Developing in house Multiphysics in-house solver ALYA for Biomechanics Applications

Author: Waleed Ahmad Mirza

Advisor:
Mariano Vazquez
Eva Casoni
Matias Rivero

Date: September 1, 2016

Acknowledgement

I would like to express my deepest gratitude goes to my advisors Mariano Vazquez, Eva Casoni for providing me the opportunity to pursue internship at BSC. I greatly appreciate their patience, support and motivation throughout my internship. Their endless guidance kept me pushing in the intent of broadening my limited knowledge and understanding of constitutive modelling and computational methods. A very special thanks goes out to my co-advisor Mathias Raviro, who provided me with direction, technical support, many encouraging conversations and fruitful discussions that helped me overcome the faltering steps encountered during my research. Moreover, I am also grateful to the entire staff of Computer Application in Science and Engineering Department of BSC for inspiring me and providing me with such a conducive research oriented learning atmosphere.

Abstract

The aim of this project is to develop the in-house multiphysics solver Alya for the study of arterial mechanics. In general, arterial wall consist of isotropic layers consisting of extracellular matrix and anisotropic layers consisting of matrix and set of collagen fibers. Therefore with an aim to capture the arterial mechanics, isotropic and anisotropic constitutive material model are implemented in Alya. The implemented isotropic material model are Neohookean and Mooney Rivlin hyperelastic models whereas the implemented anisotropic material model is Holzaphal model for soft tissues. Numerical studies consisting of single hexahedral element and geometries with multiple hexahedral elements are performed to verify the implementation. Results when compared against literature and other solvers such as Code Aster and Abaqus, show a reasonable agreement. Lastly, a model of artery subjected to blood pressure is simulated using the Holzaphal constitutive model and Neohookean model with one and multiple processors. The results of both cases demonstrate the capability of implemented models to run efficiently in multiple processors and achieve good scalability.

Notations

$\mathbf{F}$	Deformation Gradient
$\boldsymbol{x}$	Spatial Coordinates
$\boldsymbol{X}$	Undeformed Coordinates
$\boldsymbol{u}$	Displacement
$\mathbf{P}$	First Piola Krichoff stress
$\boldsymbol{b}$	Body Force
$\mathbf{S}$	Second Piola Krichoff stress
$\boldsymbol{w}$	Virtual Displacement
B_0	Undeformed Configuration
B	Deformed Configuration
$\ddot{\mathbf{x}}$	Acceleration
ρ_0	Initial Density
$\mathbf{f}_{int}$	Internal Force Vector
$\mathbf{f}_{ext}^{(b)}$	External Force Vector
$\mathbf{K}$	Stiffness Matrix
${}_{L}\mathbf{K}$	Linear Stiffness Matrix
${}_{NL}\mathbf{K}$	Non-Linear Stiffness Matrix
${}_{L}\mathbf{B}$	Linear Strain Displacement Matrix

${}_{NL}\mathbf{B}$	Non Linear Strain Displacement Matrix
$\mathbf{C}$	Right Green Cauchy Tensor
$\mathbb{C}$	4th Order Tangent Constitutive Matrix
$\mathbf{E}$	Green Langrangian Strain
I_1	First invariant of $\mathbf{C}$
I_2	Second invariant of $\mathbf{C}$
I_3	Third invariant of $\mathbf{C}$
Ψ	Strain Energy Function
Ψ_{vol}	Volumetric part of Strain Energy Function
$\bar{I}_c^{(i)}$	Distortional part of I_i
$\boldsymbol{\sigma}$	Cauchy Stress
$\mathbf{b}$	Left Green Cauchy Tensor
a_f, b_f, u_0, κ_0	Material Parameters

Contents

Chapter 1 Introduction

1.1 Motivation

Computational bio-mechanics is a broad area that aims to simulate biological systems using numerical methods. The ultimate aim [1] of researchers in this discipline is to build computational models based on histology of biological systems in order to understand and capture the working of the biological systems. Computational bio-mechanics is particularly interesting for researchers in area of solid mechanics as it poses an important challenge of constitutive modeling of biological tissues. Constitutive models govern the macroscopic behavior of the biological systems in response to external stimulus due to constituents of the material. The constitutive modeling of biological tissues pose a challenge for taking into the account of the fact that these materials are alive and show response to minute external stimulus [2] . Further challenges include multiscale modeling of biological tissues that captures the relevant properties of all scales.

With advancements in computational bio-mechanics in general and constitutive modeling of biological tissues in particular, patient specific models can be built with wide range of applications involving clinical prevention, diagnoses and treatment of disease and injuries.

1.2 Overview

Biological tissues are broadly classified into either soft or hard tissues. The later includes mineralization tissues with firm inter-cellular matrix. In human beings such tissues exist in bones, tooth enamel, dentin etc. Their behavior can be modeled using linear elastic constitutive model as they undergo very small deformations[2]. Soft tissues on the other hand occur in tendons, ligaments, muscles, arteries etc. and are composed of extra-cellular matrix of collagen and fibre embedded in ground matrix. Such tissues undergo very high deformation and can only be modeled using non- linear material models. Moreover a ground substance in these tissues gives nearly incompressible hyperelastic behavior to them.

The properties of soft tissues are broadly classified into two namely active and passive[2]. The active properties are the ones contingent on biophysical processes such as metabolism, growth etc. Whereas the passive

properties only depend on the composition of tissues. The constitutive models addressed in this work only account for the passive properties of the soft tissues.

In the literature, traditionally the passive properties of soft tissues are modeled as follow: Most of the data available in the literature apply small strains $< 10\%$ to tissues of interest to assume linear elastic behavior [3]. For soft tissues undergoing large deformations, isotropic and anisotropic [4] material models are employed. These material models take into account the histology and physiology of the tissues in different spatial and time scales. At simpler level the soft tissue are modelled as composite materials with individual components such as matrix and fibre. Hence a constitutive model would be a model the gives the additive (uncoupled) behavior of each component under different biophysical stimuli.

1.3 Goals

Alya is the BSC in-house HPC-based multi-physics simulation code [9]. It is designed from scratch to run efficiently in parallel supercomputers, solving coupled problems. The source code of ALYA is divided into several modules. The Solidz module of Alya is both an implicit and explicit FE solver developed in Fortran and is capable of solving finite-strain nonlinear three-dimensional solid mechanics problems. The computational models implemented in this module have been extensively tested, verified and validated [10].

The aim of this project is to replicate and implement existing computational material models in Alya. These computational material models target at capturing passive constitutive modeling of soft tissues for bio-mechanical applications such as modeling flow of blood in arteries, cardiovascular modeling etc. In the literature, passive response of the soft tissues has been studied using isotropic and anisotropic hyperplastic models. Isotropic hyperelastic models such as Neohookean and Mooney Rivlin models are used to study behavior of arterial layers with isotropic matrix. Whereas the transverse isotropic models such as Holzaphel model for arterial walls are used for to take into account the directional properties of the collagen fibres.

In short the aim of this project is:

1- To implement following material models in Alya: 1) Neohookean hyperelastic model 2) Mooney Rivlin hyperelastic model 3) Holzaphel transverse isotropic hyperelastic model.
2- Qualitatively and quantitatively verify these implementations against results in publications and against commercial softwares such as Code Aster and Abaqus.
3- Develop the above mentioned implementations for parallel processing environment.

1.4 Outline

The report is organized as follow:

Chapter 2

Chapter 2 introduces the readers to the architecture of Solidz module of Alya and describes the implicit implementation of the finite element algorithm.

Chapter 3

In chapter 3, a set of hyperelastic constitutive equations capable of reproducing the basic passive properties of biological tissues is proposed. First, several hyperelastic formulations are reviewed from the literature and then implementation of the two hyperelastic models namely Neo-Hookean and Mooney Rivlin are described in detailed and verified in different case studies.

Chapter 4

In Chapter 4, a literature review on anisotropic hyperelastic models is presented. A detailed description and formulation of the ingredients of Holzaphel anisotropic hyperelastic model is described in depth. The implemtation is verified agaisnt numerical studies performed in literature.

Chapter 5

A case study is presented involving a quarter model of artery subjected to blood pressure. The case study have been solved using one (series) and multiple processors (parallel) to make sure the developed constitu-

tive models work efficiently in BSC's MPI environment as well.

Chapter 6

The achievements and learning outcomes of this study are summed up, final conclusions are drawn and future work lines are outlined.

Chapter 2 Implementation of Alya/ Implicit

The objective of this chapter is to give an overview of the fundamental equations and their implementation in finite element algorithm of the Alya Solidz module. A special focus has been put on explaining the implicit algorithm of Alya. The chapter is organized as follow: First, a continuum mechanical framework involving definition and notation of the variables used in finite element implementation is explained. This framework will serve as basis for constitutive models presented in the forthcoming chapters. In later part of chapter, the algorithm used to implement implicit Total Lagrangian formulation (TL) algorithm is mentioned.

2.1 Continuum Finite Deformation Framework

For clarity of readers before explaining the finite deformation framework, its important to explain notations involved. Scalar, vectors, second order tensors and fourth order tensors are denoted by light faced italics (a,b), boldface lower case letters ($\mathbf{a}$,$\mathbf{b}$), bold face upper case letter ($\mathbf{A}$,$\mathbf{B}$) and black board capital letters ($\mathbb{A}, \mathbb{B}$) respectively. Multiplication of two second order tensors is presented as $\boldsymbol{AB}$. Tensor products between two vectors and tensors is presented as $\mathbf{a} \otimes \mathbf{b}$ and $\mathbf{A} \otimes \mathbf{B}$ respectively.

Let φ be a function that maps a material point $\mathbf{X}\, \epsilon\, B_0$ in the reference configuration to its corresponding point $\mathbf{x} = \varphi\,(\mathbf{X})\, \epsilon B$ in the deformed configuration (see figure 2.1). The deformation gradient tensor $\mathbf{F}$ is defined as:

$$\mathbf{F} := Grad\mathbf{x} = \bigtriangledown_0 \mathbf{x} \tag{2.1}$$

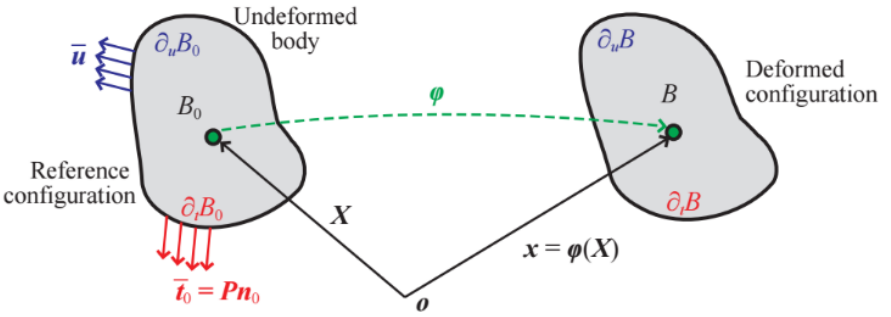

Figure 2.1: Schematic of deformed body and undeformed body

where $\boldsymbol{\nabla}_0$ is the gradient operator with respect to the reference configuration. In Einsteain notation, the components of $\mathbf{F}$ are given by $F_{i,j} = \frac{\partial x_i}{\partial x_J}$. Since $\mathbf{x(X)=X+u(X)}$, with $\mathbf{u}$ the displacement vector, thus the deformation gradiet can be given by $\mathbf{F} = \mathbf{I} + \boldsymbol{\nabla}_0\mathbf{u}$, and $\boldsymbol{\nabla}_0\mathbf{u}$ is the displacement gradient.

2.2 Governing equation

The governing equation to be solved numerically in solid mechanics is the equation of balance of momentum in terms of the reference configuration.

$$Div\mathbf{P} + \mathbf{b}_0 = \rho\ddot{\mathbf{x}}, \qquad \forall \mathbf{X}\epsilon B_0 \tag{2.2}$$

where ρ_0 is the mass density(per unit reference volume) and Div is the divergence operator with respect to the reference configuration, such that $Div\mathbf{P} = \nabla_0.\mathbf{P}$. In the above expression, tensor $\mathbf{P}$ and vector b_0 stand for, respectively, the first Piola-Kirchhoff stress and the distributed body force on the undeformed body.

Typically, the first Piola-Kirchhoff stress is determined from Second Piola Krichoff Stresss $\mathbf{S}$ by following expression $\mathbf{S} = \mathbf{F}^{-1}\mathbf{P}$.

2.3 Finite element spatial discretization

2.3.1 Weak form

To derive weak form of equation(2.2)lets assume an arbitrary admissible displacement field $\boldsymbol{w}$ such that $\boldsymbol{w}(\boldsymbol{X}_{\partial_t B_0}) = 0$ where $X_{\partial_t B_0} \in \partial_t B_0$. The weak form of the balance of momentum (2.2) is as follow:

$$\int_{B_0} Div\mathbf{P}.\mathbf{w}dV + \int_{B_0} \mathbf{b}_0.\mathbf{w}dV = \int_{B_0} \rho_0\ddot{\mathbf{x}}.\mathbf{w}dV \tag{2.3}$$

Equation (2.3) is a week form of equation (2.2) in a sense that it satisfies the balance of momentum in integral sense against test function $\boldsymbol{w}$ over the domain B_0. While equation fulfil the condition locally at each point $\boldsymbol{X}\epsilon B_0$. After integration equation (2.3) by part and assuming the applied tractions $\bar{\boldsymbol{t}}_0 = \mathbf{P}\boldsymbol{n}_0$, following equation is obtained.

$$\int_{\partial_t B_0} \bar{\mathbf{t}}_0 \mathbf{w} dS + \int_{B_0} \mathbf{b}_0.\mathbf{w} dV = \int_{B_0} \mathbf{P}.Grad\mathbf{w} dV + \int_{B_0} \rho_0 \ddot{\mathbf{x}}.\mathbf{w} dV \quad (2.4)$$

Where the First Piola Krichoff Stress $\mathbf{P(F)}$ is defined by the material constitutive law and is a function of the deformaion gradients $\mathbf{F}$ and material parameters.

2.3.2 Finite element discritization [12]

Let $\mathbf{x}$ be polynomial approximation of degree k to actual statial coordinates $\mathbf{x} = \varphi(\mathbf{X})$, such that:

$$\mathbf{x}\epsilon\mathbf{X}^k = \mathbf{x}_h \epsilon C^0[B_0^h] : \mathbf{x}_h \mid \omega_0^e \epsilon \mathbb{P}^k(\omega_0^e) \forall \omega_0^e \epsilon B_{0h}$$

The polynomial approxiamtion of actual displacments is w_h such that:

$$w_h \epsilon X_c^k = w_h \epsilon X_h^k : w_h \mid \partial_u \omega_0^e = 0 \forall \omega_0^e \epsilon B_{0h} \quad (2.5)$$

where $B_{0h} = U_e \omega_0^e$ is the fintie element approximation of an actual undeformed body. Now writing equation (4) in terms of approximate functional fields.

$$\int_{\partial_t B_{0h}} \bar{t}_0.w_h dS + \int_{B_{0h}} b_0.w_h dV = \int_{B_0} P.Grad w_h dV + \int_{B_{0h}} \rho_0 \ddot{x}_h.w_h dV \quad (2.6)$$

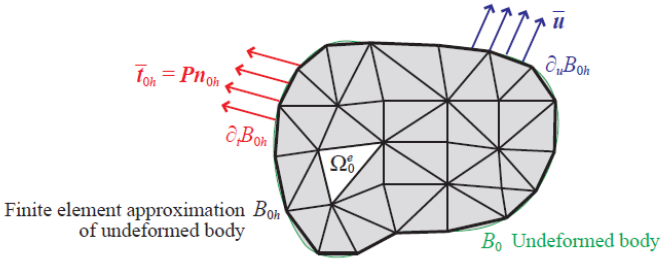

Figure 2.2: Finite element discretization of unreformed body B_0 discretised into finite elements.

Polynomial shape function $N^a(\xi)$ of an element are defined to approximate the spatial coordinates such that they have value 1 on node a

and 0 on rest of the nodes. Using the polynomials interpolation function the approximated spatial fields are defined as:

$$x_h^e(\mathbf{X}) = \sum_{a\epsilon\omega_0^e} N^a(\xi)x^a \qquad and \qquad \mathbf{w}_h^e(\mathbf{X}) = \sum_{a\epsilon\omega_0^e} N^a(\xi)\mathbf{w}^a \quad (2.7)$$

Now writing equation (6) in form of the dicritized interpolation polynomials $N^a(\xi)$

$$\sum_e \int_{\omega_0^e} (\rho_0 N^a N^b dV)\ddot{\mathbf{x}}^a.w^b + \sum_e \int_{\omega_0^e} (\mathbf{P}(N^a x^a)\nabla_0 N^b dV).\mathbf{w}^b$$

$$= \sum_e \int_{\partial_t\omega_0^e} (\bar{\mathbf{t}}_0 N^b dV).\mathbf{w}^b + \sum_e \int_{\omega_0^e} (\mathbf{b}_0 N^b dV).\mathbf{w}^b \quad (2.8)$$

Where $\mathbf{x}^a\epsilon\mathbb{R}^3$ such that for all $\mathbf{w}^b$, with $\mathbf{w}^b = 0$ and $\partial_t\omega_0^e$. In short notation form, the above equation can be written as:

$$\sum_e M^{e,ba}\ddot{\mathbf{x}}^a + \sum_e \mathbf{f}_{int}^{e,b}(\mathbf{x}^a) = \sum_e \mathbf{f}_{ext}^{e,b}. \quad (2.9)$$

Where $M^{e,ba} = \int_{\omega_0^e} \rho_0 N^a N^b dV$

$$\mathbf{f}_{int}^{e,b}(\mathbf{x}^a) = \int_{\omega_0^e} \mathbf{P}(N^a x^a)\nabla_0 N^b dV$$

$$\mathbf{f}_{ext}^{e,b} = \int_{\partial_t\omega_0^e} \bar{\mathbf{t}}_0 N^b dS + \int_{\omega_0^e} \mathbf{b}_0 N^b dV$$

2.3.3 Newton-Raphson iteration

The Newton Raphson scheme will be presented in context of static analysis. In order to initiate Newton-Raphson iteration, the initial nodal displacement of the first load step are set to zero.

$$x_{n+1}^{a,i=0} = 0$$

Where superscript 'i' indicates the iteration step, $i = 0$ is the initial guess before iteration and subscript n indicate the loading step.

Now $r^{b,i-1}$ be the remainder or residual of the balance of momentum equation at iteration step $i - 1$.

$$r^{b,i-1} := f_{int}^b(x_{n+1}^{a,i-1}) - f_{ext,n+1}^b \quad (2.10)$$

Using a Taylor expansion around the value at iteration $i-1$, the balance of momentum equation can be linearized into:

$$f^b_{int}(x^{a,i-1}_{n+1}) + \frac{\partial f^b_{int}}{\partial x^a}(x^{a,i-1}_{n+1}) \triangle x^a_{n+1} = f^b_{ext,n+1} \tag{2.11}$$

As ($\triangle x^a_{n+1} = x^{a,i}_{n+1} - x^{a,i-1}_{n+1}$).

$$x^{a,i}_{n+1} = x^{a,i-1}_{n+1} - [A^{ba}(x^{a,i-1}_{n+1})]^{-1} r^{b,i-1} \tag{2.12}$$

Where $K_{ba} := \frac{\partial f^b_{int}}{\partial x^a} = A^{ba}(x^{a,i-1}_{n+1})$ represents the Jacobian matrix (or global stiffness matrix K^{ba}). Next section explains in detailed derivation on the formulation of the stiffness matrix in context of Total Lagrangian formulation. The iteration is continued until convergence is reached with the criteria $|x^{a,i}_{n+1} - x^{a,i-1}_{n+1}|$ <tolerance.

2.3.4 Stiffness matrix

In context of finite strain deformation, the stiffness matrix is formulated as follow:

$$\mathbf{K}^{i-1}_{n+1} = {}_L\mathbf{K}^{i-1}_{n+1} + {}_{NL}\mathbf{K}^{i-1}_{n+1} \tag{2.13}$$

$${}_L\mathbf{K}^{i-1}_{n+1} = \int {}_L\mathbf{B}^{\Gamma_{i-1}}_{n+1} \mathbb{C}^{i-1}_{n+1} {}_L\mathbf{B}^{i-1}_{n+1} dV \tag{2.14}$$

$${}_{NL}\mathbf{K}^{i-1}_{n+1} = \int {}_{NL}\mathbf{B}^{\Gamma_{i-1}}_{n+1} \mathbb{C}^{i-1}_{n+1} {}_{NL}\mathbf{B}^{i-1}_{n+1} dV \tag{2.15}$$

Where ${}_L\mathbf{B}$ and ${}_{NL}\mathbf{B}$ are linear and non-linear strain displacement transformation matrices.Furthermore, a detailed derivation of $\mathbb{C}$ and $\mathbf{S}$ will be a topic of next chapter.

2.4 Algorithm of Alya-Implicit

A detailed algorithm of Alya-Implicit is mentioned in flow chart illustrated in figure 2.3. This flow chart summarizes all the steps previously mentioned in this chapter.

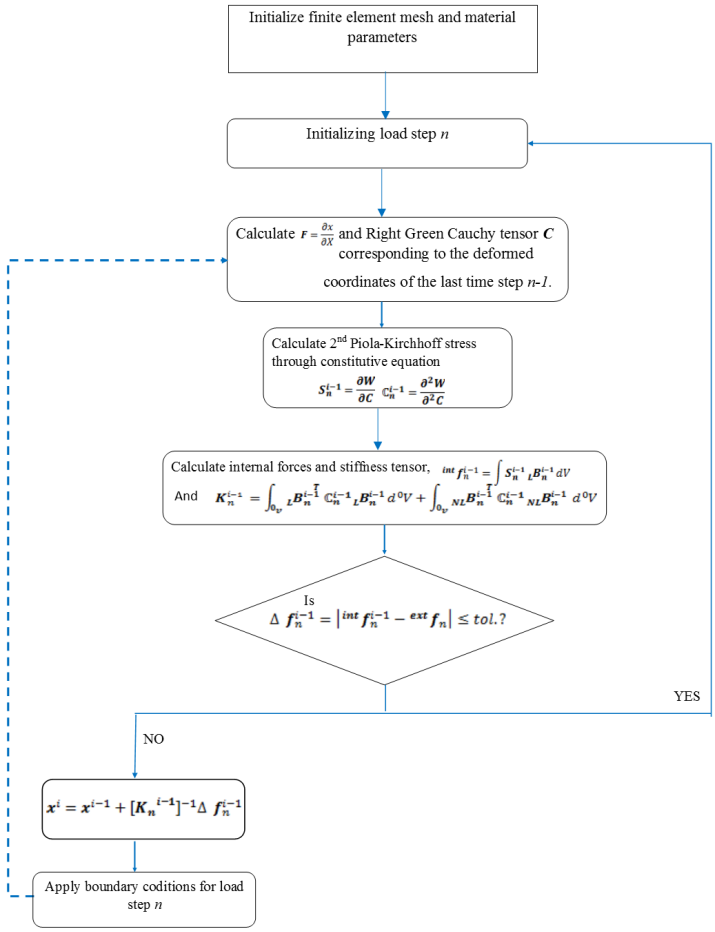

Figure 2.3: Algorithm of Alya-Implicit

Chapter 3 Isotropic constitutive models for soft tissues

Adequate modelling of the constitutive properties of soft tissues is significant for reproducing the biomechanical problems in numerical studies. In the literature, more models are based on the theories of continuum mechanics concepts than discrete mechanics. In continuum mechanics the main assumption is that the properties of the medium are represented by the continuous functions with continuous derivatives. The following represents a brief overview of such continuum mechanics based theories.

Fung [3], was among the pioneers to characterize the behaviour of soft tissues by presenting a potential function. Later on many other models such as Neohookean and Mooney Rivlin [2],[3] hyperelastic models were employed to capture the behaviour of soft tissues. Since experimentation [3] has revealed that soft tissues are highly compressible with poison ratio ranging from 0.48- 0.49, therefore most of these models assumes that the soft tissues are homogenous, isotropic and nearly incompressible. Based on these assumption the strain energy function can be formulated in terms of first, second and third invariant or in form of principle stretches like in case of Ogden model [2]. Furthermore these strain energy functions are usually expressed in exponential, logarithmic and polynomial function.[2],[3]

In this chapter, two of the most commonly used isotropic hyperelastic models namely Neohookean hyperelastic model and Mooney Rivlin hyperelastic model presented, implemented in Alya and verified against numerical studies conducted in open source software Code Aster.

3.1 Hyperelastic formulations

The idea behind the formulation of hyperelastic model is that:

1. The strain energy function must be zero in unloaded state $\psi(\mathbf{F} = \mathbf{I}) = 0$

2. The strain energy function should grow monotonously $\psi(\mathbf{F}) \geq 0$

3. The work performed should be the path followed $\int_{\Gamma_1} \mathbf{S} : d\mathbf{E} = \int_{\Gamma_2} \mathbf{S} : d\mathbf{E}$

4. For the closed deformation cycle, the deformation work must be zero $\oint \mathbf{S} : d\mathbf{E} = 0$.

In the literature several the isotropic hyperelastic models have been proposed such as Neohookean , Mooney Rivlin , Fung model etc. These hyperelastic models have strain energy function a function of invariants of Right (and Left) Green Cauchy $\mathbf{C}$ tensor as shown:

$$\Psi(\mathbf{C}) = \Psi(I_1, I_2, I_3) \tag{3.1}$$

Where the invariants of Right Green Cauchy tensor $\mathbf{C}$ are as follow:

- First invariant: $I_1 = Tr(\mathbf{C}) = C_{ii}$
- Second invariant: $I_2 = \frac{(\mathbf{C}:\mathbf{C}-I_2)}{2} = \frac{(C_{ij}C_{ji}-C_{ii}^2)}{2}$
- Third invariant: $I_3 = det(\mathbf{C}) = J^2$

As mentioned before in this chapter, two of the most commonly used isotropic hyperelastic models namely Neohookean hyperelastic model and Mooney Rivlin hyperelastic model will be discussed.

3.1.1 Neohookean Hyperelastic model

The strain energy function for the Neohookean model is an additive split of volumetric part and distortional part.

$$\Psi(\mathbf{C}) = \bar{\Psi}_0(\bar{\mathbf{C}}) + \Psi_{vol}(J) \tag{3.2}$$

$$\Psi(\mathbf{C}) = \mathbf{C}_{10}(\bar{I}_1 - 3) + D_1(J-1)^2 \tag{3.3}$$

Where the material parameters such as C_{10} and D are related to the shear μ and the bulk modulus κ as follows:

$$C_{10} = \frac{\mu}{2}; D = \frac{\kappa}{2} \tag{3.4}$$

$\bar{I}_1 = J^{-\frac{2}{3}} I_1$ is the distortional part of the first invariant of the Right Green Cauchy tensor $\mathbf{C}$.As in finite element formulation require the use of the second order tensor like Second Piola Kirchhoff Stress $\mathbf{S}$ and fourth order tensor like tangent constitutive matrix $\mathbb{C}$. The Second Piola Krichoff Stress $\mathbf{S}$ being second derivative of the strain energy function is expressed as follow:

$$\mathbf{S} = 2.\frac{\partial \bar{\Psi}_0(\bar{\mathbf{C}})}{\partial \mathbf{C}} + 2.\frac{\partial \Psi_{vol}(J)}{\partial \mathbf{C}} \tag{3.5}$$

$$\mathbf{S} = 2C_{10}J^{-\frac{2}{3}}(\mathbf{I} - \frac{1}{3}I_1\mathbf{C}^{-1}) - pJ\mathbf{C}^{-1} \tag{3.6}$$

Where

$$p = 2D(J-1)$$

The fourth order tangent constitutive tensor $\mathbb{C}$ being the second derivation of the strain energy function is expressed as follow:

$$\mathbb{C} = 4.\frac{\partial^2 \bar{\Psi}_0(\bar{C})}{\partial \mathbf{C}.\partial \mathbf{C}} + 4.\frac{\partial^2 \Psi_{vol}(J)}{\partial \mathbf{C}.\partial \mathbf{C}} \tag{3.7}$$

$$\mathbb{C} = \frac{4}{3}C_{10}J^{-\frac{2}{3}}(\frac{1}{3}Tr(\mathbf{C})\mathbf{C}^{-1} \otimes \mathbf{C}^{-1} - \mathbf{I} \otimes \mathbf{C}^{-1} - \mathbf{C}^{-1} \tag{3.8}$$

$$\otimes \mathbf{I} + Tr(\mathbf{C})\Pi_{C^{-1}}) - pJ(\mathbf{C}^{-1} \otimes \mathbf{C}^{-1} - 2\Pi_{C-1})$$

Where $[\Pi]_{ijkl}$ is the fourth order identity tensor given by:

$$[\Pi]_{ijkl} = \frac{1}{2}(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}) \tag{3.9}$$

And Π_{C-1} is the fourth order tensor given by:

$$[\Pi_{C-1}]_{ijkl} = \frac{1}{2}([C^{-1}]_{ik}[C^{-1}]_{jl} + [C^{-1}]_{il}[C^{-1}]_{jk}) \tag{3.10}$$

3.1.2 Mooney Rivlin hyperelastic formulation

The Mooney Rivlin formulation consist of strain energy function which is the linear combination of the distortional part of the first and second invariant of the Right Green Cauchy tensor $\boldsymbol{C}$ shown as follow:

$$\Psi(\mathbf{C}) = C_{10}.(\bar{I}_1 - 3) + C_{01}.(\bar{I}_2 - 3) + D(J-1)^2 \tag{3.11}$$

Where the material constants C_{10} and C_{01} are related by the following expression:

$$D = \frac{2(C_{01} + C_{10})(1 + \nu)}{3(1 - 2\nu)} \tag{3.12}$$

Where ν is the poisson ratio.
The corresponding strain energy function $\mathbf{S}$ is as follow:

$$S_{ij}^{vol} = 2D(J - 1)JC_{ij}^{-1} \tag{3.13}$$

$$S_{ij}^{iso} = 2(C_{10}J^{-\frac{2}{3}}.(\delta_{ij} - \frac{1}{3}.\mathbf{C}_{ij}^{-1}I_1) + C_{01}.J^{-\frac{4}{3}}(I_1\delta_{ij} - C_{ij} - \frac{2}{3}.C_{ij}^{-1}I_2)) \tag{3.14}$$

The tangent constitutive matrix $\mathbb{C}$ for Mooney Rivlin model is as follow,

$$\mathbb{C} = 4.\frac{\partial^2\bar{\Psi}_0(\bar{C})}{\partial\mathbf{C}.\partial\mathbf{C}} + 4.\frac{\partial^2\bar{\Psi}_{vol}(J)}{\partial\mathbf{C}.\partial\mathbf{C}} \tag{3.15}$$

$$\mathbb{C} = 4C_{10}.\frac{\partial^2\bar{I}_1}{\partial C_{ij}.\partial C_{kl}} + 4C_{01}\frac{\partial^2\bar{I}_2}{\partial C_{ij}.\partial C_{kl}} - pJ(C_{ij}^{-1} \otimes C_{kl}^{-1} - 2\Pi_{C^{-1}}) \tag{3.16}$$

Where

$$\frac{\delta^2 I_1}{\delta C_{ij}\delta C_{kl}} = I_3^{-\frac{1}{3}}.(C_{ki}^{-1}.C_{lj}^{-1}I_1 - C_{ij}^{-1} \tag{3.17}$$

$$.\delta_{kl} - C_{kl}^{-1}.\delta_{ij} + \frac{1}{3}.C_{kl}^{-1}.C_{ij}^{-1}I_1)$$

$$\frac{\delta^2 I_2}{\delta C_{ij}.\delta C_{kl}} = -\frac{2}{3}.I_3^{-\frac{2}{3}}.C_{kl}^{-1}.(\bar{I}_1.\delta_{ij} - C_{ij} - \frac{2}{3}. \tag{3.18}$$

$$C_{ij}^{-1}I_2) + I_3^{-\frac{2}{3}}.(\delta_{kl}.\delta_{ij} - \delta_{ik}.\delta_{jl} + \frac{2}{3}.C_{ki}^{-1}.$$

$$C_{lj}^{-1}I_2 - \frac{2}{3}.C_{ij}^{-1}.(I_1\delta_{kl} - C_{kl})$$

$$\frac{\delta^2 J}{\delta C_{ij}.\delta C_{kl}} = \frac{1}{4}.I_3^{\frac{1}{4}}.(C_{kl}^{-1}.C_{ij}^{-1} - 2.C_{ki}^{-1}.C_{lj}^{-1}) \tag{3.19}$$

3.2 Verification of the implemented models

The above mentioned constitutive models are implemented in Alya/Solidz solver in a Fortran 90 subroutine. The implemented subroutines have been shown in appendix A and B. Now the next step is to verify the implementation for which the following one element and multi element numerical studies have been carried out and results are compared with Code Aster.

1. Unit volume hexahedral element subjected to axial load.

2. Unit volume hexahedral element subjected to biaxial load.

3. Unit volume hexahedral element subjected to in plane shear load.

4. Cantiliver thing beam comprising of 3750 solid hexhedral elements subjected to out of the plane end loading.

The schematic of numerial experimentation on single hexahedral element is illustrated in figure (3.1) . A brief description of each test is as follow:

1. In uniaxial test a pressure of 10 unit is applied on the surface formed by nodes 5678. While surface formed by nodes 1234 has all degree of freedoms constrained.

2. In biaxial test a pressure of 10 unit is applied on the surfaces formed by nodes 5678 and 2368. While surfaces formed by nodes 1234 and 1457 has all degree of freedoms constrained.

3. In shear test, a displacement of the prescribed displacements are $u_{23} = u_{33} = u_{63} = u_{83} = 0.5$ units.

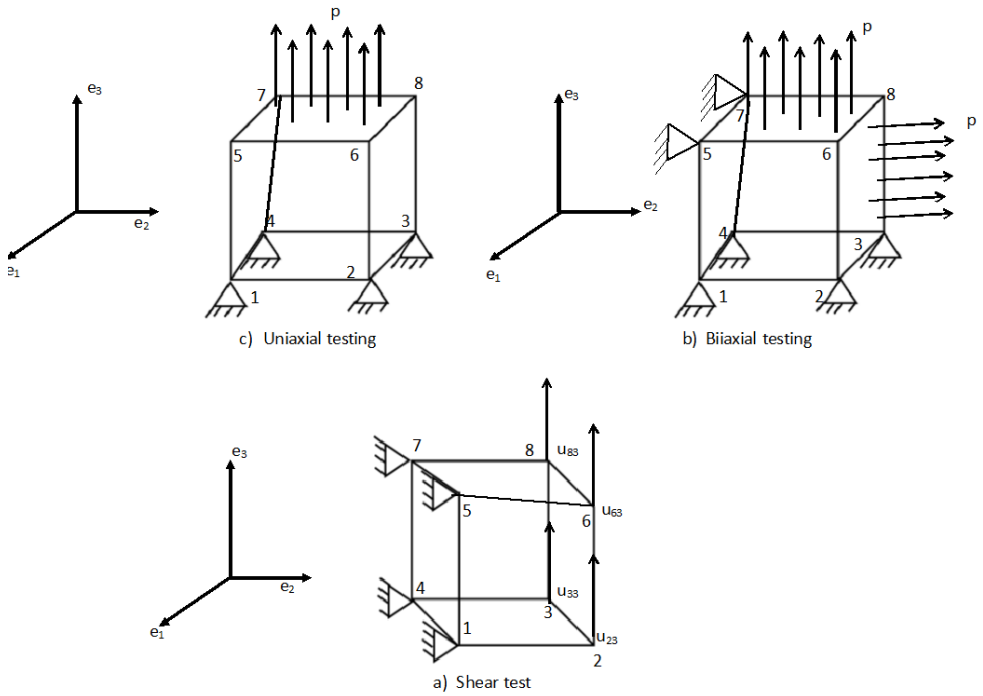

Figure 3.1: Configuration of single element case studies

3.2.1 Single element studies using Neohookean model

The results of single element uniaxial, biaxial and shear test has been for the Neohookean model are tabulated in table 3.1, 3.2 and 3.3 respectively.The material properties that are used in the analysis are $C_{10} = 1, D = 0.04$

Degree of freedom	u_{51}	u_{52}	u_{53}	u_{61}	u_{62}	u_{63}	u_{71}	u_{72}	u_{73}	u_{81}	u_{82}	u_{83}
Code Aster	0.1745216	0.1745216	0.6947036	0.1745216	0.1745216	0.6947036	0.1745216	0.1745216	0.6947036	0.1745216	0.1745216	0.6947036
Alya-Solidz	0.1745216	0.1745216	0.6947036	0.1745216	0.1745216	0.6947036	0.1745216	0.1745216	0.6947036	0.1745216	0.1745216	0.6947036
Percentage relative error	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0

Table 3.1: Neohookean model: Comparison of results of uniaxial test from Code Aster and Alya

Degree of freedom	u_{61}	u_{62}	u_{63}	u_{81}	u_{82}	u_{83}
Code Aster	0.60302997	0.42918513	0.6030296	0.60302997	-0.42918513	0.6030299
Alya-Solidz	0.60302992	0.42918515	0.6030299	0.60302992	-0.42918515	0.603011
Percentage relative error	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0

Table 3.2: Neohookean model: Comparison of results of biaxial test from Code Aster and Alya

Degree of freedom	u_{21}	u_{22}	u_{23}	u_{31}	u_{32}	u_{33}	u_{61}	u_{62}	u_{63}	u_{81}	u_{82}	u_{83}
Code Aster	0.247533E-001	0.247533E-001	0.247533E-001	0.247533E-001	0.247533E-001	0.247533E-001	0.247533E-001	0.247533E-001	0.247533E-001	0.247533E-001	0.247533E-001	0.247533E-001
Alya-Solidz	0.247533E-001	0.247533E-001	0.247533E-001	0.247533E-001	0.247533E-001	0.247533E-001	0.247533E-001	0.247533E-001	0.247533E-001	0.247533E-001	0.247533E-001	0.247533E-001
Percentage relative error	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0

Table 3.3: Neohookean model: Comparison of results of shear test from Code Aster and Alya

3.2.2 Single element studies using Mooney Rivlin Model

The results of single element uniaxial, biaxial and shear test has been for the Neohookean model are tabulated in table 3.4, 3.5 and 3.6 respectively. The material properties used in this analysis are D=0.04, $C_{10} = 1$ and $C_{01} = 0.5$

Degree of freedom	u_{51}	u_{52}	u_{53}	u_{61}	u_{62}	u_{63}	u_{71}	u_{72}	u_{73}	u_{81}	u_{82}	u_{83}
Code Aster	0.16134937	0.16134937	0.638458	-0.16134937	0.16134937	0.638458	-0.16134937	-0.16134937	0.638458	0.16134937	-0.16134937	0.63845826
Alya-Solidz	0.16134937	0.16134937	0.638458	-0.16134937	0.16134937	0.638458	-0.16134937	-0.16134937	0.638458	0.16134937	-0.16134937	0.63845826
Percentage relative error	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0

Table 3.4: Mooney Rivlin model: Comparison of results of uniaxial test from Code Aster and Alya

Degree of freedom	u_{61}	u_{62}	u_{63}	u_{81}	u_{82}	u_{83}
Code Aster	0.535520465	0.3863573	0.535520465	0.535520465	-0.3863573	0.535520465
Alya-Solidz	0.53552048	0.38635717	0.53552048	0.53552048	-0.3853571	0.53552048
Percentage relative error	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0

Table 3.5: Mooney Rivlin model: Comparison of results of biaxial test from Code Aster and Alya

Degree of freedom	u_{21}	u_{22}	u_{23}	u_{31}	u_{32}	u_{33}	u_{61}	u_{62}	u_{63}	u_{81}	u_{82}	u_{83}
Code Aster	0.9898	0.98713	0.5	0.9898	-0.96713	0.5	0.9898	-0.2426	0.5	0.9898	0.2426	0.5
Alya-Solidz	0.98932	0.96711	0.5	0.98932	-0.96711	0.5	0.98932	-0.242623	0.5	0.98932	0.242623	0.5
Percentage relative error	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0

Table 3.6: Mooney Rivlin model: Comparison of results of shear test from Code Aster and Alya

3.2.3 Cantilever beam test

The test case involves a cantilever beam ($length = 1unit, width =$ 0.3 unit and height $= 0.1unit$) consisting a mesh of $50 * 15 * 5$ eight node hexahedral elements. The model shows a very reasonable mesh convergence (relative error of $1e{-}8$) for this mesh. The beam is subjected

to end pressure of 0.1 unit as illustrated in figure 3.2. As can be seen the end pressure is applied on an area of $0.3 * 0.02\ unit^2$.

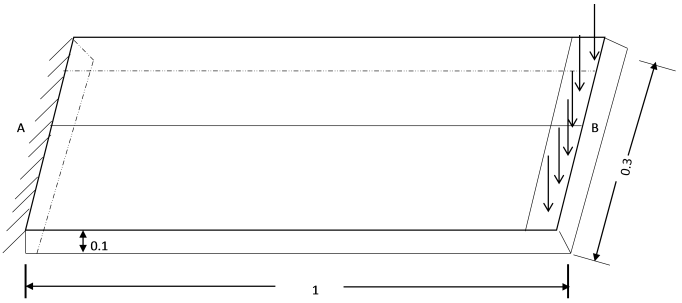

Figure 3.2: Configuration of cantiliver beam case study

A static implicit analysis is carried out using 10 load/displacement boundary condition increments. The reason for this test is two fold. Firstly, to check the accuracy and stability of the implemented constitutive model for geometries involving several finite elements. Secondly, to do a comparative analysis between Neohookean and Mooney Rivlin model.
Displacement magnitude contours obtained from Neohookean and Mooney Rivlin model from Alya-Solidz are displayed in figure 3.3. The displacement magnitude along line AB (as shown in figure 3.3) are plotted in figure 3.4. Moreover figure 3.4(top) serves to compare Mooney Rivlin model and Neohooken model. To evaluate the convergence rate of the Neohookean and Mooney Rivlin model the residual errors of the norm of energy gradient $\| \bigtriangledown E\|$ is plotted in figure 3.5.

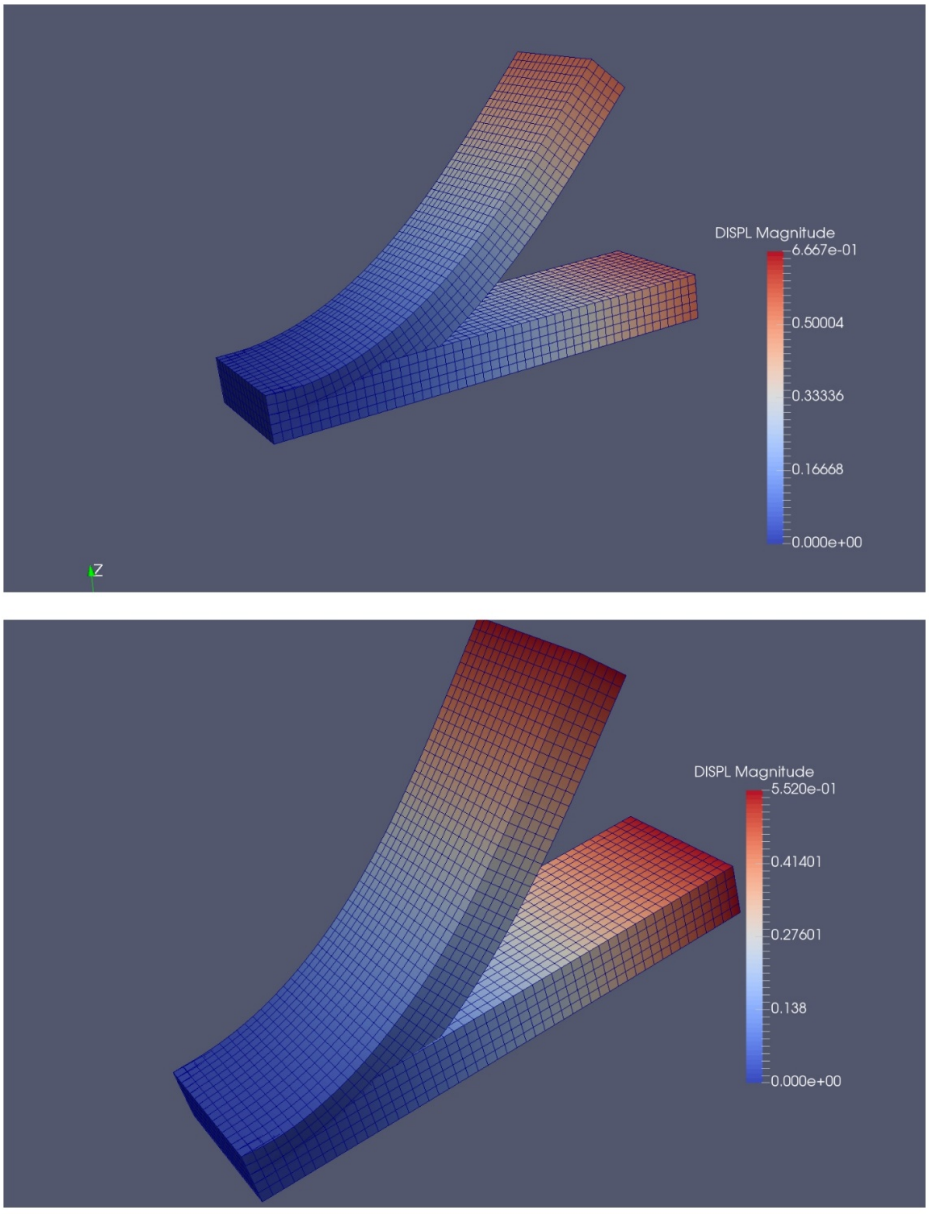

Figure 3.3: Displacement magnitude obtained in the cantiliver beam from Neohooken (top) amd Mooney Rivlin model (bottom)

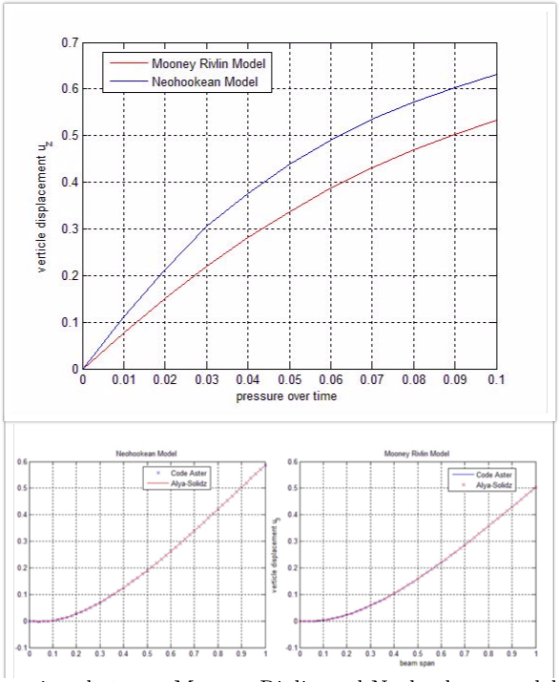

Figure 3.4: (Top)Comparison between Mooney Rivlin and Neohookean model from from Alya-Solidz, (Bottom left) Comparison of results between Code Aster and Alya-Solidz for Neohookean model, (Bottom Right) Comparison of results between Code Aster and Alya-Solidz for Mooney Rivlin model1

The convergence rate of simulations run for both models is shown in figure 3.5. As can be seen for both models the convergence rate is linear at start of every increment and becomes nearly quadratic as solver approaches to the solution. This behaviour of the solver is quite understandable as Newton Raphson method gives nearly quadratic convergence rate local to the solution. Nevertheless, a better convergence rate (almost quadratic) can be achieved if analysis is performed with more load increments.

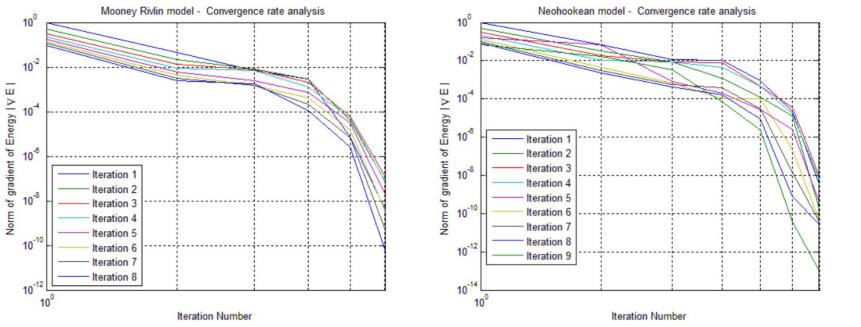

Figure 3.5: Convergence analysis of a) Neohookean model b) Mooney Rivlin model

3.3 Conclusions and discussion

From the aforementioned numerical studies we can establish the following conclusions.

1. The implemented Neohookean and Mooney Rivlin subroutines in Alya-Solidz are very accurate as can be seen in comparison to Code Aster. This serves to verify the implementation.

2. Mooney Rivlin model exhibit more nonlinearity than neoheookean model (as can be seen in figure 3.4) and hence is more recommended for biomechanics studies involving finite deformation.

3. On log-log plot shown in figure 3.5 a near quadratic rate is observed local to the solution which establishes the robustness of the implemented material models.

Chapter 4 Anisotropic hyperelasticity

Biological soft tissues such as arteries, myocardium muscles, ligaments, tendons etc. show highly anisotropic behaviour. Thus, incorporating hyperelasticity in constitutive modelling of soft tissues is an important aspect. The histology of tissues dictate that there are up to two set of collagen fibers in soft tissues that confer upon anisotropic properties to the soft tissues. The simplest form of anisotropy can be represented by transverse isotropy where the set of collagen fibers are aligned in a specific direction. The most widely used transverse isotropic model has been presented by Holzaphel et al[4,8]. Holzaphel et al presented several constitutive model for soft tissues, each with different features such as [4,8,11]:

1. Rate independent transverse isotropic constitutive model for myocardium tissues consisting of one family of fibers.

2. Rate independent transverse isotropic constitutive model for arterial tissues consisting of two families of collagen fibers.

3. Rate independent transverse isotropic constitutive model for arterial tissues with dispersed collagen fibers.

4. Elastoplastic constitutive model for soft tissues.

Above mentioned constitutive models from 1-3 describes behaviour of soft tissues within strain range before plasticity is induced in the material. In this work, the implementation of the second constitutive model i.e. Rate independent transverse isotropic constitutive model for arterial tissues consisting of two families of collagen fibers is presented. The reason of selecting this material model is due to its extensive usage in literature for capturing the mechanics of various arteries.
The outline of the chapter is as follow. In Section 4.1, the theoretical framework to be used as the background for the description of the arterial mechanics is discussed. This consists of the general equations governing the isotropic and anisotropic response of arterial tissues based on the use of an elastic free-energy function. In Section 4.2, all required ingredients that are necessary to implement the Holzaphel constitutive model are derived from the free energy functions. Section 4.3 discuss

the verification of the implemented Holzaphel model using several case studies from the literature.

4.1 Constitutive model for soft tissues

Arterial layers can be seen as composites reinforced by two families of collagen fibres arranged in symmetrical spirals. The isochronic strain energy Ψ function is additively split into isotropic $\bar{\Psi}_{iso}$ and anisotropic $\bar{\Psi}_{aniso}$ functions. At low blood pressure, the collagen fibers don't play any role and entire behaviour of the artery is governed by $\bar{\Psi}_{iso}$ part, whereas at high pressure, the fibers ($\bar{\Psi}_{aniso}$ part)govern the response of arteries. Hence the potential is written as:

$$\bar{\psi}(\bar{\mathbf{C}}, \boldsymbol{a}_{01}, \boldsymbol{a}_{02}) = \bar{\psi}_{iso}(\bar{\mathbf{C}}) + \bar{\psi}_{aniso}(\bar{\mathbf{C}}, \boldsymbol{a}_{01}, \boldsymbol{a}_{02}) \tag{4.1}$$

Where $a_i, i = 1, 2$ represents the reference directions of the two families of fibers.
In terms of invariants the free energy function is represented as,

$$\Psi(\bar{\mathbf{C}}, \mathbf{A}_1, \mathbf{A}_2) = \bar{\Psi}_{iso}(\bar{I}_1) + \bar{\Psi}_{aniso}(\bar{I}_4, \bar{I}_6) \tag{4.2}$$

Whereas the invariants are represented as:

$$I_1(\mathbf{C}) = tr\mathbf{C}, \qquad I_4(\mathbf{C}, \boldsymbol{a}_{01}) = \mathbf{C} : \mathbf{A}_1, \qquad I_6(\mathbf{C}, \boldsymbol{a}_{02}) = \mathbf{C}, \mathbf{A}_2 \tag{4.3}$$

Where $\mathbf{A}_1$ and $\mathbf{A}_2$ are defined as:

$$\mathbf{A}_1 = \boldsymbol{a}_{01} \otimes \boldsymbol{a}_{01}, \qquad \mathbf{A}_2 = \boldsymbol{a}_{02} \otimes \boldsymbol{a}_{02} \tag{4.4}$$

The isotropic behaviour is presented by Neo-Hookean model as follow:

$$\bar{\Psi}_{iso}(\bar{I}_1) = \frac{\mu}{2}(\bar{I}_1 - 3) \tag{4.5}$$

Where μ physically represents the shear modulus. The fact that collagen fibers are only active at high pressure promts to model anisotropic behavior with exponential terms as follow:

$$\bar{\Psi}_{aniso}(\bar{I}_4, \bar{I}_6) = \frac{a_f}{2b_f} \sum_{i=4,6} [exp[b_f(\bar{I}_i - 1)^2] - 1] \tag{4.6}$$

The volumetric part Ψ_{vol} is presented as follow:

$$\Psi_{vol}(J) = \frac{1}{2}\kappa(J-1)^2 \tag{4.7}$$

From the strain energy function the distortional part of Cauchy stress can be derived as:

$$\boldsymbol{\sigma} = \frac{1}{J}\mathbf{F}\frac{\partial \Psi}{\partial \mathbf{F}} \tag{4.8}$$

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}_{vol} + \bar{\boldsymbol{\sigma}}_{iso} + \boldsymbol{\sigma}_{aniso} \tag{4.9}$$

$$\bar{\boldsymbol{\sigma}}_{iso} = aJ^{-1}(\bar{\mathbf{b}} - \frac{1}{3}\bar{I}_1\mathbf{I}) \tag{4.10}$$

$$\bar{\boldsymbol{\sigma}}_{aniso} = 2a_f J^{-1} \sum_{i=4,6} (\bar{I}_i - 1)exp[b_f(\bar{I}_i - 1)^2](\bar{\mathbf{a}}_i \otimes \bar{\mathbf{a}}_i - \frac{1}{3}\bar{I}_i\mathbf{I}) \tag{4.11}$$

$$\boldsymbol{\sigma}_{vol} = \kappa_0(J-1)\mathbf{I} \tag{4.12}$$

Now the Second Piola Krichoff Stress is obtained by applying pull back operation on Cauchy stress i.e.

$$\mathbf{S} = \mathbf{J}\mathbf{F}^{-1}\boldsymbol{\sigma}\mathbf{F}^{-T} \tag{4.13}$$

Now the corresponding tangent constitutive matrix $\mathbb{C}$ for the isotropic strain energy function $\bar{\Psi}_{iso}(\bar{I}_1)$ is already presented in equation (4.5). Here the final form of tangent constitutive matrix derived for anisotropic strain energy function $\bar{\Psi}_{aniso}(\bar{I}_4, \bar{I}_6)$ is presented as follows:

$$\bar{\mathbb{C}}_{aniso} = 4 \sum_{i=4,6} \{\frac{1}{3}\psi^{\alpha}[\bar{I}_i(\mathbf{C}^{-1} \odot \mathbf{C}^{-1} - \frac{1}{3}\mathbf{C}^{-1} \otimes \mathbf{C}^{-1})] \tag{4.14}$$

$$-J^{-\frac{2}{3}}Dev(\mathbf{a}_0 \otimes \mathbf{a}_0) \otimes \mathbf{C}^{-1} - J^{-\frac{2}{3}} \otimes Dev(\mathbf{a}_0 \otimes \mathbf{a}_0)]$$

$$+J^{-\frac{4}{3}}\psi^{\alpha\alpha}Dev(\mathbf{a}_0 \otimes \mathbf{a}_0) \otimes Dev(\mathbf{a}_0 \otimes \mathbf{a}_0)\}$$

Where

$$\psi^{\alpha\alpha} = a_f[1 + 2b_f(\bar{\mathbf{I}}_i - 1)^2]exp[b_f(\bar{I}_i - 1)^2] \tag{4.15}$$

$$\psi^{\alpha} = a_f(\bar{I}_i - 1)exp[b_f(\bar{I}_i - 1)^2] \tag{4.16}$$

$$Dev(\mathbf{a}_0 \otimes \mathbf{a}_0) = \mathbf{a}_0 \otimes \mathbf{a}_0 - \frac{1}{3}\bar{I}_i\bar{\mathbf{C}}^{-1} \tag{4.17}$$

$$-(\mathbf{C}^{-1} \odot \mathbf{C}^{-1})_{ijkl} = -\frac{1}{2}(\mathbf{C}^{-1}_{ik}\mathbf{C}^{-1}_{jl} + \mathbf{C}^{e-1}_{jk}\mathbf{C}^{-1}_{il})$$

For detail derivation readers can refer to the appendix C.

4.2 Numerical Experiments

The Holzaphel model for arterial walls has been implemented in the subroutine sld-stress-model-134.f90 in Alya-Solidz. The implementation is verified using a number of quantitative and qualitative numerical studies involving one element and multi elements geometries. Results of numerical studies involving one element are compared with work of Nolan et al [5]. While the results of the multi-element geometry are compared with Abaqus 6.14. The numerical studies will serve to verify the implementation and establish it's good robustness.

4.2.1 Single element studies

The following single element studies are carried out and compared with results presented by Nolan et al.
In this case study the stretch λ_2 is imposed in direction 2 on a cube with unit volume, while no lateral stretch is permitted in 1 and 3 direction ($\lambda_1 = \lambda_3 = 1$) as shown in figure 4.1. This loading condition has bio-mechanical relevance as arterial tissues undergo high circumferential (hoop) strains but no radial or axis strain.

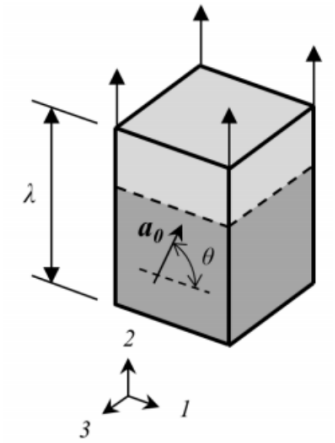

Figure 4.1: Schematic of uniaxial stretch test

A finite element analysis is carried out on a cube with unit volume (figure 4.1) with the prescribed boundary conditions as follows:

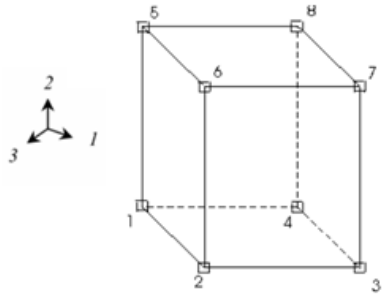

Figure 4.2: Schematic of geometry and nod

$$\begin{Bmatrix} u_{11} \\ u_{12} \\ u_{13} \end{Bmatrix} = \begin{Bmatrix} u_{21} \\ u_{22} \\ u_{23} \end{Bmatrix} = \begin{Bmatrix} u_{31} \\ u_{32} \\ u_{33} \end{Bmatrix} = \begin{Bmatrix} u_{41} \\ u_{42} \\ u_{43} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix}$$

$$\begin{Bmatrix} u_{51} \\ u_{52} \\ u_{53} \end{Bmatrix} = \begin{Bmatrix} u_{61} \\ u_{62} \\ u_{63} \end{Bmatrix} = \begin{Bmatrix} u_{71} \\ u_{72} \\ u_{73} \end{Bmatrix} = \begin{Bmatrix} u_{81} \\ u_{82} \\ u_{83} \end{Bmatrix} = \begin{Bmatrix} 0 \\ u \\ 0 \end{Bmatrix}$$

For $\lambda_2 = 1.2$, the value of prescribed displacement u is 0.2 unit. Hence the value of u is varied in range $[0, 0.2]$. The Cauchy stress component for the proposed uniaxial test for various angles θ is given in figure 4.2.

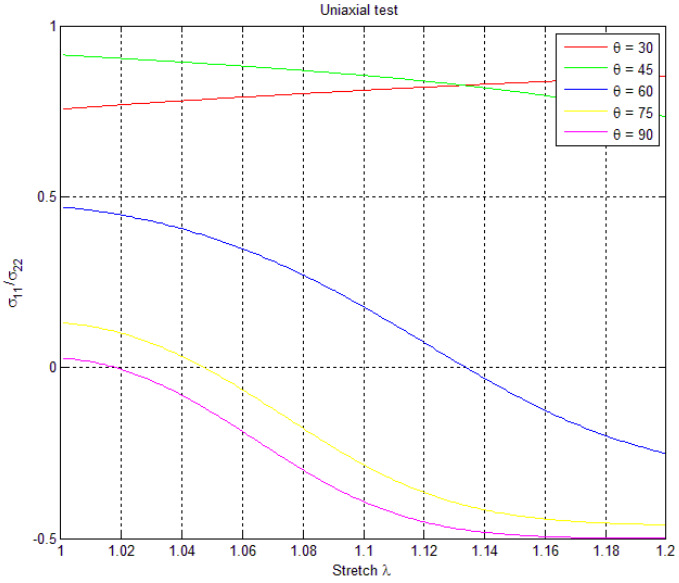

Figure 4.3: Uniaxial test: Ratio of Cauchy stress $\frac{\sigma_{11}}{\sigma_{22}}$ obtained for single set of fibers for θ in range [30 90]

4.3 Plain strain shear stress

In this case study, a pure in plane shear loading is applied to a cube with unit volume. This type of loading is pure isochoric and volume preserving i.e. $J = det(F) = 1$. The material properties employed for this study are $\mu = 0.05$, $\kappa = 1$,$a_f = 1$ and $b_f = 100$.
For finite element implementation the deformation gradient in equation (4.17) is translated into boundary conditions as shown in figure 4.4 and equation (4.18).

$$F = \begin{pmatrix} \sqrt{F_{12}^2 + 1} & F_{12} & 0 \\ F_{12} & \sqrt{F_{12}^2 + 1} & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.18}$$

$$|u_{1'}| = \frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2} - u_{2'}} \tag{4.19}$$

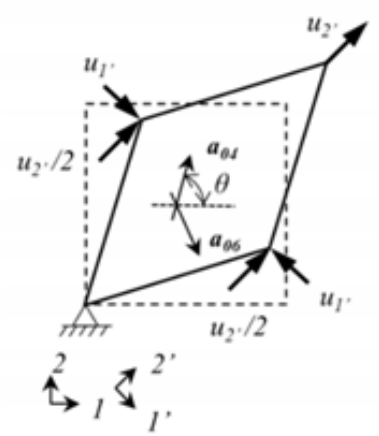

Figure 4.4: Schematic illustrating the kinematics of the pure shear deformation of the (1,2) section of a unit cube

For the in plane shear stress two numerical tests have performed: With 1)One set of fibers a_{01}. 2)Two set of fibers a_{01}, a_{02}. The first test has been performed at fiber orientations θ in range $[30°, 90°]$ and results are displayed in figure 4.5. For the second inplane shear test , two set of fibers are used and results are displayed for $\theta = 30$.

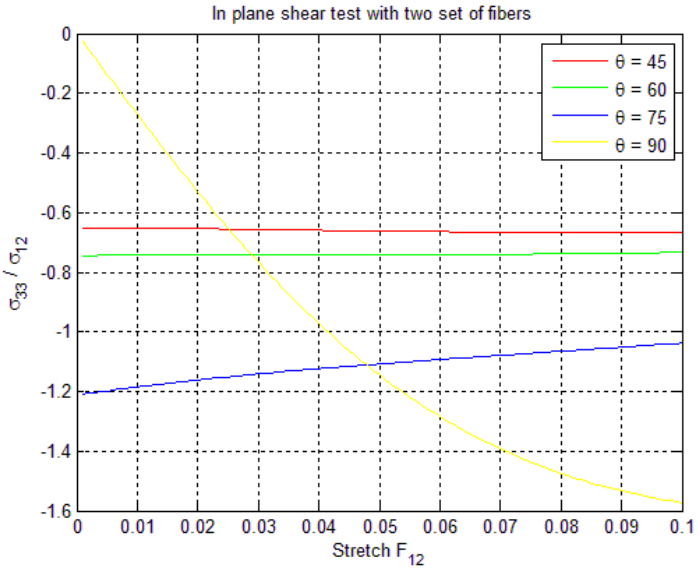

Figure 4.5: In plane shear test: Ratio of cauchy stress $\frac{\sigma_{33}}{\sigma_{12}}$ obtained for two set of fibers for θ in range $[45°, 90°]$

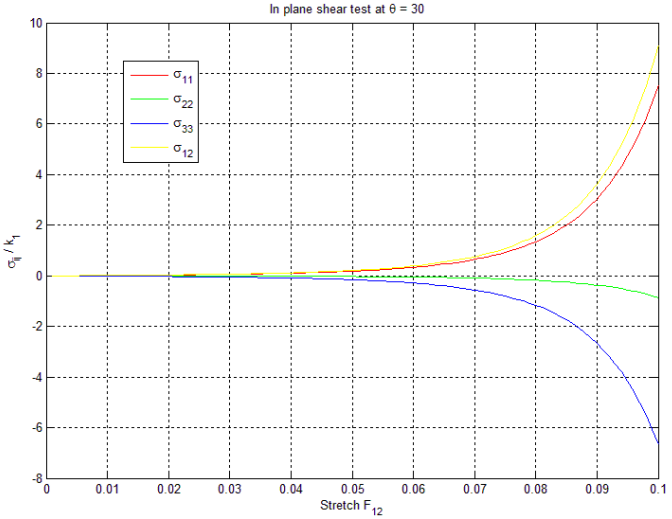

Figure 4.6: In plane shear test:Normalized Cauchy stress obtained for two set of fibers for $\theta = 30°$

4.3.1 A thin plate under axial load

In the proposed numerical study, a thin axial plate modelled using 4 node planar element is subjected to axial pressure load. The study is performed at using one set of fibres orientated θ at $+30°$, $-30°$ and $0°$. A schematic of the numerical study is presented in figure below. The material parameters used for this study are $D = 0.04, C_{10} = 1.0, a_f = 80.0, b_f = 5.0$ and $C_{01} = 0.5$.

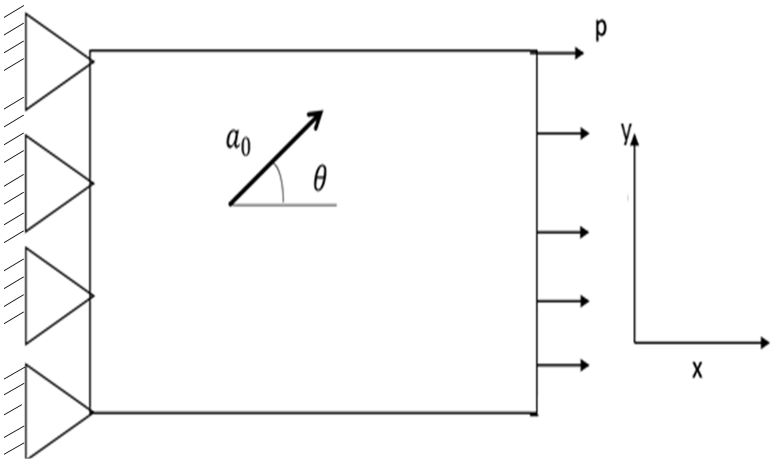

Figure 4.7: Schematic of thin plate under axial load.

The study is conducted in Alya and results are compared with a similar study in Abaqus. The study serves two purpose. 1) Verify the implementation of the Holzaphel model in Alya. 2) Demonstrated the effect of fibre orientation θ on the deformation.

At fibre orientation $\theta = 0°$

The applied pressure for this study is 10 units. A comparison of results from Alya and Abaqus are shown in figure 4.8. Since the the fibre orientation $\theta = 0°$, therefore the plate undergoes symmetric deflection about the x axis and (as will be seen in the following studies) the plate undergo less deformation compared to when $\theta = \mp 30°$. Maximum relative error of displacement field generated in Abaqus and Alya is 2.1%.

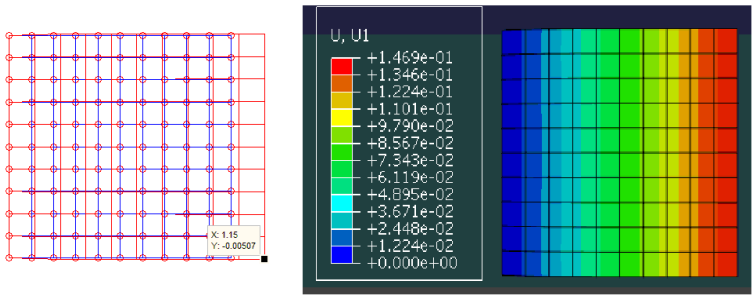

Figure 4.8: Fiber Orientation $\theta = 0°$, (Left) results from Alya, postprocessed in Matlab (Right) Results from Abaqus

At fibre orientation $\theta = 30°$

The applied pressure for this study is 10 units. A comparison of results from Alya and Abaqus are shown in figure 4.9. Since the the

fibre orientation is $\theta = 30°$., therefore the plate undergoes unsymmetric deflection about the x axis and the deflection tends to be more towards lower right corner of the plate than upper right corner. Maximum relative error of displacement field obtained from Abaqus and Alya is 3.9%.

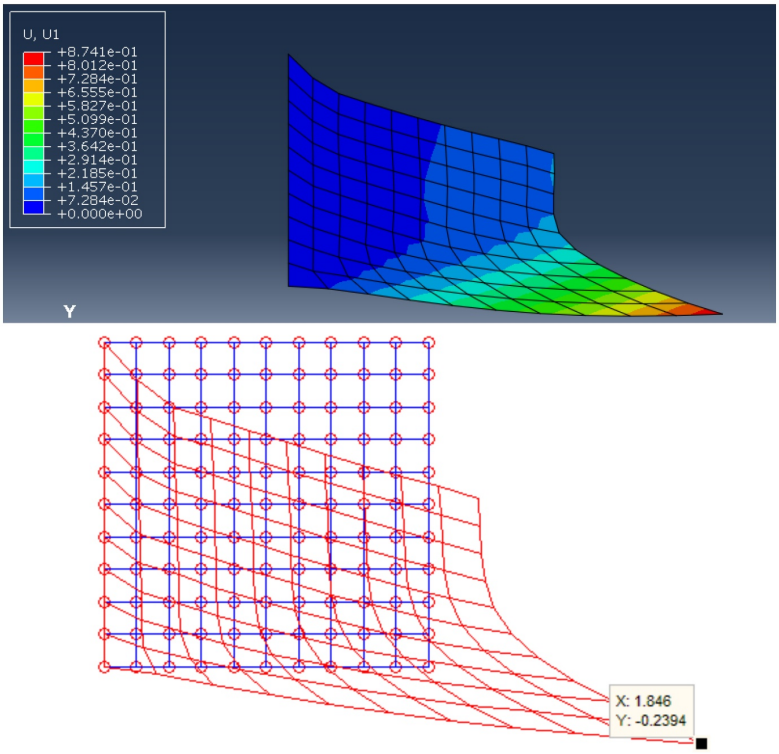

Figure 4.9: Fiber Orientation $\theta = 30°$, (Bottom) results from Alya, postprocessed in Matlab (Top) Results from Abaqus

At fibre orientation $\theta = -30°$

The applied pressure for this study is 10 units. A comparison of results from Alya and Abaqus are shown in figure 4.11. Since the the fibre orientation is $\theta = -30°$. therefore the plate undergoes unsymmetric deflection about the x axis and the deflection tends to be more towards upper right corner of the plate than lower right corner. Maximum relative error of displacement field obtained from Abaqus and Alya is 3.9%.

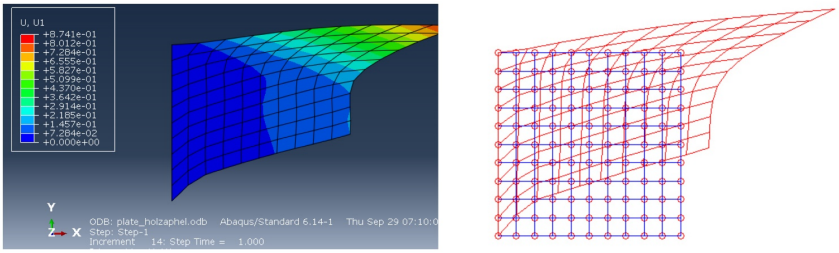

Figure 4.10: Fiber Orientation $\theta = -30°$, (Right) results from Alya, postprocessed in Matlab (Left) Results from Abaqus

4.4 Discussion and Conclusion

Following conclusions can be derived from one element and thin plate study:

1. The results of one element uniaxial and in-plane shear test show excellent similarity to the results displayed by Nolan et al [5] for similar studies. These results establishes the accuracy of the implementation.

2. The thin plate study for fibre orientated θ at $\mp 30°, -30°$ and $0°$ shows a maximum of 4% difference from results obtained from Abaqus. The difference in the values if displacement is coming from the fact that the volumetric term of the strain energy function $\Psi_{vol}(J)$ in Abaqus is different from the one implemented in Alya, as shown in equation [7].

$$\Psi_{vol}(J) = \frac{1}{D}(\frac{(J)^2 - 1}{2} - lnJ) \tag{4.20}$$

   As the plate is undergoing increase in volume because of the axial load therefore $J \neq 1$, therefore the volumetric term $\Psi_{vol}(J)$ played a role in producing the difference in values of the displacement.

3. The thin plate study illustrated the effect of fibre orientation on the displacement. It can be seen in figure that for $\theta = 0°$ the plate undergoes symmetric and lesser deformation compare to $\theta = \mp 30°$ as fibers are oriented along the axial pressure load. For $\theta = \mp 30°$ the plate undergoes unsymmetric deformation which can be attributed to the inclined orientation of the collagen fibers as at $\theta = 30°$ the plate deflects more in lower right corner, whereas for $\theta = -30°$ there is an increased deflection at upper right corner.

Chapter 5 Finite Element analysis of realistic arterial deformation

In this chapter, a practical numerical study involving simulation of a quarter piece of artery under pressure expansion is performed. The reason of presenting this numerical study is twofold 1) To simulate a realistic problem using the Holzaphel and Neohookean model. 2) To test Holzaphel model in MPI environment of Supercomputig Center Alya. For this purpose the numerical study is conducted using single and multiple processors to gauge the performance of implemented subroutines in Alya.

5.1 Pressure expansion of an artery

In this study a quarter model of artery under pressure expansion is presented. The artery model consist of an annulus with inner radius of $r_i = 6cm$ and external radius of $r_0 = 9cm$. The length of artery in z direction is 3cm and is constrained with both ends in z direction. The schematic of artery mode illustrating the geometric dimensions, boundary conditions and pressure loading is shown in figure 5.1(A).

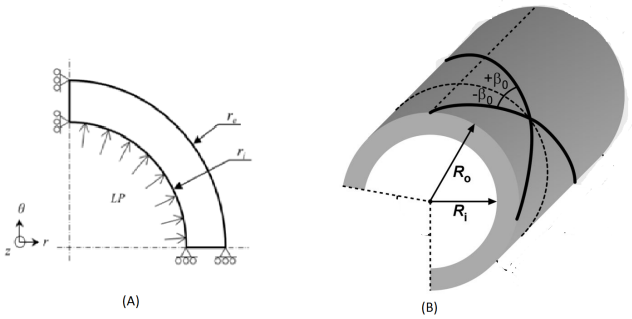

Figure 5.1: (A) Schematic illustrating the geometry, lines of symmetry and boundary conditions of the artery model (B) illustration of fibre orientations

Two set of fibres are placed at $\mp\beta_0 = 50°$ with respect to the circumferential direction in local rz plane (as shown in figure 5.1(B)). In Alya the fibre vectors can only be defined on global Cartesian coordinate axis , therefore an algorithm is developed which for every nodal point and given β_0 evaluates the fiber orientations global coordinate axis.
A mesh sensitivity study verifies that the model shows convergence in displacement for a mesh of hexahedral elements with 1044 elements.

Moreover the analysis is carried out using Holzpahel model and the Neohookean model to highlight the difference between employing anisotropic and isotropic models for arterial studies.

Material parameter	**Material parameter**
K_1	1e7(Ba)
K_2	2
μ_0	3e5(Ba)
κ	1e7(Ba)

Table 5.1: Holzaphel and Neohookean material parameters employed in the artery model

5.2 Results

The obtained displacement and von misses stress in the study is show in figure 5.2. It can be seen that the displacement and Von Mises stresses are unsymmetrical along the circumference. This is due to the two set of fibre, which make stresses distribute uneven with respect to the circumferential direction. Moreover it can be seem in figure 5.3 that Neohookean model gives symmetrical displacement field with respect to the circumferential direction.

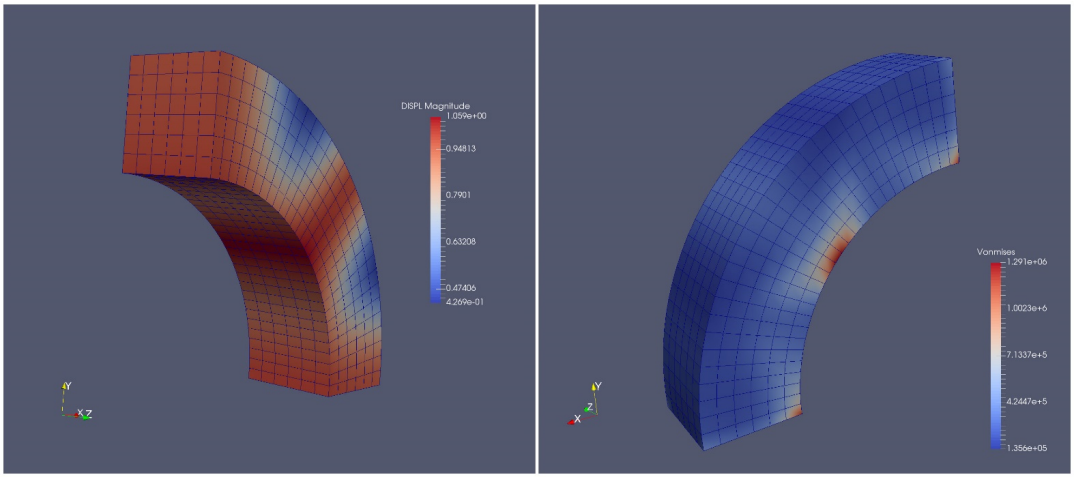

Figure 5.2: Holzaphel model: (Left) Displacement field (Right) Stress Von Mises field obtained from artery subjected to uniform pressure loading

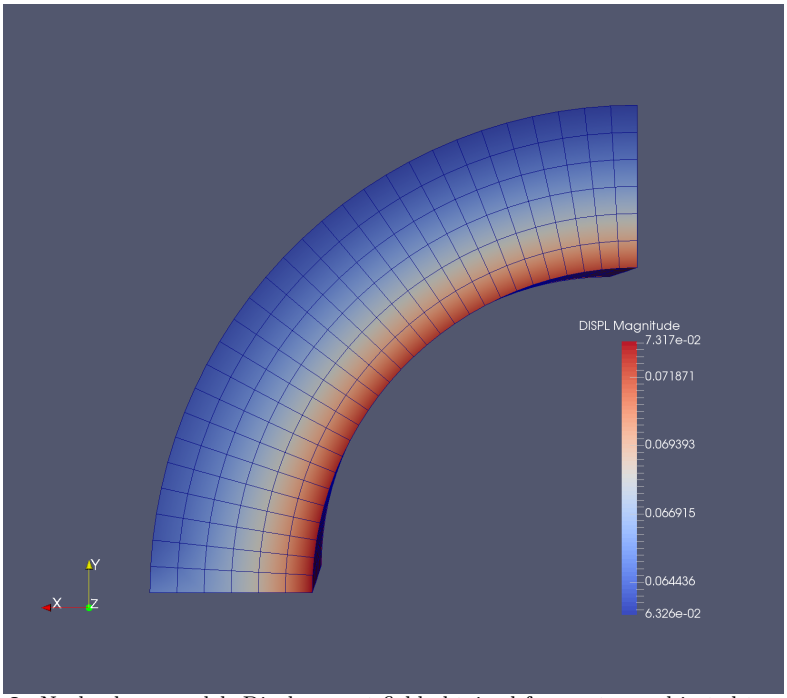

Figure 5.3: Neohookean model: Displacement field obtained from artery subjected to uniform pressure loading

5.3 Parallel Efficiency

One of the key feature of our computational model is its parallel efficiency. This serves to solve very large problems (for example simulation with very refined mesh etc.) in a reasonably wall clock time or a simple problem very quickly. In this numerical study a refined artery model is run using up to 100 processors. The parallel efficiency is gauged using scalability. Strong ideal scalability means how problem of a definite size is solved faster linearly with increase in the number of processors. This scalability measure is linear when the speedup increases linearly with the number of processors involved. To measure it, this artery simulation with refined mesh of 727,000 elements is used as bench mark and is ran with 1 , 33, 66 and 100 processors, taking the smallest one as the reference value. The tests were carried out in Marenostrum supercomputer consisting of 10 240 IBM PPC processors.
Figure 5.4 shows the result of the scalability test where speedup ratios is plotted on y-axis and no. of processors are plotted on x-axis. The speedup of a problem of size x with n processors is the execution time on one processor divided by the time on n processors.

Figure 5.4 shows that with increasing processors the speed up ratio also increases, which proves the yields good scalability with multiple processors.

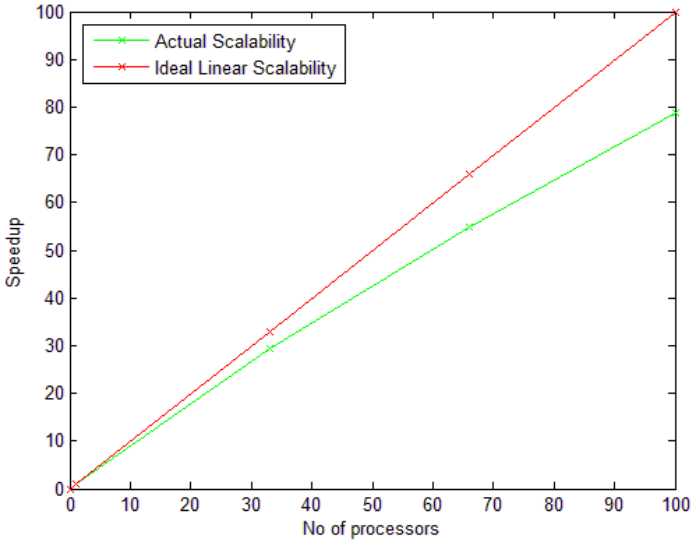

Figure 5.4: Strong scalability test

Chapter 6 Discussion and Future research

In this final chapter, the conclusion, learning outcomes and future avenues of research as a continuation of this project are outlined.

6.1 Conclusion and Discussion

In this study constitutive material models are implemented in Alya with an aim to develop the Solidz module of Alya for study of arterial mechanics. For this purpose two of the most widely used isotropic hyperelastic material models, namely Neohookean and Mooney Rivlin models are implemented in Alya. Whereas the most popular anisotropic hyperelastic material model namely, Holzaphel model for arterial walls is implemented for anisotropic arteries consisting of collagen fibre. The implemented material models are verified in one element and multi element numerical studies against literature and other finite element solvers.

1. For Neohookean and Mooney Rivlin material model the conclusions are as following:

   One element numerical studies involving finite deformations caused by uniaxial, biaxial and multiaxial loading conditions are performed. Results when compared with similar studies performed in Code Asters show nearly 0% relative error which verifies the accuracy of implementation. In numerical experiment involving cantilever beam under tip loading, the Neohookean model and Mooney Rivlin model are compared and it has been established that Monney Rivlin model is capable to capture more nonlinearity in material response , both model are robust and give almost quadratic convergence.

2. For Holzaphel material model the conclusions are as follow.

   One element numerical studies involving shear and uniaxial test exactly replicated the results presented by Nolan et al [5] which establishes the accuracy of the implementation of Holzaphel model in Alya. Moreover the tensile beam study highlights the influence of fiber orientations on the deformation and verifies both quantitatively and qualitatively the Holzpahel model implementation for multi-element geometries.

3. For the arterial model study that conclusions are following.

   The implemented material models are very robust for realistic biomedical studies . The presence of collagen fiber set orientations yield unsymmetric deformation in artery whereas for Neohookean model due to its isotropic nature, symmetric deformation is obtained. Moreover scalability analysis with this numerical study as benchmark shows parallel efficiency of the Holzaphel material model.

6.2 Future Research

There are two possible avenues of future research as a continuation of this project.

1. Improving the anisotropic constitutive model of arterial walls by incorporating plasticity and damage-softening models in Alya. An excellent anisotropic plasticity model and damage softening model has been proposed by Holzaphel et al [9] and Aelxander et al [8] respectively which should be incorporated in framework on Alya.

2. Use the implemented material models in other biomedical applications such as fluid-solid interaction analysis of blood flow in arteries etc.

6.3 Outcomes

The outcome of this study are categorized into the following.

6.3.1 Outcomes for Barcelona Supercomputing Centre

This project has contributed in advancing the in house multiphysics solver Alya for studies related to arterial mechanics. This material models implemented in this study will be used by the research group working on fluid-structure modelling of aorta.

6.3.2 Personal learning outcome

For myself this was a tremendous learning experience in following ways.

1. Learn the following concepts as part of the project: nonlinear finite element analysis, constitutive modelling, arterial mechanics and parallel computing running simulations on supercomputer.

2. Had exposure working with following solvers and tools.

    - Alya
    - Abaqus
    - Code Aster
    - Fortran 90
    - Operating system Linux
    - GNU Plot
    - Paraview

3. Last but not least, learnt about a scientific methodology of investigating and approaching a research question and how to work out a complex problem by breaking it down into subcomponents and solving each one of them to solve the bigger problem.

4. Learnt to work independently as well as in close collaboration with colleagues in a research oriented environment.

Chapter 7 References

1. Humphrey, J. D. (2003). *Review Paper: Continuum biomechanics of soft biological tissues.*In Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences (Vol. 459, No. 2029, pp. 3-46). The Royal Society.

2. Ph.D. dissertation:*Numerical modelling of the growth and remodelling phenomena in biological tissues.* Research (PDF Available).February 2016. DOI: 10.13140/RG.2.1.4485.8648

3. Duong, and Itskov (2014). *Hyperelastic modeling and soft-tissue growth integrated with the smoothed finite element method—SFEM.* (Doctoral dissertation, Dissertation, RWTH Aachen University).

4. Holzapfel, G. A., and Gasser, T. C. (2001). *A viscoelastic model for fiber-reinforced composites at finite strains: Continuum basis, computational aspects and applications.* Computer methods in applied mechanics and engineering, 190(34), 4379-4403.

5. Nolan, D. R., Gower, A. L., Destrade, M., Ogden, R. W., and McGarry, J. P. (2014). *A robust anisotropic hyperelastic formulation for the modelling of soft tissue.* Journal of the mechanical behavior of biomedical materials, 39, 48-60.

6. Abaqus/Standard, Abaqus/Explicit, Abaqus/CAE . http://dsk.ippt.pan.pl/docs/abaqus/v6.13/books/usb/default.htm?startat=pt05ch22s05abm09.html.usb-mat-canisohyperelastic

7. Ehret, A. E., and Itskov, M. (2009). *Modeling of anisotropic softening phenomena: application to soft biological tissues.* International Journal of Plasticity, 25(5), 901-919.

8. Gasser, T. C. and Holzapfel, G. A. (2002).*A rate-independent elastoplastic constitutive model for biological fiber-reinforced composites at finite strains: continuum basis, algorithmic formulation and finite element implementation.* Computational Mechanics, 29(4-5), 340-360.

9. Casoni, E., Jérusalem, A., Samaniego, C., Eguzkitza, B., Lafortune, P., Tjahjanto, D. D., ... and Vázquez, M. (2015).*Alya: computational solid mechanics for supercomputers*. Archives of Computational Methods in Engineering, 22(4), 557-576.

10. Denny Tjahjanto, Antoine Jerusalem, Guillaume Houzeaux, Mariano Vazquez. *Validation of Alya/implicit: Longitudinal deformation of 3-D elastic beam*

11. Holzapfel, G. A., and Ogden, R. W. (2009). *Constitutive modelling of passive myocardium: a structurally based framework for material characterization*. Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 367(1902), 3445-3475.

12. Denny Tjahjanto, Antoine Jerusalem, Guillaume Houzaeux, Mariano Vazquez, *Algorithm For Alya/Implicit*, IMDEA, BSC

Chapter 8 Appendix

8.1 Appendix A:Mooney-Rivlin Model

Given below is the fortran 90 subroutine of Mooney-Rivlin model implemented in Alya-Solidz

```
subroutine sld_stress_model_102(pgaus,pmate,gpgdi,gpstr,gpdet,flagt,gpdds,gpmof)
  !------------------------------------------------------------------------
  !****f* Solidz/sld_elmcla
  ! NAME
  !    sld_stress_model_102
  ! DESCRIPTION
  !    Neo-Hookean material, Ansys' formulation
  !    Compute second Piola-Kirchoff stress tensor S_{IJ}
  !
  !    GPGDI ... Deformation tensor ..................... F = grad(phi)
  !    GPCAU ... Right Cauchy-Green deformation tensor ... C = F^t x F
  !    GPSTR ... 2nd P-K Stress tensor ..................... S
  !    GPPIO ... 1st Piola-Kirchhoff stress tensor ....... P = F.S
  !    GPDDS ... Stress tangent moduli ................... dS/dE
  !    FLAGT ... Flag to activate GPDDS (when implicit)
  ! USES
  ! USED BY
  !    sld_elmope
  !***
  !------------------------------------------------------------------------
  use def_kintyp, only       :  ip,rp
  use def_domain, only       :  ndime
  use def_solidz, only       :  lawco_sld,parco_sld,kfl_serei_sld,densi_sld,velas_sld
  use def_master, only       :  ittim
  implicit none
  integer(ip), intent(in)    :: pgaus,pmate,flagt
  real(rp)                   :: gpcau(ndime,ndime,pgaus),CC(ndime,ndime,pgaus),gpmof(pgaus),C01,i2,tracecc,&
                                I_4,X,term1,term2,bulk_1,bulk_2,bulk_3,shear_1,shear_2,shear_3,shear_4,shear
  real(rp),    intent(in)    :: gpgdi(ndime,ndime,pgaus),gpdet(pgaus)  !NOTE: F NOT USED
  real(rp)                   :: gpgdi2(ndime,ndime,pgaus)
  real(rp),    intent(out)   :: gpstr(ndime,ndime,pgaus)
  real(rp),    intent(out)   :: gpdds(ndime,ndime,ndime,ndime,pgaus)
  real(rp)                   :: gpcin(ndime,ndime),tkron(ndime,ndime)
  integer(ip)                :: igaus,i,j,k,l,switch,idime,jdime,kdime,ldime,ktmo,ltmo,test(ndime,ndime)
  real(rp)                   :: K0,mu0,tracec,traceb,bidon,detf0,hglas,hfac1,hfac2,logj

  real(rp)        :: gpfin(ndime,ndime,pgaus)
  real(rp)        :: cauch(ndime,ndime,pgaus)

  real(rp)        :: gpstr_tmo(ndime,ndime),gpcau_tmo(ndime,ndime)

  integer(ip), parameter     :: tmeth = 0_ip ! tmeth = 0 => approximation by Belytschko model
                                             ! tmeth = 1 => numerical method using finite diff
  real(rp), parameter        :: ptmo  = 1.0e-8_rp
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Extracting material Data
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

mu0  = parco_sld(1,pmate)
K0   = parco_sld(2,pmate)
C01  = parco_sld(3,pmate)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Kronecker delta
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

tkron = 0.0_rp
do i = 1, ndime
    tkron(i,i) = 1.0_rp
end do

gpstr = 0.0_rp
gpdds = 0.0_rp
test  =  0.0_rp
switch = 1_ip



!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Calculating first and second invariants
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

   do igaus=1,pgaus
      CC(:,:,igaus)= matmul(gpcau(:,:,igaus),gpcau(:,:,igaus))
   end do


   do igaus=1,pgaus

      gpcin = tkron

      call invmtx(gpcau(:,:,igaus),gpcin,bidon,ndime) !C^(-1)

      tracec = 0.0_rp
      do i=1,ndime
         tracec = tracec + gpcau(i,i,igaus)
      end do


      i2 =  0.0_rp
      tracecc  = 0.0_rp
```

```
         tracecc = tracecc + CC(i,i,igaus)
      end do
      i2 =  0.5_rp* (tracec**2-tracecc)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! GPSTR: 2nd Piola--Kirchhoff stress tensor
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

      gpstr(:,:,igaus)= &
            ((-mu0/3.0_rp)*gpdet(igaus)**(-2.0_rp/3.0_rp)*tracec + &
            gpdet(igaus)*K0*(gpdet(igaus)-1.0_rp))*gpcin + &
            mu0*gpdet(igaus)**(-2.0_rp/3.0_rp)*tkron + &
            2*C01*(gpdet(igaus)**(-4.0_rp/3.0_rp))*(tracec*tkron-(gpcau(:,:,igaus))-(2.0_rp/3.0_rp)*gpcin*i2)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! ! Compute the tangent moduli if required (i.e., implicit scheme)
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
      if (flagt == 1_ip) then


         I_4  = 0.0_rp
         X    = 0.0_rp
         term1 = 0.0_rp
         term2 = 0.0_rp
         bulk_1 = 0.0_rp
         bulk_2 = 0.0_rp
         do i=1,ndime; do j=1,ndime; do k=1,ndime; do l=1,ndime

            X    = 0.5_rp*(gpcin(i,k)*gpcin(j,l) + gpcin(j,k)*gpcin(i,l))
            I_4  = 0.5_rp*(tkron(i,k)*tkron(j,l) + tkron(i,l)*tkron(j,k))

                     bulk_1 = K0*gpdet(igaus)*(gpdet(igaus)-1.0_rp)*gpcin(i,j)*gpcin(k,l)

                     bulk_2 = -2.0_rp*K0*gpdet(igaus)*(gpdet(igaus)-1.0_rp)*X

                     bulk_3 = K0*(gpdet(igaus)**2.0_rp)*gpcin(i,j)*gpcin(k,l)


                     shear_1 = (1.0_rp/3.0_rp)*tracec*X

                     shear_2 = -(1.0_rp/3.0_rp)*tkron(i,j)*gpcin(k,l)

                     shear_3 = -(1.0_rp/3.0_rp)*tkron(k,l)*gpcin(i,j)

                     shear_4 =  (1.0_rp/9.0_rp)*tracec*gpcin(i,j)*gpcin(k,l)

                     shear = shear_1+shear_2+shear_3+shear_4


                  term2  = gpdet(igaus)**(-4.0_rp/3.0_rp)*(tkron(i,j)*tkron(k,l)-&
                           I_4 + (2.0_rp/3.0_rp)*i2*X -&
                          (2.0_rp/3.0_rp)*gpcin(i,j)*(tracec*tkron(k,l)-gpcau(k,l,igaus)))

                  gpdds(i,j,k,l,igaus)= gpdds(i,j,k,l,igaus) + &
                                                bulk_1+bulk_2+bulk_3+ &
                                                2.0_rp*mu0*gpdet(igaus)**(-2.0_rp/3.0_rp)*shear + 4*C01*(term1+term2

         enddo; enddo; enddo; enddo

      endif


   end do !ifgaus

end subroutine sld_stress_model_102
```

8.2 Appendix B: Neohookean Model

Given below is the fortran 90 subroutine of Neohookean model model implemented in Alya-Solidz

```
subroutine sld_stress_model_104pgaus,pmate,gpgdi,gpstr,gpdet,flagt,gpdds,gpmof)
  !------------------------------------------------------------------------
  !****f* Solidz/sld_elmcla
  ! NAME
  !    sld_stress_model_102
  ! DESCRIPTION
  !    Neo-Hookean material
  !    Compute second Piola-Kirchoff stress tensor S_{IJ}
  !
  !    GPGDI ... Deformation tensor ..................... F = grad(phi)
  !    GPCAU ... Right Cauchy-Green deformation tensor ... C = F^t x F
  !    GPSTR ... 2nd P-K Stress tensor ......................... S
  !    GPPIO ... 1st Piola-Kirchhoff stress tensor ....... P = F.S
  !    GPDDS ... Stress tangent moduli ................... dS/dE
  !    FLAGT ... Flag to activate GPDDS (when implicit)
  ! USES
  ! USED BY
  !    sld_elmope
  !***
  !------------------------------------------------------------------------
  use def_kintyp, only       :  ip,rp
  use def_domain, only       :  ndime
  use def_solidz, only       :  lawco_sld,parco_sld,kfl_serei_sld,densi_sld,velas_sld
  use def_master, only       :  ittim
  implicit none
  integer(ip), intent(in)    :: pgaus,pmate,flagt
  real(rp)                   :: gpcau(ndime,ndime,pgaus),CC(ndime,ndime,pgaus),gpmof(pgaus),C01,i2,tracecc,&
                                I_4,X,term1,term2,bulk_1,bulk_2,bulk_3,shear_1,shear_2,shear_3,shear_4,shear
  real(rp),    intent(in)    :: gpgdi(ndime,ndime,pgaus),gpdet(pgaus)  !NOTE: F NOT USED
  real(rp)                   :: gpgdi2(ndime,ndime,pgaus)
  real(rp),    intent(out)   :: gpstr(ndime,ndime,pgaus)
  real(rp),    intent(out)   :: gpdds(ndime,ndime,ndime,ndime,pgaus)
  real(rp)                   :: gpcin(ndime,ndime),tkron(ndime,ndime)
  integer(ip)                :: igaus,i,j,k,l,switch,idime,jdime,kdime,ldime,ktmo,ltmo,test(ndime,ndime)
  real(rp)                   :: K0,mu0,tracec,traceb,bidon,detf0,hglas,hfac1,hfac2,logj

  real(rp)        :: gpfin(ndime,ndime,pgaus)
  real(rp)        :: cauch(ndime,ndime,pgaus)

  real(rp)        :: gpstr_tmo(ndime,ndime),gpcau_tmo(ndime,ndime)


  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
  ! Extracting material Data
  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Extracting material Data
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

mu0  = parco_sld(1,pmate)
K0   = parco_sld(2,pmate)
C01  = parco_sld(3,pmate)
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Kronecker delta
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

tkron = 0.0_rp
do i = 1, ndime
    tkron(i,i) = 1.0_rp
end do

gpstr = 0.0_rp
gpdds = 0.0_rp




!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Calculating first  invariants
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

   do igaus=1,pgaus
      CC(:,:,igaus)= matmul(gpcau(:,:,igaus),gpcau(:,:,igaus))
   end do


   do igaus=1,pgaus

      gpcin = tkron

      call invmtx(gpcau(:,:,igaus),gpcin,bidon,ndime) !C^(-1)

      tracec = 0.0_rp
      do i=1,ndime
         tracec = tracec + gpcau(i,i,igaus)
      end do


!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! GPSTR: 2nd Piola--Kirchhoff stress tensor
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! GPSTR: 2nd Piola--Kirchhoff stress tensor
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

      gpstr(:,:,igaus)= &
            ((-mu0/3.0_rp)*gpdet(igaus)**(-2.0_rp/3.0_rp)*tracec + &
            gpdet(igaus)*K0*(gpdet(igaus)-1.0_rp))*gpcin + &
            mu0*gpdet(igaus)**(-2.0_rp/3.0_rp)*tkron


!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! ! Compute the tangent moduli if required (i.e., implicit scheme)
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
      if (flagt == 1_ip) then


         I_4  = 0.0_rp
         X    = 0.0_rp
         term1 = 0.0_rp
         term2 = 0.0_rp
         bulk_1 = 0.0_rp
         bulk_2 = 0.0_rp
         do i=1,ndime; do j=1,ndime; do k=1,ndime; do l=1,ndime

            X    = 0.5_rp*(gpcin(i,k)*gpcin(j,l) + gpcin(j,k)*gpcin(i,l))
            I_4  = 0.5_rp*(tkron(i,k)*tkron(j,l) + tkron(i,l)*tkron(j,k))

                     bulk_1 = K0*gpdet(igaus)*(gpdet(igaus)-1.0_rp)*gpcin(i,j)*gpcin(k,l)

                     bulk_2 = -2.0_rp*K0*gpdet(igaus)*(gpdet(igaus)-1.0_rp)*X

                     bulk_3 = K0*(gpdet(igaus)**2.0_rp)*gpcin(i,j)*gpcin(k,l)


                     shear_1 = (1.0_rp/3.0_rp)*tracec*X

                     shear_2 = -(1.0_rp/3.0_rp)*tkron(i,j)*gpcin(k,l)

                     shear_3 = -(1.0_rp/3.0_rp)*tkron(k,l)*gpcin(i,j)

                     shear_4 =  (1.0_rp/9.0_rp)*tracec*gpcin(i,j)*gpcin(k,l)

                     shear = shear_1+shear_2+shear_3+shear_4







                  gpdds(i,j,k,l,igaus)= gpdds(i,j,k,l,igaus) + &
                                                bulk_1+bulk_2+bulk_3+ &
                                                2.0_rp*mu0*gpdet(igaus)**(-2.0_rp/3.0_rp)*shear

         enddo; enddo; enddo; enddo

      endif

   end do !ifgaus

end subroutine sld_stress_model_104
```

8.3 Appendix C: Derivation of anisotropic part of tangent constitutive matrix $\mathbb{C}_f$ of Holzaphel model

(Note that the derivation is performed for just one set of fibres, for the second set of fibre the derivation is similar. From here onward $\boldsymbol{a}_{01}, \boldsymbol{a}_{02}$ will be referred as $\boldsymbol{a}_{0i}$)

The anisotropic strain energy function for the Holzaphel model is presented as follow:

$$\boldsymbol{\Psi}_{aniso}(\bar{I}_i) = \frac{a_f}{2b_f} \sum_{i=4,6} [exp[b_f(\bar{I}_i - 1)^2] - 1] \tag{c.1}$$

The corresponding Second Piola Kricoff stress which is the derivative of equation c.1 with respect to Right Green Cauchy tensor $\mathbf{C}$ is as follow.

$$\bar{\boldsymbol{S}}_{\boldsymbol{f}} = 2\frac{\boldsymbol{\Psi}_{aniso}(\bar{I}_i)}{\partial \mathbf{C}} = 2\psi_{aniso}\frac{\partial \bar{I}_i}{\partial \mathbf{C}} \tag{c.2}$$

$$\bar{\mathbf{S}_f} = 2J^{-\frac{2}{3}}\psi^{\alpha} Dev(\boldsymbol{a}_0 \otimes \boldsymbol{a}_0)$$

Where

$$\frac{\partial \bar{I}_i}{\partial \mathbf{C}} = \frac{\partial \bar{I}_i}{\partial \bar{\mathbf{C}}} : \frac{\partial \bar{\mathbf{C}}}{\partial \mathbf{C}} = J^{-\frac{2}{3}}(\boldsymbol{a}_{0i} \otimes \boldsymbol{a}_{0i} - \frac{1}{3}\bar{I}_i \bar{\mathbf{C}}^{-1}) \tag{c.3}$$

$$= J^{-\frac{2}{3}} Dev(\boldsymbol{a}_{0i} \otimes \boldsymbol{a}_{0i})$$

$$Dev(\bullet) = (\bullet) - (\frac{1}{3})[(\bullet) : \bar{\mathbf{C}}]\bar{\mathbf{C}}^{-1}$$

Anisotropic part of tangent constitutive matrix $\bar{\mathbb{C}}_{aniso}$ is the derivative of equation c.2 , shown as follow:

$$\bar{\mathbb{C}}_{aniso} = 2\frac{\partial \bar{\mathbf{S}}_f}{\partial \mathbf{C}} \tag{c.4}$$

To calculate anisotropic part of the tangent constitutive matrix $\bar{\mathbb{C}}_{aniso}$, following relations are required , which can be calculated with principles of tensor calculus.

$$\frac{\partial Dev(\boldsymbol{a}_{0i} \otimes \boldsymbol{a}_{0i})}{\partial \mathbf{C}} = -\frac{1}{3}[\mathbf{C}^{-1} \otimes Dev(\boldsymbol{a}_{0i} \otimes \boldsymbol{a}_{0i}) \tag{c.5}$$

$$-J^{\frac{2}{3}}\bar{I}_i(\mathbf{C}^{-1} \odot \mathbf{C}^{-1} - \frac{1}{3}\mathbf{C}^{-1} \otimes \mathbf{C}^{-1})]$$

$$-(C^{-1}\odot C^{-1})_{ijkl}=-\frac{1}{2}(C^{-1}_{ik}C^{-1}_{jl}+C^{-1}_{jk}C^{-1}_{il})=\frac{\partial C^{-1}_{ij}}{\partial C^{-1}_{kl}} \tag{c.6}$$

$$\frac{\partial \mathbf{J}}{\partial \mathbf{C}}=\frac{1}{2}\mathbf{J}\mathbf{C}^{-1},\frac{\partial \bar{\mathbf{J}}}{\partial \mathbf{C}}=\mathbf{J}^{-\frac{2}{3}}(\mathbb{I}-\frac{1}{3}\bar{\mathbf{C}}\otimes\bar{\mathbf{C}}^{-1}) \tag{c.7}$$

$$(\mathbb{I})_{ijkl}=\frac{(\delta_{ik}\delta_{jl}+\delta_{il}\delta_{jk})}{2} \tag{c.8}$$

Now using equation (c.2) and the chain rule the anisotropic elasticity $\bar{\mathbb{C}_{aniso}}$, follows from the definition (c.4) as

$$\bar{\mathbb{C}}_{aniso}=4[\psi^{\alpha}Dev(\boldsymbol{a}_{0i}\otimes\boldsymbol{a}_{0i})\otimes\frac{\partial J^{-\frac{2}{3}}}{\partial \mathbf{C}}+J^{-\frac{2}{3}} \tag{c.9}$$

$$Dev(\boldsymbol{a}_{0i}\otimes\boldsymbol{a}_{0i})\otimes\frac{\partial \psi^{\alpha}}{\partial \mathbf{C}}+J^{-\frac{2}{3}}\psi^{\alpha}\frac{Dev(\boldsymbol{a}_{0i}\otimes\boldsymbol{a}_{0i})}{\partial \mathbf{C}}] \tag{c.10}$$

Now the equations (c.5), (c.6) and (c.7) are plugged in the equation (c.8) and following formulation of $\bar{\mathbb{C}}_{aniso}$ is obtained.

$$\bar{\mathbb{C}}_{aniso}=4\{\frac{1}{3}\psi^{\alpha}[\bar{I}_i(\mathbf{C}^{-1}\odot\mathbf{C}^{-1}-\frac{1}{3}\mathbf{C}^{-1}\otimes\mathbf{C}^{-1})] \tag{c.11}$$

$$-J^{-\frac{2}{3}}Dev(\boldsymbol{a}_{0i}\otimes\boldsymbol{a}_{0i})\otimes\mathbf{C}^{-1}$$

$$-J^{-\frac{2}{3}}\mathbf{C}^{-1}\otimes Dev(\boldsymbol{a}_{0i}\otimes\boldsymbol{a}_{0i})]$$

$$+J^{-\frac{4}{3}}\psi^{\alpha\alpha}Dev(\boldsymbol{a}_{0i}\otimes\boldsymbol{a}_{0i})\otimes Dev(\boldsymbol{a}_{0i}\otimes\boldsymbol{a}_{0i})\}$$

Where

$$\psi^{\alpha\alpha}=d^2\frac{\psi_{aniso}}{d\quad\bar{I}_i\quad d\quad\bar{I}_i}=\mathbf{a}_f[1+2b_f(\bar{I}_i-1)^2]exp[(\bar{I}_i-1)^2]$$

8.4 Appendix D: Holzaphel Model for arterial walls

Given below is the fortran 90 subroutine of Neohookean model model implemented in Alya-Solidz

```
subroutine sld_stress_model_134(&
     pgaus,pmate,gpcau,gpgdi,gpigd,gpstr,gpdet,&
     gptlo,nfibe,ielem,elcod,pnode,lnods,gpsha,flagt,gpdds,gpmof)
  !------------------------------------------------------------------------
  !****f* Solidz/sld_stress_model_134
  ! NAME
  !    sld__stress_model_134
  ! DESCRIPTION
  !    Law of Holzapfel et Ogden (2009)
  !
  !    GPGDI ... Deformation tensor ..................... F = grad(u) + I
  !    GPIGD ... Inverse of Deformation tensor ...........F^(-1)
  !    GPCAU ... Right Cauchy-Green deformation tensor ... C = F^t x F
  !    GPCAL ... Left Cauchy-Green deformation tensor ...  b = F x F^T
  !    GPSTR ... 2nd P-K Stress tensor ...................S
  !    CASTR ... Cauchy Stress tensor ..................... sigma
  !    GPPIO ... 1st Piola-Kirchhoff stress tensor ....... P = F.S
  !    GPENE ... Stored energy function .................. W
  !    GPLEP ... Log strain in the {f s n} system
  !    GPDDS ... Tangent moduli in terms of 2nd P-K stress ....... dS/dE
  !
  !    Special postproc values for this material:
  !    OUTPUT_&_POST-PROCESS
  !    POSTPROCESS LOCEPSILON,   => Log strain in the fibers CS - ln(lambda)
  !    END_OUTPUT_&_POST_PROCESS
  !
  ! USES
  ! USED BY
  !    sld_elmope
  !***
  !----------------------------------------- ---------------------------
  use def_kintyp, only       :  ip,rp
  use def_domain, only       :  ndime,mnode,lmate,npoin,coord
  use def_solidz
  use def_master, only       :  &
       cutim,ittim,postp,ittyp,ITASK_ENDRUN,fiber,gpfib,kfl_ephys,vconc,coupling,kfl_paral,fisoc


  implicit none
  integer(ip), intent(in)    :: pgaus,pmate,pnode,lnods(pnode),flagt
  real(rp),    intent(in)    :: gpcau(ndime,ndime,pgaus),gpdet(pgaus),gpgdi(ndime,ndime,pgaus),gpmof(pga
  real(rp),    intent(in)    :: gpigd(ndime,ndime,pgaus),gpsha(pnode,pgaus)
  real(rp),    intent(out)   :: gpstr(ndime,ndime,pgaus)
  real(rp),    intent(out)   :: gpdds(ndime,ndime,ndime,ndime,pgaus)
  real(rp)                   :: detf0,nfibe_length,tactf,nfibe_2(ndime,pgaus),nfibe(ndime,pgaus)

  integer(ip)                :: igaus,idime,jdime,kdime,ldime,ielem,kangl,inode,ipoin,ikact
  real(rp)                   :: gpcal(ndime,ndime),castr(ndime,ndime),castv(ndime,ndime),sleci(3,3,3)
```

```
  real(rp)                  :: nfibe0(ndime,pgaus),nfibe0_2(ndime,pgaus),norma0(ndime,pgaus),nshet0(ndime,pgaus) !f0, s0, n0 (ref. config)
  real(rp)                  ::                     norma(ndime,pgaus) ,nshet(ndime,pgaus)  !f,s,n (fF as define in paper) (nfibe is "output")
  real(rp)                  :: nfibt(ndime,pgaus) ,normt(ndime,pgaus) ,nshtt(ndime,pgaus)  !f,s,n (updated-Unit)

  real(rp)                  :: a,b,af,afm,bf,as,bs,afs,bfs,afc,flag_2

  real(rp)                  :: i1,i4f,i4s,i4n,i8fs,i4ff,i4sf,i6f
  real(rp)                  :: fporf_t(ndime,ndime),spors(ndime,ndime)
  real(rp)                  :: fpors(ndime,ndime),sporf(ndime,ndime)
  real(rp)                  :: term1,term2,term3,term4,term1_2,term2_2,term3_2,term4_2,termK
  real(rp)                  :: gpcai(ndime,ndime),bidon,bidon_2,Kct,dwvdc(ndime,ndime),trace

  real(rp)                  :: lamda(3),Cal50,n,norme1,norme2
  real(rp)                  :: scasta,scastr,scastv,sdwvdc,bidon2,dummr,ovlam(3)

  real(rp)                  :: castr_active(ndime,ndime),gpstr_active(ndime,ndime),gpgre(ndime,ndime)

  real(rp)                  :: xcord,ycord,zcord,vuni1(3),vuni2(3),&
       elfib(3,mnode),elfib_2(3,mnode),elfis(3,mnode),elfin(3,mnode),elcac(mnode),gpcac

  real(rp)                  :: lenght_pnt,angle1,omega,a_epi,b_epi,yp_epi,zp_epi,lenght_end,l_star
  real(rp)                  :: rotma(3,3),lenght_epi,yp_end,zp_end,a_end,b_end,nore1,scalf

  real(rp),    intent(in)   :: gptlo(pgaus)
  real(rp),    intent(in)   :: elcod(ndime,mnode)

  real(rp)                  :: tkron(3,3)

real(KIND=rp)                  :: fporf(ndime,ndime),fporf_0(ndime,ndime),fporf_2(ndime,ndime),fporf_0_2(ndime,ndime)

real(KIND=rp)                  :: term5,term6,term7,term11,term22,&
                                  term33,term44,term55,term66,term77,inter1,traceb_bar,trace_fporf,trace_fporf_2

real(KIND=rp)                  :: gpcau_bar(ndime,ndime),a1(ndime,ndime),energy_al,energy_al_2,energy_al_al,energy_al_al_2,&
                                  det,bulk_1,bulk_2,bulk_3, &
                                  shear_1,shear_2,shear_3,shear_4,shear,C4,Dev(ndime,ndime),Dev_2(ndime,ndime),invC(ndime,ndime) &
                                  ,invC_bar(ndime,ndime)

real(KIND=rp)                  :: devb(ndime,ndime),gpcal_bar(ndime,ndime),identity_second(ndime,ndime),&
                                  identity_fourth(ndime,ndime,ndime,ndime),dev_fporf(ndime,ndime),dev_fporf_2(ndime,ndime),&
                                   cauchy_gs(ndime,ndime)&
                                  , cauchy_f(ndime,ndime) , cauchy_f_2(ndime,ndime) , cauchy_l(ndime,ndime) , cauchy_fg(ndime,ndime)
```

```
   ! * * * * * ********************
   ! Extracting material parameters
   ! * * * * * ********************

flag_2 = 1_ip
Kct = parco_sld( 2,pmate)
a   = parco_sld( 3,pmate)
b   = parco_sld( 4,pmate)
af  = parco_sld( 5,pmate)
bf  = parco_sld( 6,pmate)
as  = parco_sld( 7,pmate)
bs  = parco_sld( 8,pmate)
afs = parco_sld( 9,pmate)
bfs = parco_sld(10,pmate)
scalf = parco_sld(11,pmate)




gpcai = 0.0_rp
gpstr = 0.0_rp
gpdds = 0.0_rp

nfibt = 0.0_rp
nshtt = 0.0_rp
normt = 0.0_rp

nfibe = 0.0_rp
nfibe_2 = 0.0_rp
nshet = 0.0_rp
norma = 0.0_rp

nfibe0 = 0.0_rp
nfibe0_2 = 0.0_rp
norma0 = 0.0_rp
nshet0 = 0.0_rp
elfib = 0.0_rp
elfib_2 = 0.0_rp

tkron = 0.0_rp

do idime=1,ndime
   tkron(idime,idime) = 1.0_rp
end do
```

```
  do idime=1,ndime
     tkron(idime,idime) = 1.0_rp
  end do

    ! * * * * * ********************
    ! Loop for gauss point
    ! * * * * * ********************



  do igaus = 1,pgaus


          gpcai = 0.0_rp


     !Calcul of b = F F^T
     gpcal = 0.0_rp
     do idime = 1,ndime
        do jdime = 1,ndime
           do kdime = 1,ndime
              gpcal(idime,jdime) = &
                   gpcal(idime,jdime) + gpgdi(idime,kdime,igaus) * gpgdi(jdime,kdime,igau
           end do
        end do
     end do
! print *, pgaus,pnode


     ! * * * * * * *
     ! fiber orientations
     ! * * * * * * *

        !
        ! Fibers interpolated on Gauss point
        !
!        nfibe0(1,igaus) = 0.0_rp
!        nfibe0(2,igaus) = 0.0_rp
!        nfibe0(3,igaus) = 0.0_rp
!        nfibe0_2(1,igaus) = 0.0_rp
!        nfibe0_2(2,igaus) = 0.0_rp
!        nfibe0_2(3,igaus) = 0.0_rp
```

```
        !
        ! Fibers interpolated on Gauss point
        !
!        nfibe0(1,igaus) = 0.0_rp
!        nfibe0(2,igaus) = 0.0_rp
!        nfibe0(3,igaus) = 0.0_rp
!        nfibe0_2(1,igaus) = 0.0_rp
!        nfibe0_2(2,igaus) = 0.0_rp
!        nfibe0_2(3,igaus) = 0.0_rp

        do inode = 1,pnode
            nfibe0(1,igaus)   =  nfibe0(1,igaus) + gpsha(inode,igaus)*elfib(1,inode)
            nfibe0(2,igaus)   =  nfibe0(2,igaus) + gpsha(inode,igaus)*elfib(2,inode)
            nfibe0(3,igaus)   =  nfibe0(3,igaus) + gpsha(inode,igaus)*elfib(3,inode)
            nfibe0_2(1,igaus) =  nfibe0_2(1,igaus) + gpsha(inode,igaus)*elfib(1,inode)
            nfibe0_2(2,igaus) =  nfibe0_2(2,igaus) + gpsha(inode,igaus)*elfib(2,inode)
            nfibe0_2(3,igaus) =  nfibe0_2(3,igaus) + gpsha(inode,igaus)*(-elfib(3,inode))


        end do

    ! * * * * * ********************
    ! Normalizing fiber orientatons
    ! * * * * * ********************

        bidon=sqrt(nfibe0(1,igaus)**2.0_rp+nfibe0(2,igaus)**2.0_rp+nfibe0(3,igaus)**2.0_rp)
        bidon_2=sqrt(nfibe0_2(1,igaus)**2.0_rp+nfibe0_2(2,igaus)**2.0_rp+nfibe0_2(3,igaus)**2.0_rp)         !new cahnge

        if (bidon == 0.0_rp) bidon= 1.0_rp
        do idime=1,ndime
           nfibe0(idime,igaus)=nfibe0(idime,igaus)/bidon
           nfibe0_2(idime,igaus)=nfibe0_2(idime,igaus)/bidon_2                    ! new change
        end do



     ! * * * *
     !INVARIANTS
     ! * * * *

     !I_1=Trace(C)
     i1=0.0_rp
     do idime = 1,ndime
        i1 = i1 + gpcau(idime,idime,igaus)
     end do
```

```
    i1=0.0_rp
    do idime = 1,ndime
       i1 = i1 + gpcau(idime,idime,igaus)
    end do
    i4f = 0.0_rp
    i4s = 0.0_rp
    i4n = 0.0_rp
    i8fs = 0.0_rp
    i6f = 0.0_rp

gpcau_bar = gpdet(igaus)**(-2.0_rp/3.0_rp)*gpcau(idime,idime,igaus)

    do idime = 1,ndime
       do jdime = 1,ndime
          i4f  =  i4f + nfibe0(idime,igaus) * gpdet(igaus)**(-2.0_rp/3.0_rp)*gpcau(idime,jdime,igaus) * nfibe0(jdime,igaus) ! I_4f  = f0 * C_ij * f0 (eq. 5.1)
          i6f  =  i6f + nfibe0_2(idime,igaus) * gpdet(igaus)**(-2.0_rp/3.0_rp)*gpcau(idime,jdime,igaus) * nfibe0_2(jdime,igaus) ! I_6f  = f0_2 * C_ij * f0_2

          ! * * * * *
          !Transform f=F'*f0 ; s=F'*s0 ; n=F'*n0 (PUSH FORWARD)

          nfibe(idime,igaus)   = nfibe(idime,igaus)   +  (gpdet(igaus)**(-1.0_rp/3.0_rp))*gpgdi(idime,jdime,igaus) * nfibe0(jdime,igaus)
          nfibe_2(idime,igaus) = nfibe_2(idime,igaus) +  (gpdet(igaus)**(-1.0_rp/3.0_rp))*gpgdi(idime,jdime,igaus) * nfibe0_2(jdime,igaus)

       end do
    end do

    !
    ! f x f  (outer products [3x3])
    !
    fporf = 0.0_rp
    fporf_2 = 0.0_rp

    do idime = 1,ndime
       do jdime = 1,ndime

          fporf(idime,jdime)   = nfibe(idime,igaus)   * nfibe(jdime,igaus)
          fporf_0(idime,jdime) = nfibe0(idime,igaus)  * nfibe0(jdime,igaus)
          fporf_0_2(idime,jdime) = nfibe0_2(idime,igaus) * nfibe0_2(jdime,igaus)
          fporf_2(idime,jdime) = nfibe_2(idime,igaus) * nfibe_2(jdime,igaus)

       end do
    end do
```

```
      ! Non-collageous material
      term1 = (a/gpdet(igaus))*exp(b*(i1-3.0_rp))-(a/gpdet(igaus))

      term2= 0.0_rp
      term3= 0.0_rp
      term4= 0.0_rp



         term2 =  2.0_rp*af*(i4f-1.0_rp)*exp(bf *(i4f-1.0_rp)**2.0_rp)/gpdet(igaus)
         term2_2 =  2.0_rp*af*(i6f-1.0_rp)*exp(bf *(i6f-1.0_rp)**2.0_rp)/gpdet(igaus)

         !shear
         term4 = (afs/gpdet(igaus))*i8fs*exp(bfs*i8fs**2.0_rp)
      end if

         term3 = 2.0_rp*af*(i6f-1.0_rp)*exp(bf *(i6f-1.0_rp)**2.0_rp)      ! for second fiber


trace_fporf = 0.0_rp
trace_fporf_2 = 0.0_rp

do idime = 1,ndime
   do jdime = 1,ndime

    trace_fporf  = trace_fporf  + tkron(idime ,jdime )*fporf(jdime ,idime )
    trace_fporf_2  = trace_fporf_2  + tkron(idime ,jdime )*fporf_2(jdime ,idime )

    end do
end do
dev_fporf = fporf- (1.0_rp/3.0_rp)*trace_fporf*tkron
dev_fporf_2 = fporf_2- (1.0_rp/3.0_rp)*trace_fporf_2*tkron

    ! * * * * * ********************
    ! Calculating cauchy stress
    ! * * * * * ********************
```

```
cauchy_gs = 0.0_rp
cauchy_f  = 0.0_rp
cauchy_l  = 0.0_rp
cauchy_fg  = 0.0_rp

     do idime = 1,ndime
        do jdime = 1,ndime
cauchy_gs(idime,jdime) =  a*(gpcal(idime,jdime)-(1.0_rp/3.0_rp)*tkron(idime,jdime)*i1)*gpdet(igaus)**(-5.0_rp/3.0_rp)
! print *, idime,jdime
! print *, cauchy_gs(idime,jdime)
! print *,  a*(gpcal(idime,jdime)-(1.0_rp/3.0_rp)*tkron(idime,jdime)*i1)*gpdet(igaus)**(-5.0_rp/3.0_rp)
! print *,  (1.0_rp/3.0_rp)*tkron(idime,jdime)*i1


cauchy_f(idime,jdime)  =   term2 *dev_fporf(idime,jdime)


cauchy_l(idime,jdime)  =   Kct* (gpdet(igaus)-1)*tkron(idime,jdime)

cauchy_fg(idime,jdime)   =  term2_2*dev_fporf_2(idime,jdime)

        end do
     end do

castr = cauchy_gs+ cauchy_f + flag_2*cauchy_fg +cauchy_l




     !Transform Cauchy stress to S
     do idime = 1,ndime
        do jdime = 1,ndime
           do kdime = 1,ndime
              do ldime = 1,ndime
                 gpstr(idime,jdime,igaus) = gpstr(idime,jdime,igaus)+ &
                      gpdet(igaus) *&
                      gpigd(idime,kdime,igaus) * &
                      (castr(kdime,ldime) )*&
                      gpigd(jdime,ldime,igaus)
              end do
           end do
        end do
     end do
```

```
! * * * ****************************
! Calculate tangent constitutive matrix
! * * * ****************************


a1               = 0.0_rp
energy_al          = af*(i4f-1.0_rp)*exp(bf*(i4f-1.0_rp)**2)
energy_al_2        = af*(i6f-1.0_rp)*exp(bf*(i6f-1.0_rp)**2)
energy_al_al       = af*(1.0_rp + 2.0_rp*bf*(i4f-1.0_rp)**2.0_rp)*exp(bf*(i4f-1)**2)
energy_al_al_2     = af*(1.0_rp + 2.0_rp*bf*(i6f-1.0_rp)**2.0_rp)*exp(bf*(i6f-1)**2)

invC              = 0.0_rp
det               = 0.0_rp
if (ndime ==3) then
a1(1,1) = gpcau(3,3,igaus)*gpcau(2,2,igaus)-gpcau(3,2,igaus)*gpcau(2,3,igaus)
a1(1,2) = -(gpcau(3,3,igaus)*gpcau(1,2,igaus)-gpcau(3,2,igaus)*gpcau(1,3,igaus))
a1(1,3) = gpcau(2,3,igaus)*gpcau(1,2,igaus)-gpcau(2,2,igaus)*gpcau(1,3,igaus)
a1(2,1) = -(gpcau(3,3,igaus)*gpcau(2,1,igaus)-gpcau(3,1,igaus)*gpcau(2,3,igaus))
a1(2,2) = gpcau(3,3,igaus)*gpcau(1,1,igaus)-gpcau(3,1,igaus)*gpcau(1,3,igaus)
a1(2,3) = -( gpcau(2,3,igaus)*gpcau(1,1,igaus)-gpcau(2,1,igaus)*gpcau(1,3,igaus))
a1(3,1) = gpcau(3,2,igaus)*gpcau(2,1,igaus)-gpcau(3,1,igaus)*gpcau(2,2,igaus)
a1(3,2) = -(gpcau(3,2,igaus)*gpcau(1,1,igaus)-gpcau(3,1,igaus)*gpcau(1,2,igaus))
a1(3,3) = gpcau(2,2,igaus)*gpcau(1,1,igaus)-gpcau(2,1,igaus)*gpcau(1,2,igaus)

det =   gpcau(1,1,igaus)*(gpcau(3,3,igaus)*gpcau(2,2,igaus)-gpcau(3,2,igaus)*gpcau(2,3,igaus)) &
       -gpcau(2,1,igaus)*(gpcau(3,3,igaus)*gpcau(1,2,igaus)-gpcau(3,2,igaus)*gpcau(1,3,igaus)) &
       +gpcau(3,1,igaus)*(gpcau(2,3,igaus)*gpcau(1,2,igaus)-gpcau(2,2,igaus)*gpcau(1,3,igaus))

invC = (1.0_rp/det)* a1

else

a1(1,1)=   gpcau(2,2,igaus)
a1(2,2)=   gpcau(1,1,igaus)
a1(1,2)= - gpcau(1,2,igaus)
a1(2,1)= - gpcau(2,1,igaus)

det = gpcau(1,1,igaus)*gpcau(2,2,igaus)-gpcau(1,2,igaus)*gpcau(2,1,igaus)
invC = (1.0_rp/det)* a1


end if


invC_bar = invC*gpdet(igaus)**(2.0_rp/3.0_rp)
```

```
det = gpcau(1,1,igaus)*gpcau(2,2,igaus)-gpcau(1,2,igaus)*gpcau(2,1,igaus)
invC = (1.0_rp/det)* a1


end if


invC_bar = invC*gpdet(igaus)**(2.0_rp/3.0_rp)

Dev       = fporf_0   - (1.0_rp/3.0_rp)*i4f*invC_bar
Dev_2     = fporf_0_2 - (1.0_rp/3.0_rp)*i6f*invC_bar



C4 = 0.0_rp
shear_1= 0.0_rp
shear_2= 0.0_rp
shear_3= 0.0_rp
shear_4= 0.0_rp
term1=0.0_rp
term2=0.0_rp
term3=0.0_rp
term4=0.0_rp
term1_2=0.0_rp
term2_2=0.0_rp
term3_2=0.0_rp
term4_2=0.0_rp
bulk_1=0.0_rp
bulk_2=0.0_rp
bulk_3=0.0_rp

     if (flagt == 1_ip) then
        ! Tangent moduli d2W/dE_ij*dE_kl
!        call invmtx(gpcau(:,:,igaus),gpcai,bidon,ndime)
        do idime = 1,ndime
           do jdime = 1,ndime
              do kdime = 1,ndime
                 do ldime=1,ndime
```

```
                    gpdds(idime,jdime,kdime,ldime,igaus)= gpdds(idime,jdime,kdime,ldime,igaus) + &
                                                          term1+term2+term3+term4+bulk_1+bulk_2+bulk_3+ &
                                                          2.0_rp*a*gpdet(igaus)**(-2.0_rp/3.0_rp)*shear +&
                                                          (term1_2+term2_2+term3_2+term4_2)*flag_2


                end do
              end do
            end do
          end do

      end if


  end do !gauss points

end subroutine sld_stress_model_134
```