Arruda-Boyce material shows less void growth than a Mises material. The exception is between strain values of 5% and

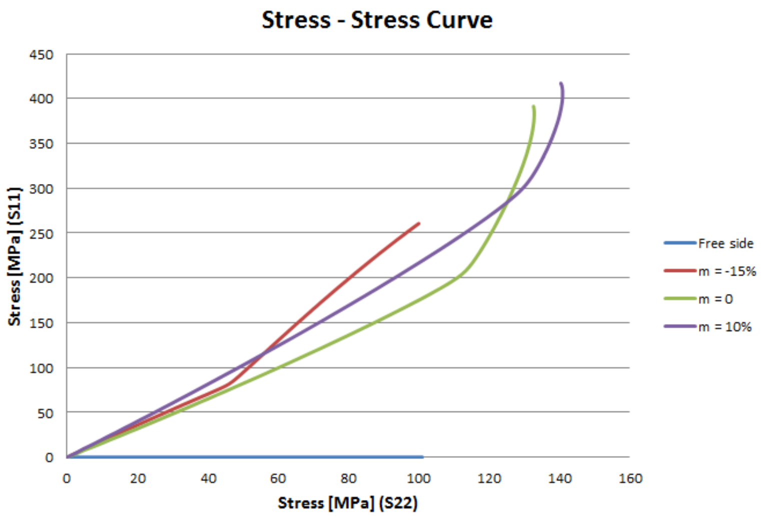


Figure 5.13: The stress in the y-direction (S22) is plotted against the stress in the r-direction (S11)

14%. This can be explained by the strain softening in the Arruda-Boyce material. The difference in void growth between this material and polycarbonate modeled by the Arruda-Boyce model is illustrated in figure 5.15. The Mises material shows good signs of coalescence even at moderate strains. Even at 10% strain, the void had grown considerably in the horizontal direction. In figure 5.16, it can be observed that the stress concentrates at the same hight as the void and that the void has grown considerably in the radial direction. This is good visual signs of void coalescence. Since the user material subroutine crashed at low strains, such signs could not be observed for polycarbonate.

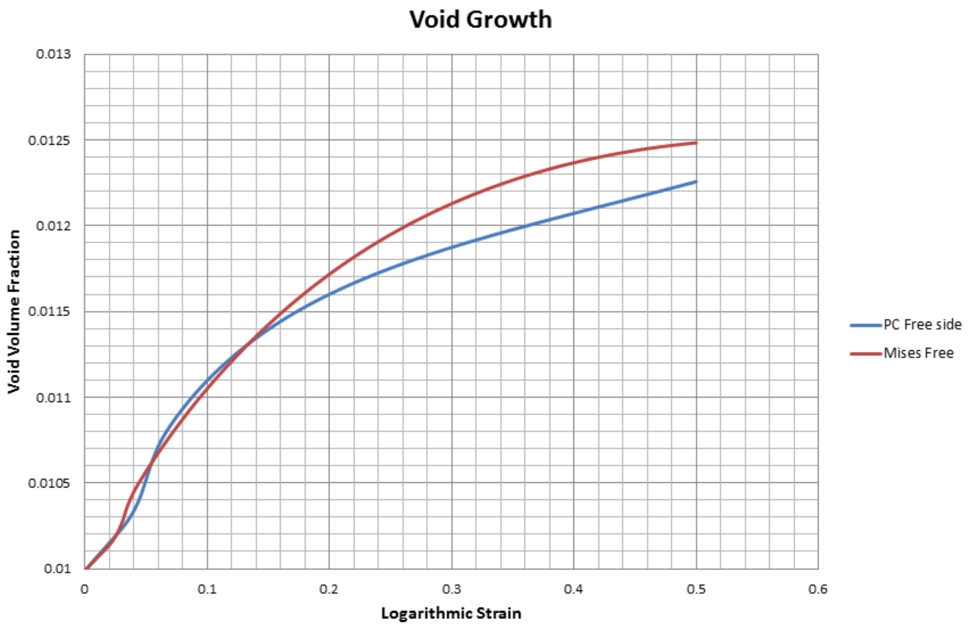


Figure 5.14: A purely elastic plastic material has slightly more void growth than polycarbonate in an uniaxial test

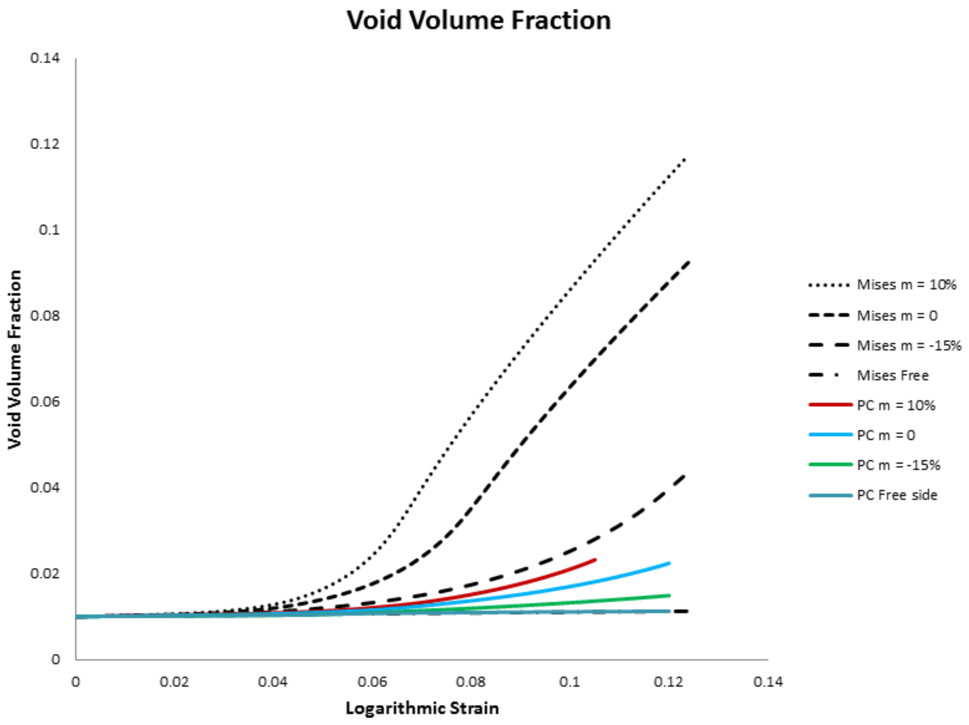


Figure 5.15: A purely elastic plastic material has significantly higher void growth than polycarbonate

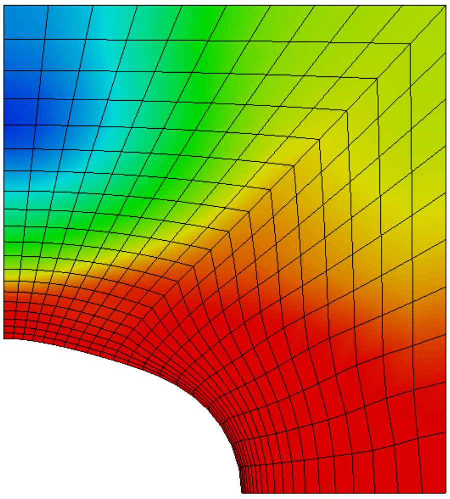


Figure 5.16: Stress distribution and void shape of the Mises material at 20% strain and $m = 10\%$

Chapter 6

Conclusion and Suggestions for Further Work

The main goal of this master thesis was to implement the Arruda-Boyce model for polymers as a user material subroutine for Abaqus. After some literature study regarding explicit and implicit user material subroutines, the choice landed on VUMAT, which is an explicit approach. There was no time left to advance it further into an implicit material model(UMAT), but the methods were studied and the main theory needed is given in appendix A. The VUMAT was developed and tested with several material parameters and loading conditions using a single element. It was also illustrated that the user material did work for more complex geometries with several elements, although somewhat limited. The studies showed that it is possible to simulate void growth with the user material, and that the void growth of polymers depends on the hydrostatic stress or strain.

In the development of the explicit user material, some problems regarding the calculation of the Hencky strain were encountered. Normally, when implementing the Arruda-Boyce material model, a Padé approximation is used in the calculation of the Hencky strain. This second order approximation proved to fail at strains larger than 100%. Another method proposed in this thesis was using the calculation of eigenvalues and eigenvectors, which proved not to affect the simulation time. However, this method also failed at very large deformations, and could be the reason for the early failure in the void growth analysis.

Even though the explicit user subroutine was tested with a single axisymmetric element with several load cases at extreme deformations, Abaqus had difficulties using the model at more complex geometries with multiple elements. A direct extension of this work should be to make the user subroutine work on more complex problem like for example a void growth problem with large deformations.

It should be fairly easy to extend the user material to account for 3D-problems. This could be done simply by changing how the subroutine receives variables from Abaqus and how it updates the values in the end as all calculations made in the subroutine assume a 3D behavior of the material.

Because of the complexity of the Arruda-Boyce model, an analytical solution to the consistent tangent modulus matrix is not easily found. If the explicit user material is extended to include large deformations with more complex mesh conditions, it should be theoretically possible to make an implicit user material subroutine(UMAT). This could be performed using the perturbation method for calculating the consistent tangent modulus matrix.

Furthermore, when a fully functional material model is made, either explicit or implicit, it would be interesting to see if an Arruda-Boyce material experiences further void growth and coalescence in the same manner as metals.

References

[1] E. Van Der Giessen A. C. Steenbrink and P. D. WU. Void growth in glassy polymers.

[2] Inc ABAQUS. Abaqus theory manual, sec 2.3.2 modified riks algorithm.

[3] Inc ABAQUS. Abaqus user subroutine reference manual.

[4] T.L. Anderson. *Fracture Mechanics: Fundamentals and Applications, Third Edition*. Taylor & Francis, 2005.

[5] Ellen M. Arruda and Mary C. Boyce. Evolution of plastic anisotropy in amorphous polymers during finite straining.

[6] Quentin Fichot. Characterization, modelling, and consequences of the development during plastic flow of large anisotropy in the wave-speeds.

[7] A.L. Gurson. Plastic flow and fracture behaviour of ductile materials incorporating void nucleation, growth and coalescence.

[8] P. M. Frontini Ignacio Tomas, Adrian P. Cisilino. A simple numerical implementation of the arruda- boyce viscoplatic model to reproduce the mechanical behavior of uhmwpe.

[9] P. M. Frontini Ignacio Tomass, Adrian P. Cisilino. Acccurate, efficient and robust explicit and implicit integration schemes for the arruda- boyce viscoplatic model.

[10] Willam Kaspar J. Constitutive models for engineering materials.

[11] Joachim Kopp. Efficient numerical diagonalization of hermitian 3×3 matrices.

[12] Joachim Kopp. Numerical diagonalization of 3x3 matrices: http://www.mpi-hd.mpg.de/personalhomes/globes/3x3/, June 2014.

[13] Mohamed Elmeguenni Sid Ahmed Reffas and Mohamed Benguediab. Analysis of void growth and coalescencem in porous polymer materials.

[14] H. Xiao and L. S. Chen. Hencky's elasticity model and linear stress-strain relations in isotropic finite hyperelasticity.

[15] C. Thaulow Z.L. Zhang and J. Ødegård. A complete gurson model approach for ductile fracture.

Appendix A

Perturbation Method

In order to make an implicit user subroutine, the consistent tangent modulus has to be obtained. Obtaining an exact tangent stiffness matrix for the Arruda-Boyce model is a complicated task. The strategy described in this chapter is a numerical calculation of an approximated tangent stiffness matrix [8]. For a 2D analysis, such as plane strain or axisymmetric, Abaqus requires a $4x4$ Jacobian matrix. For a 3D analysis however, it requires a $6x6$ Jacobian matrix. For an isotropic material, each individual element in the Jacobian matrix is given by equation A.1.

$$\frac{d\Delta\boldsymbol{\sigma}}{d\Delta\boldsymbol{\epsilon}} = \frac{d\Delta\sigma_i}{d\Delta\epsilon_j} = (\frac{d\Delta\sigma_i}{d\Delta\epsilon_j})^T \tag{A.1}$$

The definition of the Jacobi matrix given in equation A.1 defines the change in σ_i at the end of the time increment caused by an infinitesimal perturbation of ϵ_j. For a "hypoelastic material", change of stress can be calculated as a function of the change in strain: $\Delta\boldsymbol{\sigma} = \Delta\boldsymbol{\sigma}(\Delta\boldsymbol{\epsilon})$. The perturbation method is an analytical approximation based on the elementary definition of the differential.

$$\frac{d\Delta\boldsymbol{\sigma}}{d\Delta\boldsymbol{\epsilon}} \cong \frac{\delta\Delta\boldsymbol{\sigma}}{\delta\Delta\boldsymbol{\epsilon}} = \frac{\Delta\boldsymbol{\sigma}(\Delta\boldsymbol{\epsilon} + \delta\boldsymbol{\epsilon}) - \Delta\boldsymbol{\sigma}(\Delta\boldsymbol{\epsilon})}{\delta} \tag{A.2}$$

Where δ is the perturbation, $\Delta\boldsymbol{\epsilon}$ is the strain increment from time t to $t + dt$. $\delta\boldsymbol{\epsilon}$ is a tensor near zero with two small perturbations of size δ in the components $\delta\boldsymbol{\epsilon_{ij}}$ and $\delta\boldsymbol{\epsilon_{ji}}$. The remaining entries of $\delta\boldsymbol{\epsilon}$ are equal to zero. Since the Arruda-Boyce theory is not an hypoelastic constitutive law, the stress increment function $\Delta\boldsymbol{\sigma} = \Delta\boldsymbol{\sigma}(\Delta\boldsymbol{\epsilon})$ is not known. Instead, the total stress function is known: $\boldsymbol{\sigma}_{t+dt} = \boldsymbol{\sigma}({F^e}_t, {F^i}_t, F_{t+dt})$. The approach that can be used for the Arruda-Boyce model is a perturbation of the total deformation gradient.

A perturbation in the deformation gradient can be expressed by first acquiring the total strain tensor at the end of the increment. This is done by calculating the Hencky strain, ϵ_{t+dt}, from the total deformation gradient F_{t+dt}. This can be done using equation A.3

$$\boldsymbol{\epsilon}_{t+dt} = log[V_{t+dt}] \tag{A.3}$$

Where V is the left stretch tensor acquired from the deformation gradient F. When the total strain is found, add the perturbation and make a new total perturbated strain tensor

$$\tilde{\boldsymbol{\epsilon}}_{t+dt} = \boldsymbol{\epsilon}_{t+dt} + \delta\boldsymbol{\epsilon} \tag{A.4}$$

Reassemble the perturbated strain tensor into a perturbated deformation gradient using equation 2.2.3

$$\tilde{F}_{t+dt} = exp[\tilde{\boldsymbol{\epsilon}}_{t+dt}]R \tag{A.5}$$

Where R is the rotation tensor found in the calculation of the left stretch tensor. The approach for calculating the Jacobian matrix is then shown in equation A.6.

$$\frac{\delta\Delta\boldsymbol{\sigma}}{\delta\Delta\boldsymbol{\epsilon}} = \frac{\tilde{\boldsymbol{\sigma}}({F^e}_t, {F^i}_t, \tilde{F}_{t+dt}) - \boldsymbol{\sigma}({F^e}_t, {F^i}_t, F_{t+dt})}{\delta} \tag{A.6}$$

For a plane strain analysis or an axisymmetric analysis, this process has to be repeated 4 times.

- Pertubation in ϵ_{11}.
- Pertubation in ϵ_{22}.
- Pertubation in ϵ_{33}.
- Pertubation in ϵ_{12} and ϵ_{21}.

This calculation comes on top of the already calculated stress $\boldsymbol{\sigma}({F^e}_t, {F^i}_t, F_{t+dt})$, and will increase the computational power required by a factor of 5. For a 3D analysis, the process has to run 6 times.

Appendix B
The User Material Subroutine - VUMAT

Aboyce.for

```
      subroutine vumat(
C Read only (unmodifiable)variables -
     1  nblock, ndir, nshr, nstatev, nfieldv, nprops, lanneal,
     2  stepTime, totalTime, dt, cmname, coordMp, charLength,
     3  props, density, strainInc, relSpinInc,
     4  tempOld, stretchOld, defgradOld, fieldOld,
     5  stressOld, stateOld, enerInternOld, enerInelasOld,
     6  tempNew, stretchNew, defgradNew, fieldNew,
C Write only (modifiable) variables -
     7  stressNew, stateNew, enerInternNew, enerInelasNew )
C
      include 'VABA_PARAM.INC'
C
      dimension props(nprops), density(nblock), coordMp(nblock,*),
     1  charLength(nblock), strainInc(nblock,ndir+nshr),
     2  relSpinInc(nblock,nshr), tempOld(nblock),
     3  stretchOld(nblock,ndir+nshr),
     4  defgradOld(nblock,ndir+nshr+nshr),
     5  fieldOld(nblock,nfieldv), stressOld(nblock,ndir+nshr),
     6  stateOld(nblock,nstatev), enerInternOld(nblock),
     7  enerInelasOld(nblock), tempNew(nblock),
     8  stretchNew(nblock,ndir+nshr),
     8  defgradNew(nblock,ndir+nshr+nshr),
     9  fieldNew(nblock,nfieldv),
     1  stressNew(nblock,ndir+nshr), stateNew(nblock,nstatev),
     2  enerInternNew(nblock), enerInelasNew(nblock)
      character*80 cmname

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C     Some explanation
C     Variables
C           Deformation gradient          [F]
C           Stress tensor                 [T]
C           strain softening parameter    [s]
C           Stretch Tensor                [U]
C           Rotation Tensor               [R]
```

```
C       Type of variables (subscript):
C             D       means derivative with respect to time
C             e       means elastic
C             i       means inelastic
C             n       means time = t        (start of increment)
C             nn      means time = t+dt      (end of increment)
C             _ave    means time = t+dt/2   (middle of increment)
C       Examples:
C         Fin:        Inelastic deformation gradient at start of increment
C         Tnn:        Stress at end of increment
C         s_ave:      Strain softening parameter at middle of increment
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

C       DEFINING VARIABLES
c       Arruda Boyce Theory:
c       Variables taken from abaqus and statevariables:
        Double precision,Dimension(3,3)  :: Fn,Fin,U,Fnn
        Double precision                 :: sn
c       Variables given to abaqus and statevariables:
        Double precision,Dimension(3,3)  :: Finn,Tnn
        Double precision                 :: snn
c       Variables used in the calculation
        Double Precision,dimension(3,3)  :: Fnn2,Fen2,Fin2,Finn2,Fenn2,Fn2
        Double precision,Dimension(3,3)  :: R,Fen,Fenn,F_ave,Fe_ave
        Double precision,Dimension(3,3)  :: Fi_ave,DFi_ave,DFin,DFin2
        Double precision                 :: s_ave,Ds_ave
        Double Precision                 :: Dsn,sn2,snn2,Dsn2
        integer                          :: i,j
c       The material constants
        double precision  ::  E,nu,gammadot0,A,h,sfrac,Cr,N,G
        double precision  ::  boltz,abstemp,sss,flowconst,s0,Lambda
c       lumping material constants together in a vector:
        Double precision,Dimension(8) :: mprop

c       Iteration:
c       variables defined for the iteration process:
        Double precision                 ::  kk,b,c
        integer                          ::  MaxIter
c       variables used in the iteration process
        Double Precision,dimension(3,3)  :: FFE,Tb,Tvp
        Double Precision                 :: eps,temp1,temp2,MinItStep,diff
        Double precision                 :: dt_new,dt_sugg,dt_temp
        integer                          :: k,iter,k2,iter2

C************************************************************************
C********* Defining material and iteration parameter *********************
C************************************************************************

C       Defining all material material properties:
        E             = 2300                         !    MPa
        nu            = 0.33                         !
        gammadot0     = 2.0e15                       !
```

```
      A             = 3.31e-18                         !   mm^3
      h             = 500                              !   MPa
      sfrac         = 0.78                             !
      Cr            = 18.0                             !   MPa
      N             = 2.78                             !

      boltz         = 1.3806503e-20                    !
      abstemp       = 273+22.5                         !   K

      G             = E/(2*(1+nu))                     !   MPa
      s0            = 0.11*G                           !   MPa
      sss           = sfrac * s0                       !   MPa
      flowconst     = A/(abstemp*boltz)                !  (ms^2)kg
      Lambda        = (E*nu)/((1+nu)*(1-2*nu))         !   MPa

C     Lumping relevant material properties in a vector mprop
      mprop(1)          =   G
      mprop(2)          =   Lambda
      mprop(3)          =   Cr
      mprop(4)          =   N
      mprop(5)          =   gammadot0
      mprop(6)          =   h
      mprop(7)          =   sss
      mprop(8)          =   flowconst

C     Iteration
      kk        = 0.1     !smaller kk -> more iterations -> more accurate
      b         = 0.95    !saftey factor
      c         = 0.5     !safety factor
      MaxIter   = 10000   !Max iterations
      MinItStep = 1e-10   !avoid deviding by zero
      diff      = 1e-10   !size of increment

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

c     Abaqus vumat: do for all elements
      do 100 km = 1,nblock

C***********************************************************************
C********************** Initializing variables *************************
C***********************************************************************

      sn        = 0
      snn       = 0
      s_ave     = 0
      Ds_ave    = 0
      Dsn       = 0
      sn2       = 0
      snn2      = 0
      Dsn2      = 0
      eps       = 0
      temp1     = 0
      temp2     = 0
      dt_new    = 0
      dt_sugg   = 0
      dt_temp   = 0

      do i = 1,3
        do j = 1,3
          Fn(i,j)        = 0
          F_ave(i,j)     = 0
          Fnn(i,j)       = 0
          U(i,j)         = 0
          Tnn(i,j)       = 0
          Fin(i,j)       = 0
          Fi_ave(i,j)    = 0
          Finn(i,j)      = 0
          Fen(i,j)       = 0
          Fe_ave(i,j)    = 0
          Fenn(i,j)      = 0
          DFin(i,j)      = 0
          DFi_ave(i,j)   = 0
          FFE            = 0
          Fn2(i,j)       = 0
          Fnn2(i,j)      = 0
          Fin2(i,j)      = 0
          Finn2(i,j)     = 0
          Fen2(i,j)      = 0
          Fenn2(i,j)     = 0
          DFin2(i,j)     = 0
        end do
      end do

C**********************************************************************
C*************** Getting variables from abaqus ***************************
C**********************************************************************

c     get Fn from defgradold
      Fn(1,1) = defgradold(km,1)
      Fn(2,2) = defgradold(km,2)
      Fn(3,3) = defgradold(km,3)
      Fn(1,2) = defgradold(km,4)
      Fn(2,1) = defgradold(km,5)

c     get Fnn from defgradnew
      Fnn(1,1) = defgradnew(km,1)
      Fnn(2,2) = defgradnew(km,2)
      Fnn(3,3) = defgradnew(km,3)
      Fnn(1,2) = defgradnew(km,4)
      Fnn(2,1) = defgradnew(km,5)

c     get the stretch tensor U from stretchnew
c     Not needed until after the iteration.
      U(1,1) = stretchnew(km,1)
      U(2,2) = stretchnew(km,2)
      U(3,3) = stretchnew(km,3)
      U(1,2) = stretchnew(km,4)
      U(2,1) = U(1,2)

c     get Fin from statev, Statev = Fin - I
      do i = 1,3
        Fin(i,i) = 1.0 + StateOld(km,i)
      end do
      Fin(1,2) = StateOld(km,4)
      Fin(2,1) = StateOld(km,5)

c     Get Strain softening parameter from statev: Statev = snn - 1
      sn = s0 + StateOld(km,6)

c     calculate the inelastic deformation gradient at start of increment
      call MatProdInv(Fn,Fin,Fen)

c     Get the rotation tensor R
      call MatProdInv(Fnn,U,R)

C***********************************************************************
C************** Main Calculations **************************************
C***********************************************************************

c     Check if this is the first increment
      if (steptime .eq. 0) then
        Finn = Fin
        snn  = sn
        call MatProdInv(Fnn,Finn,Fenn)
        call ElasticStress(10*mprop,Fenn,Tnn)
      goto 12
      end if

c     We are not in the first increment -> Viscoplastic response
       call FlowRule(mprop,Fen,
     1      Fin,DFin,sn,Dsn,Tb,Tvp)
            Fi_ave = Fin + 0.5*DT*DFin
            s_ave  = sn  + 0.5*DT*Dsn
            F_ave = 0.5*(Fnn + Fn)
      call MatProdInv(F_ave,Fi_ave,Fe_ave)
      call FlowRule(mprop,Fe_ave,
     1      Fi_ave,DFi_ave,s_ave,Ds_ave,Tb,Tvp)
            Finn = Fin + DT * DFi_ave  !Finn using Midpoint scheeme.
            snn  = sn  + DT * Ds_ave

C     uncomment "goto␣10" to accept the calculations from Midpoint
C     and skip the "accuracy␣check␣and␣iteration␣scheme".

C         goto 10

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
CCCCCCCCCCC ACCURACY CHECK AND ITERATION CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

            FFE = Fin + dt*DFin       !Finn using Forward Euler(1. order)
      call MatNorm(FFE-Finn,temp1)
      call MatNorm(Finn-Fin,temp2)
      if (temp2 .lt. diff) goto 10
              eps = temp1/temp2
          if (eps .ne. eps) then
              print*, "␣"
              print*, "epsilon␣=␣NaN"
              print*, "Finn␣is␣not␣defined␣correctly"
              print*, "possible␣cause:␣Flow␣Rule␣blows␣up"
              pause
          end if
          if (eps .gt. kk) then     !Error is too big!
              dt_new = b*DT*(kk/eps)
              dt_temp = min(dt_new,c*DT)
              iter = ceiling(DT/dt_temp)
              dt_sugg = DT/iter
c             Same theory but with lower timestep
   20             if (iter .gt. MaxIter) goto 10
                  Fn2  = Fn
                  Fen2 = Fen
                  Fin2 = Fin
                  sn2  = sn
               do k = 1,iter
                          k2 = k
                          iter2 = iter
                          Fnn2 = Fn + (k2/iter2)*(Fnn-Fn)
                          F_ave = 0.5 * (Fnn2 + Fn2)
                      call FlowRule(mprop,Fen2,
     1                    Fin2,DFin2,sn2,Dsn2,Tb,Tvp)
                          Fi_ave = Fin2 + 0.5*dt_sugg * Dfin2
                          s_ave  = sn2  + 0.5*dt_sugg * Dsn2
                      call MatProdInv(F_ave,Fi_ave,Fe_ave)
                      call FlowRule(mprop,Fe_ave,
     1                    Fi_ave,DFi_ave,s_ave,Ds_ave,Tb,Tvp)
                          Finn2 = Fin2 + dt_sugg * DFi_ave
                          snn2 = sn2 + dt_sugg * Ds_ave
                      call MatProdInv(Fnn2,Finn2,Fenn2)
                                   !Checking Error again:
                                   FFE = Fin2 + dt_sugg*DFin2
                            call MatNorm(FFE-Finn2,temp1)
                            call MatNorm(Finn2-Fin2,temp2)
                            if (temp2 .lt. MinItStep) goto 30
                                eps = temp1/temp2
                            if (eps .gt. kk) then
                                dt_new = b*dt_sugg*(kk/eps)
                                dt_temp = min(dt_new,c*dt_sugg)
                                iter = ceiling(DT/dt_temp)
                                dt_sugg = DT/iter
                                goto 20
```

```
                              end if
 30                        Fn2 = Fnn2
                           Fen2 = Fenn2
                           Fin2 = Finn2
                           sn2  = snn2
                 end do
                    Fenn = Fenn2
                    Finn = Finn2
                    snn  = snn2
              end if

CCCCCCCCCC END ACCURACY CHECK AND ITERATION CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

      !calculate Fenn using Fnn and Finn, and calculate the stress Tnn
 10   call MatProdInv(Fnn,Finn,Fenn)
      call ElasticStress(mprop,Fenn,Tnn)

c     Rotate the stress tensor:
12    call TMatProd3(R,Tnn,R,Tnn)

C**********************************************************************
C******** Main Calculations Finished -> Storing variables ****************
C**********************************************************************

c     Updating the stress
      stressnew(km,1) = Tnn(1,1)
      stressnew(km,2) = Tnn(2,2)
      stressnew(km,3) = Tnn(3,3)
      stressnew(km,4) = Tnn(1,2)

c     store statevariables
      do i = 1,3
        Statenew(km,i) = Finn(i,i) - 1.0
      end do
      Statenew(km,4) = Finn(1,2)
      Statenew(km,5) = Finn(2,1)
      StateNew(km,6) = snn - s0

 100  continue

      return
      end

C**********************************************************************
C**************** Arruda Boyce Theory - subroutines *********************
C**********************************************************************

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CC              FLOWRULE AND VISCOPLASTIC DAMPENING
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      Subroutine FlowRule(mprop,Fe,Fi,DFi,s,sdot,Tb,Tvp)
```

```
         implicit none
         Double precision :: Tr_Tvp,tau,gammadotp
         Double precision :: Tr_Dev_Tvp2,s,sdot,temp,const
         Double precision :: gammadot0,h,flowconst,s0,sss
         Double precision,Dimension(3,3)::F,Fe,Fi,DFi,Te
         Double precision,Dimension(3,3)::B,Be,Bi,Tb
         Double precision,Dimension(3,3)::Tvp, Dev_Tvp, dev_Tvp2
         Double precision,Dimension(3,3)::Identity,Nij,D,R
         Double precision,Dimension(8)::mprop
         integer::k,l
c     material parameters
         gammadot0        =   mprop(5)
         h                =   mprop(6)
         sss              =   mprop(7)
         flowconst        =   mprop(8)
c     Preparations
         Tr_Tvp = 0.0
         temp   = 0.0
         tau    = 0.0
         do k = 1,3
           do l = 1,3
             D(k,l)          = 0.0
             Identity(k,l)   = 0.0
             Te(k,l)         = 0.0
             Tvp(k,l)        = 0.0
             dev_Tvp(k,l)    = 0.0
             Nij(k,l)        = 0.0
           end do
         end do
C     Main calculations
         call MatIdent(Identity)
         call ElasticStress(mprop,Fe,Te)
         call FeBackStressFeT(mprop,Fe,Fi,Tb)
              Tvp = Te - Tb
         call MatTrace(Tvp,Tr_Tvp)
              dev_Tvp = Tvp - (Tr_Tvp/3)*Identity
              do k = 1,3
               do l = 1,3
                  temp = temp + dev_tvp(k,l)**2.0
               end do
              end do
              tau = sqrt(0.5)*sqrt(temp)
         if (tau .gt. 0) then       !We have plastic deformation
                Nij=(sqrt(0.5)/tau)*dev_Tvp
         end if
              gammadotp=gammadot0*
    1         exp(-(flowconst*s)*(1-((tau/s)**(5.0/6.0))))
              D = gammadotp*Nij
         call MatProd(D,Fi,DFi)
              sdot = h*(1-(s/sss))*gammadotp
         return
      end subroutine FlowRule
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CC                 CALCULATE THE BACKSTRESS
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      Subroutine FeBackStressFeT(mprop,Fe,Fi,Tb)
      implicit none
          Double precision::Tr_Bi, Lambdapch, Cr, N, temp, IL,I,IV
          Double precision,Dimension(3,3)::Fe,Fi,Fe_Tb_FeT,Bi
          Double precision,Dimension(8)::mprop
          Double precision,dimension(3,3)::Q,BB,Tb,C,C_hat,U,V,R
          Double precision,dimension(3)::EigV,EigC
          Double precision::third,detFe
          integer::k,l
c     material parameters
          Cr            = mprop(3)
          N             = mprop(4)
c     Preparations
          Lambdapch = 0.0
          temp      = 0.0
          IL        = 0.0
          do k = 1,3
                EigV(k) = 0.0
                do l = 1,3
                    BB(k,l)         = 0.0
                    Bi(k,l)         = 0.0
                    Q(k,l)          = 0.0
                    Fe_Tb_FeT(k,l)  = 0.0
                end do
          end do
C     Main Calculations
          call MatDet(Fe,detFe)
          call MatIdent(C_hat)
          call TMatProd(Fi,Fi,C)
          call DSYEVJ3(C,Q,eigC)
          do k = 1,3
            c_hat(k,k) = sqrt(eigC(k))
          end do
          call TMatProd3(Q,c_hat,Q,U)
          call MatProdInv(Fi,U,R)
          call MatProdInv(Fi,R,V)
          Q = 0.0
          call DSYEVJ3(C,Q,EigV)
               IV = EigV(1) + EigV(2) + EigV(3)
               Lambdapch=sqrt(IV/3.0)
               temp=Lambdapch/sqrt(N)
               call InvLang(Temp,IL)
            do k=1,3
        BB(k,k)=(Cr/3.0)*sqrt(N)*IL*(EigV(k)-IV/3.0)/Lambdapch
            end do
          call MatProdT3(Fe,BB,Fe,Fe_Tb_FeT)
          Tb = (1.0/detFe)*Fe_Tb_FeT
          return
      end subroutine FeBackStressFeT

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CC          CALCULATE THE ELASTIC STRESS RESPONSE
CC     This subroutine uses the material properties and the elastic
CC     deformation gradient to calculate the elastic stress
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      Subroutine ElasticStress(mprop,Fe,Te)
      include 'VABA_PARAM.INC'
          Double precision :: DetFe,G,Lambda,Tr_lnsqrtBe
          Double precision,Dimension(3,3)::Te,Fe,Be,lnsqrtBe
          Double precision,Dimension(3,3)::temp
          Double precision,Dimension(8)::mprop
          integer :: i,j
c     Material properties:
          G       = mprop(1)
          Lambda  = mprop(2)
c     preparations:
          DetFe = 0.0
          do i = 1,3
            do j = 1,3
                Te(i,j)       = 0.0
                temp(i,j)     = 0.0
                Be(i,j)       = 0.0
                lnsqrtBe(i,j) = 0.0
            end do
          end do
c     Main calculations
          call MatDet(Fe,DetFe)
          call MatProdT(Fe,Fe,Be)
          call TMatProd(Fe,Fe)
          ! Using Pade approximation or analytical solution
          ! to find: henky strain = ln(sqrt(Be))
          !call PadeApprox(Be,lnsqrtBe)
          call PadeExact(Be,lnsqrtBe)
          tr_lnsqrtBe = lnsqrtBe(1,1)+lnsqrtBe(2,2)+lnsqrtBe(3,3)
          do i = 1,3
            do j = 1,3
                Te(i,j) = 2*G*lnsqrtBe(i,j)
                if (i .eq. j) then
                    Te(i,j) = Te(i,j) + Lambda*tr_lnsqrtBe
                end if
            end do
          end do
          temp = Te
          Te = (1.0/DetFe)*Temp
          return
      end subroutine ElasticStress
```

Appendix C

Explanation to Subroutines Used in VUMAT

- PadeApprox(Mat,lnsqrtMat): Calculates the logarithm and square root of a matrix, $lnsqrtMat$ using Mat (Padé approach)
- PadeExact(Mat,lnsqrtMat): Calculates the logarithm and square root of a matrix, $lnsqrtMat$ using Mat. (Analytical approach)
- InvLang(x,L): Calculates the inverse langvin, L, using x
- MatProdInv3(A,B,C,D): Calculates $D = ABC^{-1}$
- TMatProd3(A,B,C,D): Calculates $D = A^T BC$
- MatProdT3(A,B,C,D): Calculates $D = ABC^T$
- MatProdInv(A,B,C): Calculates $C = AB^{-1}$
- TMatProd(A,B,C): Calculates $C = A^T B$
- MatProdT(A,B,C): Calculates $C = AB^T$
- MatInv(A,B): Calculates: $B = A^{-1}$
- MatTrans(A,B): Calculates: $B = A^T$
- MatDet(A,B): Calculates $B = det[A]$
- MatProd(A,B,C): Calculates $C = AB$
- MatIdent(I): Defines I as the $3x3$ identity matrix
- MatTrace(A,b): Calculates $b = trace(A)$
- MatNorm(A,b): Calculates $b = ||A||_2$
- DSYEVJ3(A, Q, W): Calculates the eigenvalues, W, and the eigenvector matrix, Q, from the matrix A.

Appendix D Strain Triaxiality

An analysis regarding strain triaxiality was made during the Master's project. The strain triaxiality is simulated using a constant displacement ratio m. The following figures are presented in the appendix in case future students need them for reference.

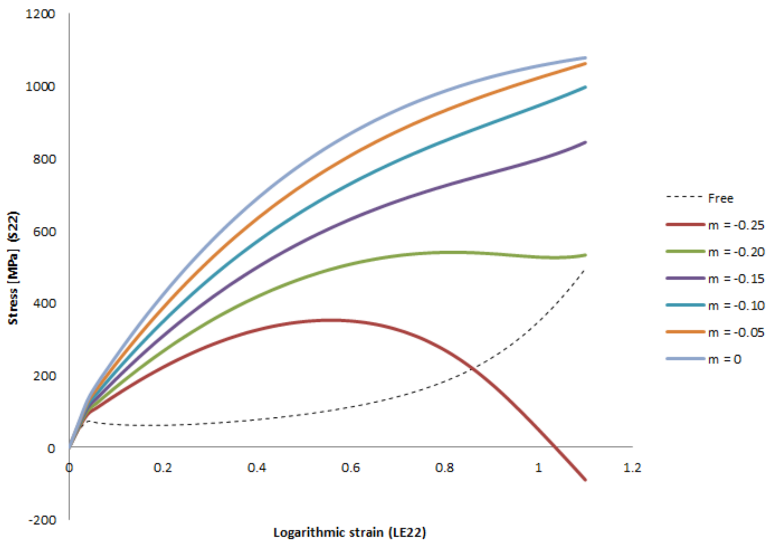


Figure D.1: Stress in the y-direction vs strain in the y-direction at the given strain ratios

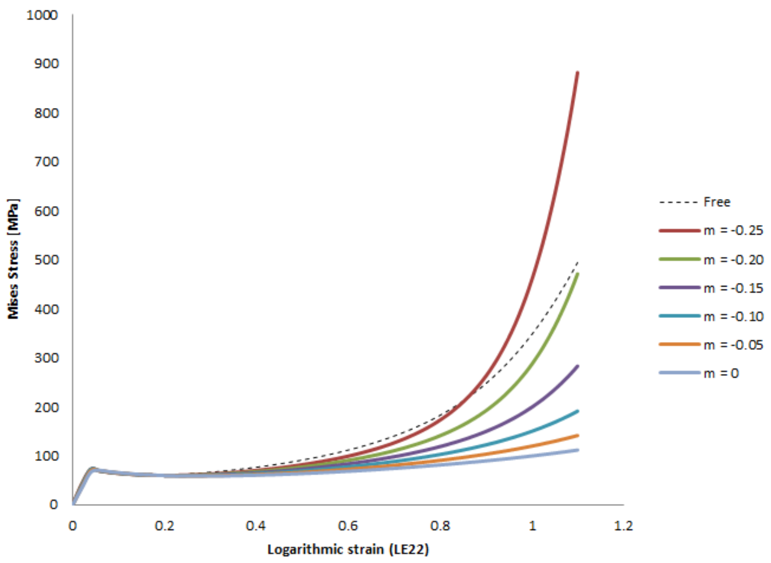


Figure D.2: This figure illustrates the Mises stress vs strain in the y-direction

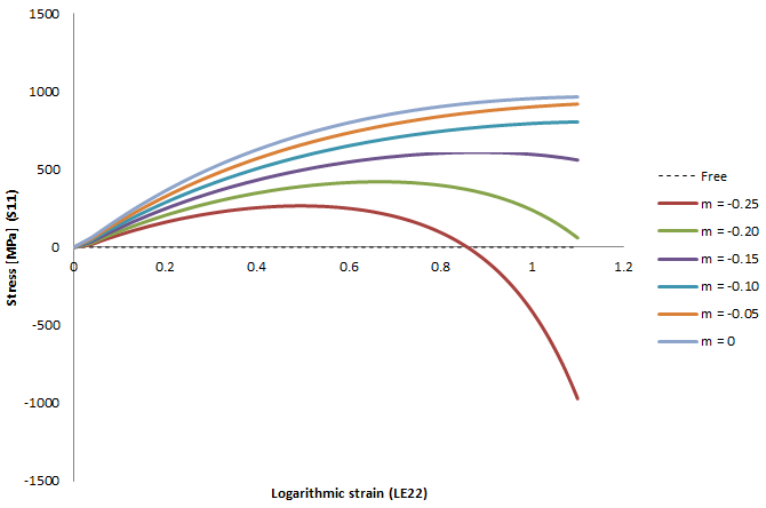


Figure D.3: This figure illustrates the stress in the radial direction vs strain in the y-direction

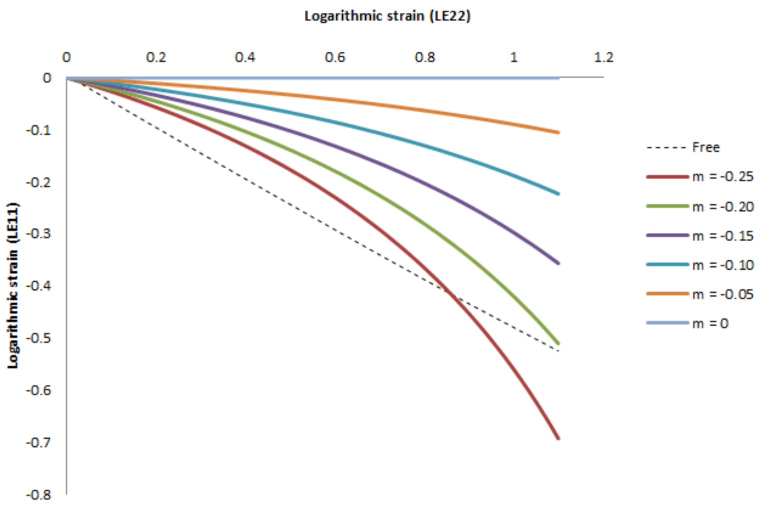


Figure D.4: This figure illustrates the strain at the given displacement rates

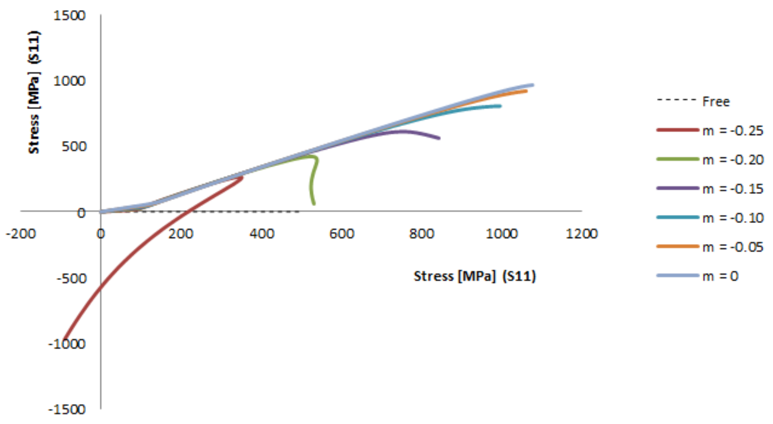


Figure D.5: This figure illustrates the stress ratio at the given displacement rates