NTNU – Trondheim
Norwegian University of
Science and Technology

Numerical Modelling and Analysis of the Left Ventricle

Gaute Aasen Slinde

Civil and Environmental Engineering
Submission date: June 2015
Supervisor: Bjørn Helge Skallerud, KT

Norwegian University of Science and Technology
Department of Structural Engineering

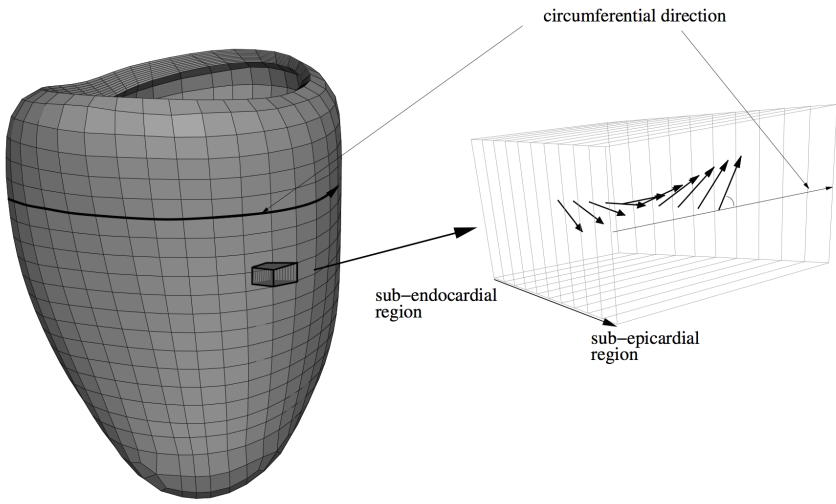
Numerical modelling and analysis of the left ventricle

Candidate: Gaute Aasen Slinde
Supervisors: Bjørn Skallerud, Victorien Prot

January 21, 2015

The heart is an electro-mechanical pump which consists of four chambers: the left and right ventricles and the left and right atria. The left ventricle is a thick-walled muscular chamber that pumps blood at physiologically high pressures throughout the body. The cavity of the LV resembles a truncated ellipsoid in which both the inflow and outflow tracts are adjacent. The material properties and the complex anatomy of the myocardium presents a computational challenge and are of crucial interest in order to understand the pumping function of the intact heart. The topic includes:

- the development of a finite element model of the left ventricle accounting for the complex overall geometry and the structure of the myocardium;
- the investigation of the influence of material anisotropy on the computed wall stress and strain;
- the modelling of the active cardiac muscle fibers;
- the inclusion of the mitral valve and the LVOT in the finite element model.



A finite element model of a human left ventricle taking into account a local material orientation.

Abstract

The development of a well functioning finite element model of the left ventricle is an important step to better understand the pumping function of the human heart. This may be of interest when developing effective treatments for different heart diseases.

The goal of this thesis is to develop a finite element model of the left ventricle, taking into account the material properties and complex structure of the myocardium. The model uses a truncated ellipsoid as geometry and is assigned a linear, transmural variation of both fiber and sheet orientations in the myocardium. Using existing constitutive models of the myocardium, the deformation of the ventricle in systole was analysed using a simple, active stress component. The behaviour of the model was evaluated using the parameters: ejection fraction, torsion, wall thickening, longitudinal shortening and radial shortening. Describing the left ventricular function, these parameters are compared with the physical values.

The results show that in order to realistically model the different ventricular features, active stress components in fiber, sheet normal and shear (sn) directions are all needed. The model is not able to show a realistic ejection fraction even when the active stress contribution is raised to non-physiological levels. The model has in particular problems of producing wall thickening and radial displacement, but still a relatively realistic systolic contraction is seen. The model significantly overestimates the left ventricular torsion. This is in part due to the symmetrical geometry and the fact that the right ventricle is not included.

Samandrag

Utviklingen av ein velfungerande elementmetodemodell av den venstre ventrikkelen er eit steg på vegen til ein betre forståing av pumpeeigenskapane til menneskehjartet. Dette er av interesse for utviklinga av effektive behandlingsmetodar for hjartesjukdom.

Målet for denne oppgåva er å utvikle ein elementmetodemodell av den venstre ventrikkelen, der materialeigenskapane og den komplekse strukturen til myokard er inkludert. Modellen bruker ein trunkert ellipsoide som geometri og en lineær, transmural variasjon av orienteringen til muskelfibrane i myokard. Ved å bruke eksisterande materialmodellar av myokard, har deformasjonen av ventrikkelen i systole blitt analysert ved å implementer ein enkel, aktiv spenningskomponent. Responsen til modellen vart vurdert ved å sjå på parametarane: ejeksjonsfraksjon, torsjon, samt endring i veggtjukkelse, lengde og radius. Parametarane vart så samanlikna med fysioligiske verdiar.

Resultata viser at for å realistisk modellere ventrikkelkontraksjon, må aktive spenningskomponentar i både fiber, normalt på fiber og skjær (sn) retning inkluderast. Modellen er ikkje kapabel til å gi ein realistisk ejeksjonsfraksjon, sjølv når den aktive stresskomponenten vert auka til ikkje-fysiologiske verdiar. Modellen har spesielt problem med å produsere auka veggtjukkelse og radiel forskyvning, men til tross for dette gir modellen ein relativt realistisk systolisk kontraksjon. Torsjonen av ventrikkelen vert i stor grad overestimert i modellen, noko som kan forklarast av den symmetriske geometrien og mangelen av ein høgre ventrikkel.

Preface

This master thesis is submitted to the Norwegian University of Science and Technology (NTNU) for the degree Master of Science (M.Sc.). The work has been carried out at the Department of Structural Engineering, NTNU. My supervisors has been Prof. Bjrn Skallerud and Assoc. Prof. Victorien Prot at the Department of Structural Engineering, NTNU.

Acknowedgements

I would like to thank my supervisor Prof. Bjørn Skallerud and co-supervisor Assoc. Prof. Victorien Prot for all the help and guidance throughout the work with this thesis. Especially, I thank Prof. Bjørn Skallerud for his encouragement to take on this topic and his invaluable input and feedback on my work. I would also like to thank Assoc. Prof. Victorien Prot for his guidance in working with Abaqus and for providing much of the background literature.

I would like to thank Assoc. Prof. Ian LeGrice at The University of Auckland for providing me with the background data from the passive myocardium shear experiments.

Finally, I would like to thank my family and friends for their encouragement during the work on my thesis.

Trondheim, June 2015
Gaute Aasen Slinde

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Heart disease is one of the primary causes of death in the world today. There is a hope that advances made within biomechanics can improve our scientific understanding of the heart. A deeper knowledge of the structural and mechanical functions of the heart may lead to a better understanding of different diseases and enable us to develop more effective treatments.

The mechanical function of the heart depends greatly on its mechanical properties. Understanding how these properties relates to the structure of the myocardium and how the structure varies within the heart is an important step to fully understand how the pumping effect of the heart is created. The purpose of this thesis is to review the current knowledge of the myocardium structure and the models which tries to explain it. By implementing a constitutive material law into a finite element analysis we can get a better understanding of the left ventricular mechanics.

The analysis of soft biological tissue present some challenges. The myocardium is anisotropic, inhomogeneous, incompressible and non-linear, all which must be taken into account. The problem calls for the need of finite deformation theory, as opposed to small-strain elasticity theory.

This thesis will be based in great part of the work of Holzapfel and Ogden [14]. The constitutive model presented there will be implemented in to the finite element analysis program Abaqus in a full scale model of the left ventricle. A truncated ellipsoid is used and the geometry is chosen to approximate the volume of a physical human heart. An active stress component will be added and the contraction of the left ventricle in systole will be studied. As this has not been attempted earlier, some computational aspects will also be studied. Soft tissue finite element problems is known to exhibit some numerical challenges, and computational cost for different mesh refinement will be weighed against the needed accuracy.

In chapter 2 of this thesis the anatomy of the heart and the cardiovascular system is presented. Here the architecture of the heart and how it functions as a pump in the cardiovascular system is explained. The the structure cardiac muscle given some attention and

the mechanics behind ventricular contraction is explained. Further, the essential elements of continuum mechanics is presented. This includes a look at the kinematics and invariant theory, and the derivation of the strain-energy function and stress tensors. We then review some of the existing constitutive models of the myocardium. Here the evolution of the different models is outlined, moving from isotropic and transverse isotropic models, to the current orthotropic models. Lastly, the constitutive model used this thesis is presented thoroughly. In chapter 3 we move on to the modelling aspects of the thesis. This includes the implementation of both the passive and active part of the constitutive model in to Abaqus. The different aspects of the Abaqus model is then presented, including the choice of loads and boundary conditions, and reflection around the meshing technique. The implementation of the fiber field is also presented in this chapter. In chapter 4 the results of the different analysis is presented. Studies on different aspects of the model is performed, including some discussion of the results. In chapter 5 the findings is discussed further, both by comparing with physiological values and reflecting on the implication of the different assumptions and simplifications. Finally, some concluding remarks are presented in chapter 6.

CHAPTER 2
THEORY

2.1 Anatomy

2.1.1 The Cardiovascular System

The cardiovascular system is the rapid connective transport system which supplies the body with among other oxygen, glucose, vitamins, drugs and water. It is also a control system distributing hormones, and is crucial for the regulation of body temperature. The human heart is a hollow muscle which functions as the pump that drives the blood through cardiovascular system. The heart has four chambers, where the right and left ventricles are filled from the right and left atrium, respectively. The right ventricle pumps de-oxygenated blood through the pulmonary trunk to the lungs after which oxygenated blood continues through the pulmonary veins into the left atrium. This completes the short, low pressure pulmonary circulation. The left ventricle pumps oxygenated blood through the aorta which after repeated branching reaches the many capillaries where oxygen diffuses to all of the different cells in the body. De-oxygenated blood returns to the right atrium through the superior and inferior vena cava. The ventricles pumps at the same time and the same volume of blood, but the left ventricle at a much higher pressure.

The left ventricle is the main pumping chamber of the heart. It has a thick wall, typically around 10 mm, and is able to generate a high pulse pressure of 120 mmHg, or $\sim$16 kPa. In comparison the pressure in the right ventricle is only up to 25 mmHg $\sim$3-4kPa and therefore has thinner wall than the left. The atria again has even lower pressures and the walls are thin in comparison with the ventricular walls.

The heart has four valves to prevent a back flow of blood. The tricuspid valve connects the right atrium to the right ventricle. The pulmonary valve guards the outlet from the right ventricle to the pulmonary artery. The mitral valve lies between the left atrium and ventricle and the aortic valve lies at the root of the aorta. These valves open and close depending on the pressure gradient through the valves. Both the tricuspid and mitral cusp margins are tethered by the chordea tendineae. These tendunous strings, with inward

projection from the ventricular wall, are tense during systole and thus preventing the valves from reverting into the atrium as the pressure rises.

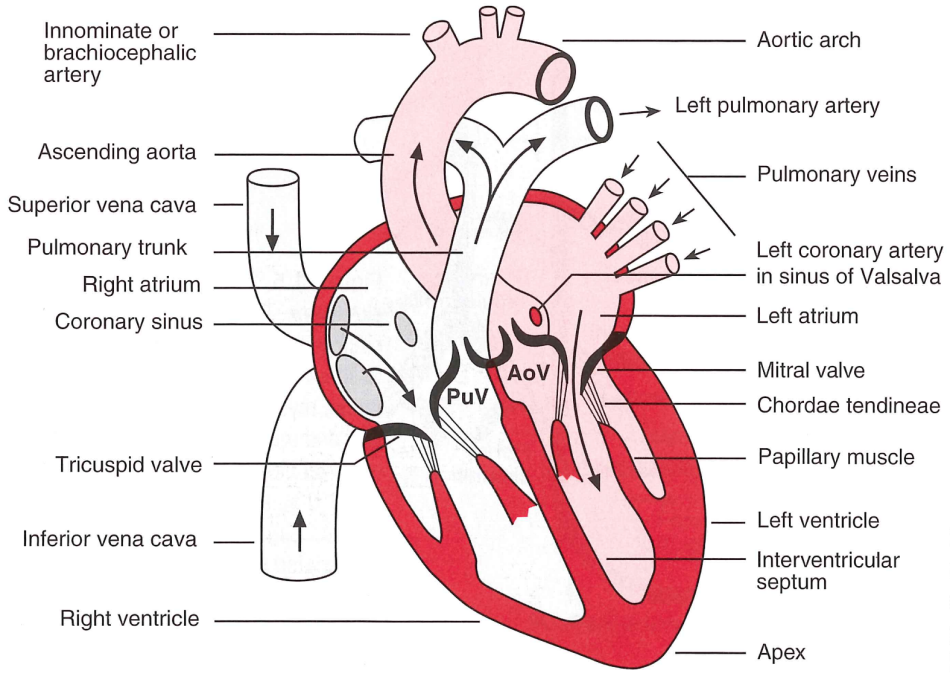


Figure 2.1: Structure of human heart. Pink indicates oxygenated blood, where as grey indicates deoxygenated blood. [20]

The cardiac output is defined as the volume of blood ejected by a ventricle in a minute, and thus the output is a product of the stroke volume and the heart rate. A resting adult has a stroke volume of about 70-80 ml and with a heart rate of 50-75 beats per minutes the resting cardiac output is generally 5 liter per minute. The heart has to make available sufficient amount of blood to allow organs to perform their function, and the output can increase five times during hard exercise.

The walls of the ventricles consists of three layers as shown in Figure 2.2. The *endocardium* is a thin sheet lining the inner surface. The *myocardium* is the muscle layer responsible for the contraction of the heart. The *epicardium*, the outer layer, is a thin sheet of connective tissue. The entire heart is enclosed in a fibrous sac called the *pericardium*. [20]

2.1.2 The Ventricular Cycle

The ventricular cycle describes the cycle of ventricular contraction. It is divided into four phases: ventricular filling, isovolumetric contraction, ejection and isovolumetric relaxation. The cycle can also be divided in to the contracting phase, called systole, and the relaxing phase, called diastole.

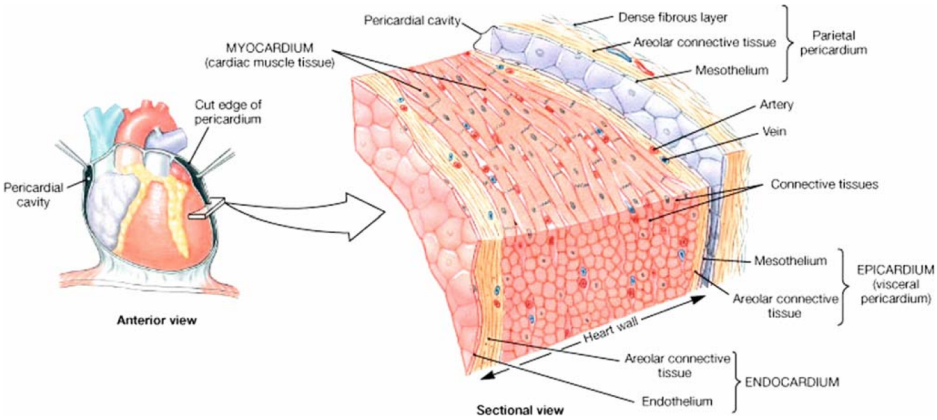


Figure 2.2: Cross section of the heart wall. [18]

During the initial phase of the *ventricular filling* both the atria and ventricles are in diastole and thus blood is passively flowing from the veins through the atria and valves into the ventricles. As the ventricle is recoiling elastically from its end-systolic shape, it is creating a pressure gradient sucking blood in to the ventricles in this early diastole phase. Because of this, the ventricular pressure is actually falling in the initial rapid-filling phase. The ventricle reaches a relaxed volume and the rate of filling slows only driven by the venous pressure. Finally, atrial contraction pumps blood into the ventricle completing the filling phase, which lasts about 0.5s in a resting human. The volume of blood in the ventricle at this time is referred to as the end-diastolic volume and the corresponding pressure, end-diastolic pressure.

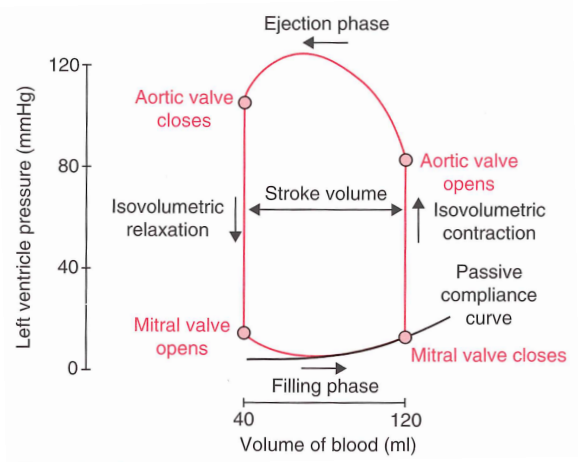


Figure 2.3: Pressure-volume loop for resting human left ventricle. [20]

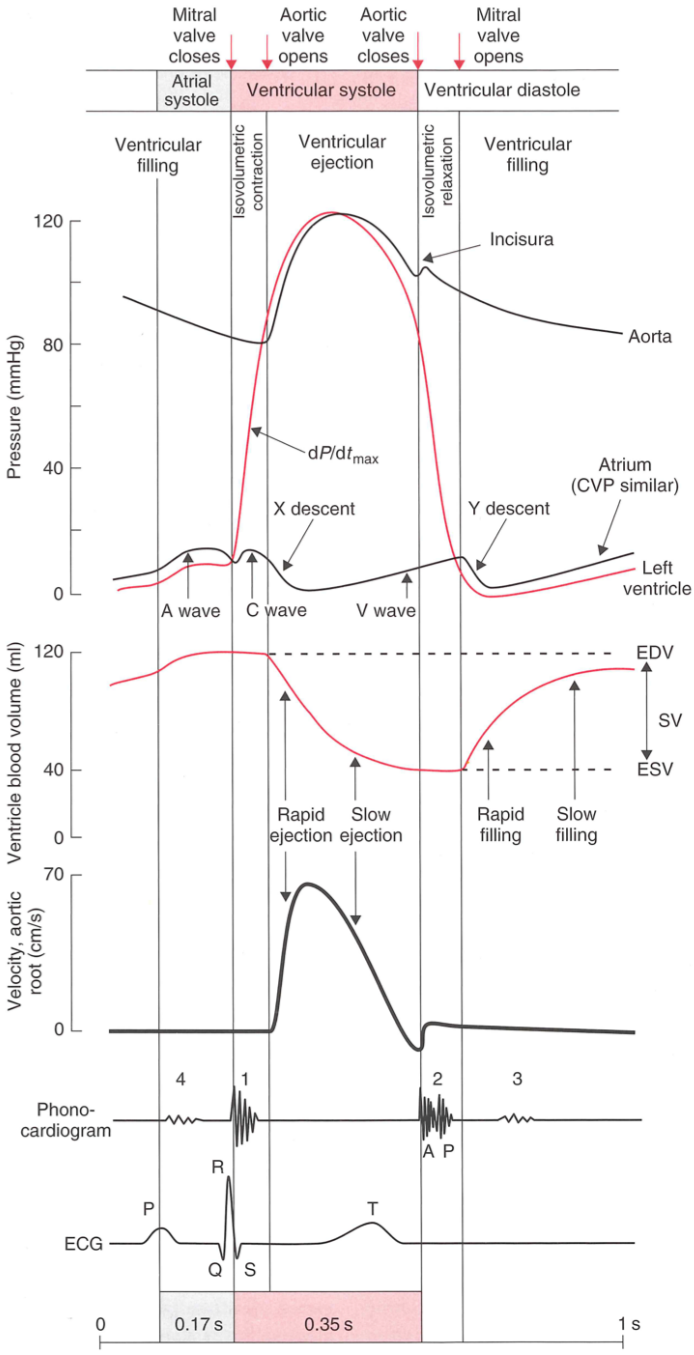


Figure 2.4: Changes in pressure, volume and flow for aorta, left ventricle and left atrium during human cardiac cycle. [20]

Ventricular systole begins with a brief *isovolumetric contraction* lasting 0.05s. The pressure in the ventricles rise above the pressure in the atria and the valves are thus shut, making the ventricles closed chambers. The tension of the contracting walls causes the blood pressure to rise very fast. Just as the ventricular pressure surpasses the arterial pressure, the outflow valves open and the *ejection* phase starts. The initial rapid ejection phase, lasting 0.15s, accounts for three quarters of the stroke volume. Later, as the rate of which the aortic blood is draining away exceeds the ventricular ejection, the pressure drops. The outward momentum of the blood keeps the aortic valve open, but since the pressure gradient is now reversed, the outflow decelerates until the valve is closed by a backflow. This marks the end of ventricular systole.

Both the pulmonary and the aortic valves are now closed, creating a closed chamber in the *isovolumetric relaxation* phase. The elastic recoil of the deformed, relaxing myocardium causes the ventricular blood pressure to drop rapidly. As soon as the pressure drops below the atrial pressure, after approximately 0.08s, the valves open, starting the next ventricular cycle. The entire cycle is visually described through the ventricular pressure-volume loop in Figure 2.3, and also in Figure 2.4. [20]

2.1.3 The Cardiac Myocyte

The cardiac myocytes are the muscle cells which adds up to create the myocardium. The myocytes are packed with with long, contractile bundles called myofibrils. These myofibriles are composed of many, basic contractile unites called *sarcomeres*. The sarcomere comprises a set of filamentous proteins, between two thin partitions called the Z lines, composed of the protein α-acitin. Between the Z lines lies the thick filaments, composed of the protein myosin, and thin filaments, composed of the protein actin. *Thick myosin filaments* are arranged parallel in the center of the sarcomere, called the A band. Each myosin molecule has two heavy chains forming a double helix making up the tail of the molecule. At one end, two free heads stick out on the side of the filament, making the molecule resemble the form of a golf club. *Thin actin filament* lies in between the myosin filaments, with one end in the A band and one in the Z line, forming the I band.

The Mechanics of Contraction

The contraction of the heart is done by the shortening of the sarcomeres. During this shortening the thin actin filaments slides into the spaces between the thick myosin filaments. The motion is created by the the repeating making, rotation and breaking of the so called *crossbriges*, which are biochemical bonds between the thin and thick filaments. Each myosin head contributes to the force generation by protruding from the side of the thick filament. Each head acts as a force generator and the countless numbers of myosin heads sum up to a significant force. Each actin subunit has a binding site for a myosin head, however the site is blocked at rest by the molecule tropomyosin. Each tropomyosin has a troponin complex attaced to one end. Exposure of the binding site is created by a sudden rise in the concentration of free Ca^{2+}. This causes the tropomyosin-troponin complex to move deeper into the thin filament, which it lies around, and thus exposing the myosin binding sites. This enables the myosin head to bind to the actin and form a crossbridge. The force of the contraction is proportional to the number of crossbridges formed and

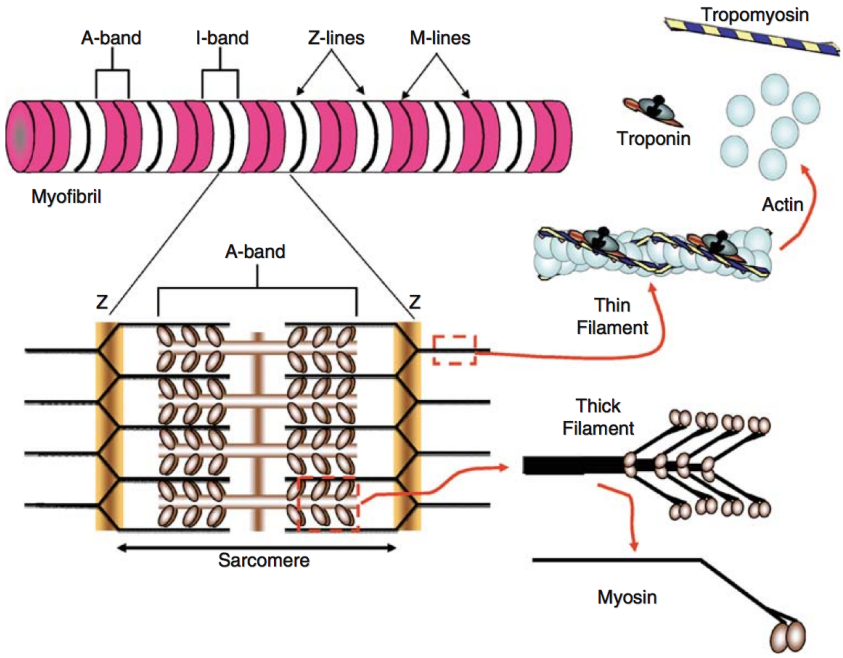


Figure 2.5: Schematic of myofibrils and sarcomeres. [2]

is therefore dependant on the Ca^{2+} concentration during exitation. In the heart, the Ca^{2+} concentration only reach 0.5-2μM, activating only a small part of the potential crossbridge sites. Adrenaline, which increases the Ca^{2+} level, is therefore efficient in increasing the force of the heartbeat. [20]

2.1.4 The Cardiac Structure

The architecture of the heart still remains controversial and debated, and several competing models explaining the heart structure and function, exist. Seeing as the biological understanding lies as a foundation for a solid computational model, insight into the myocardial architecture is important. [10]

Several anatomical studies have revealed that the cardiac tissue is a composite of discrete layers of myocardial muscle fibers bound tightly by endomysial collagen. The myocardial laminea, or sheets, are coupled by perimysial collagen and can slide over each other without much resistance. The lamineas are four to six cells thick and continuously branch throughout the ventricular wall. [23] As can be seen from Figure 2.6, the muscle fiber orientations change with the position through the wall. This feature is important for the ventricular function and plays an important part in the further work. [14]

As an important part of describing the mechanical properties of the myocardium, the laminar structure is characterized by identifying the axes align

1. with the myocyte direction ($\mathbf{f}_0$), also called the fiber direction,

2. transverse to the myocyte axis within a layer ($\mathbf{s}_0$), and

3. normal to the layer ($\mathbf{n}_0$).

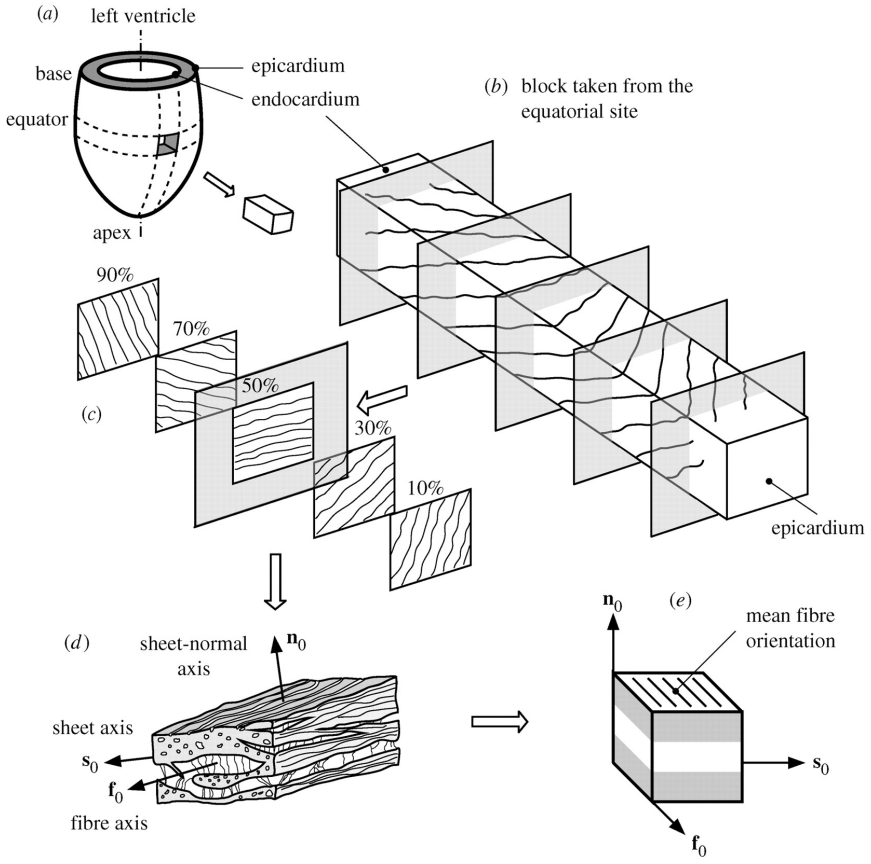


Figure 2.6: Definition of local myocyte coordinate system and visualization of transmural variation of layer orientation.[14]

2.1.5 Features of Ventricular Contraction

The deformation of the left ventricle during systole is described by some geometrical parameters, which are presented in this section. The physical values of these parameters are used later for comparison with those obtained in the finite element model.

Ejection Fraction

The ejection fraction (EF) is the parameter which is most widely used to describe the ventricular function. It is defined as

$$EF = \frac{ESV - EDV}{EDV}, \tag{2.1}$$

where ESV is the end-systole volume and EDV is the end-diasole volume. Levick [20] defines the normal value of the ejection fraction to be 67% in a healty, resting human heart and values in the range 55-65% is considered to be normal. [8]

Left Ventricular Torsion

Left ventricular torsion is the wringing motion of the ventricle around its long axis. Some definitions are appropriate as the terminology differs in the literature. The rotation is the rotatory movement about the long axis and during systole counterclockwise rotation will be expressed with positive values when looking from the apex towards the base.

The fibers is running in a left handed helix from the apex to the base in the sub-epicardium and a right handed helix in the sub-endocardium. As the muscle fibers contracts during systole, the fiber orientation cause the apex to rotate in a counterclockwise direction and the base in a clockwise direction. Seeing as the direction of the fibers is opposite in the sub-epicardium and sub-endocardium, the rotation is also opposite on the two surfaces. The global torsion of the left ventricle follows predominantly the rotation of the sub-epicardium in a counter clockwise direction.

The torsion is understood to play an important part in the ventricular contraction. The shortening of myocardial fibers is about 10-15% and if there were no rotation this shortening is not able to account for the ejection fraction of the human heart being 55-65%. Therefore the torsion is crucial for the overall function of the left ventricle.

In this thesis the definition of absolute myocardial torsion (AMT) will be used, where AMT = AMR - BMR is the difference between the maximum rotation at the most basal myocardial section (BMR) and at the apex (AMR). At end-systole these angles has be experimentally record to -3.71° $\pm$ 0.84° and 6.73° $\pm$ 1.69° for basal and apical planes, respectively, resulting in left ventricular mean AMT of 10.48 ° $\pm$ 1.63 °. [3] [26] [21]

Wall thickening

It is generally accepted that the ventricular wall thickening during systole plays an important part in the contractile function. By defining the the wall thickness at end-diastole, h_{ed}, and the wall thickness at end-systole, h_{es}, we calculate the fractional thickening,

$$h_f = \frac{h_{ed} - h_{es}}{h_{ed}}. \tag{2.2}$$

Dumesnil and Shoucri [7] gives a value for the ventricular wall thickening h_f = 0.52, which will be used as comparison to the model results.[9]

Longitudinal Shortening

The relative ventricular longitudinal shortening is defined as $|\frac{\Delta L}{L}|$, where L is the internal longitudinal distance from the apex to base. Dumesnil and Shoucri [7] found a mean value of 0.21 from a study of 9 healthy human hearts, while Carreras et al. [3] found found a similar mean value of $19.07^\circ \pm 2.71^\circ$ using the the distance between the base and the epicardial part of the apex.

Radial Shortening

The relative radial shortening is defined as $\frac{\Delta R}{R}$, where R is the internal radius of the left ventricle. Dumesnil and Shoucri [7] found the radial shortening to be in the range 31-39% in healthy human hearts.

2.2 Continuum Mechanics

2.2.1 Kinematics

Lets consider a continuum body $\mathfrak{B}$ initially occupying the region Ω_0 at the reference time t = 0, known as the reference configuration. A point in Ω_0 is characterized by the position vector $\mathbf{X}$. At a time $t > 0$ the continuum body is in a deformed configuration and now occupying a region Ω. The same point in Ω is now characterized by the position vector $\mathbf{x}$. The deformation gradient $\mathbf{F}$ is the primary measure of deformation in nonlinear continuum mechanics and is defined as

$$\mathbf{F}(\mathbf{X}, t) = \frac{d\mathbf{x}}{d\mathbf{X}} \tag{2.3}$$

In an anisotropic material, the stress at a material point depends not only on the deformation gradient, but also the preferred direction of the material at the point, called the fiber direction. Let $\mathbf{a}_0$ denote the preferred direction in the reference configuration. Under deformation the fiber length changes, and we define the stretch λ as the ratio between the length of the fiber in the deformed and reference configuration along it direction $\mathbf{a}_0$,

$$\lambda \mathbf{a}(\mathbf{x}, t) = \mathbf{F}(\mathbf{X}, t)\mathbf{a}_0(\mathbf{X}). \tag{2.4}$$

Using standard convension we have

$$J = det\, \mathbf{F} > 0, \tag{2.5}$$

where J is the Jacobian determinant for the deformation gradient $\mathbf{F}$. For an incompressible material, we have the definition

$$J = det\, \mathbf{F} \equiv 1 \tag{2.6}$$

The right and left Cauchy-Green tensors are defined by

$$\mathbf{C} = \mathbf{F}^T\mathbf{F} \quad \text{and} \quad \mathbf{B} = \mathbf{F}\mathbf{F}^T, \tag{2.7}$$

respectively, and the Green-Lagrange strain tensor is defined by

$$\mathbf{E} = \frac{1}{2}(\mathbf{C} - \mathbf{I}), \tag{2.8}$$

where **I** is the identity matrix. Introducing the principle invariants of **C**, we have

$$I_1 = tr\ \mathbf{C}, \quad I_2 = \frac{1}{2}[I_1^2 - tr\ (\mathbf{C}^2)] \quad \text{and} \quad I_3 = det\ \mathbf{C}, \tag{2.9}$$

and thus $I_3 = J^2 = 1$ for an incompressible material. These are only the isotropic invariants, and to introduce anisotropy more will have to be included. Remembering $\mathbf{a}_0$ denoting the preferred direction, we then introduce two transversely isotropic invariants,

$$I_4 = \mathbf{a}_0 \cdot (\mathbf{C}\mathbf{a}_0) \quad \text{and} \quad I_5 = \mathbf{a}_0 \cdot (\mathbf{C}^2\mathbf{a}_0). \tag{2.10}$$

When there are two preferred direction, the second direction is denoted $\mathbf{b}_0$, which introduces the invariants

$$I_6 = \mathbf{b}_0 \cdot (\mathbf{C}\mathbf{b}_0) \quad \text{and} \quad I_7 = \mathbf{b}_0 \cdot (\mathbf{C}^2\mathbf{b}_0). \tag{2.11}$$

A final coupling invariant is defined as

$$I_8 = \mathbf{a}_0 \cdot (\mathbf{C}\mathbf{b}_0) = \mathbf{b}_0 \cdot (\mathbf{C}\mathbf{a}_0) \tag{2.12}$$

Note that I_8 is not a true invariant, seeing as reversing the sign of either $\mathbf{a}_0$ or $\mathbf{b}_0$, changes the sign if I_8. The formulation is however convenient in the following.

2.2.2 The Strain-Energy Function and Stress Tensors

In this section we look at the material properties described by a strain-energy function $\mathbf{\Psi}$, measured per unit reference volume. The strain-energy function depends on the deformation gradient **F** through **C**. For an elastic material, the Cauchy stress tensor $\boldsymbol{\sigma}$ is given by

$$J\boldsymbol{\sigma} = \mathbf{F}\frac{\partial \mathbf{\Psi}}{\partial \mathbf{F}} = \mathbf{F}\frac{\partial \mathbf{\Psi}}{\partial \mathbf{E}}\mathbf{F}^T \tag{2.13}$$

for a compressible material, which modified becomes

$$\boldsymbol{\sigma} = \mathbf{F}\frac{\partial \mathbf{\Psi}}{\partial \mathbf{F}} - p\mathbf{I} = \mathbf{F}\frac{\partial \mathbf{\Psi}}{\partial \mathbf{E}}\mathbf{F}^T - p\mathbf{I} \tag{2.14}$$

for an incompressible material, where $J = 1$ accommodated in the expression by the Langrange multiplier p. For an elastic material with a strain-energy function $\mathbf{\Psi}$ depending on a list of invariants I_1, I_2, ..., I_N for some N, equations (2.13) and (2.14) are expanded to the forms

$$J\boldsymbol{\sigma} = \mathbf{F}\sum_{i=1}^{N}\psi_i\frac{\partial I_i}{\partial \mathbf{F}} \quad \text{and} \quad \boldsymbol{\sigma} = \mathbf{F}\sum_{i=1,i\neq 3}^{N}\psi_i\frac{\partial I_i}{\partial \mathbf{F}} - p\mathbf{I}, \tag{2.15}$$

respectively. We have introduced the notation

$$\psi_i = \frac{\partial \mathbf{\Psi}}{\partial I_i}, \quad i = 1, 2, ..., N, \tag{2.16}$$

where $i = 3$ is omitted from the summation and I_3 from the list of invariants for the incompressible material.

The second Piola-Kirchoff stress tensor **S** is given in terms of the Cauchy stress tensor via the formula $\mathbf{S} = J\mathbf{F}^{-1}\boldsymbol{\sigma}\mathbf{F}^{-T}$. Explicitly, by using equations (2.13) and (2.14), and with **E** as the independent variable, we have

$$\mathbf{S} = \frac{\partial \boldsymbol{\Psi}}{\partial \mathbf{E}} \quad \text{and} \quad \mathbf{S} = \frac{\partial \boldsymbol{\Psi}}{\partial \mathbf{E}} - p(\mathbf{I} + 2\mathbf{E})^{-1} \tag{2.17}$$

for compressible and incompressible materials, respectively. [13] [14]

2.3 Constitutive Models for Passive Myocardium

In this section a selection of the existing constitutive models of the myocardium will be reviewed. Some of the earlier models which are based on linear isotropic elasticity are not mentioned, as they do not capture the definite anisotropy of the myocardium.

2.3.1 Transversely Isotropic Models

One of the first proposed invariant based models that takes to into account the fiber structure of the myocardium was, Humphrey and Yin [15]. They proposed the strain energy function

$$\boldsymbol{\Psi} = c\,\{\exp[b(I_1 - 3)] - 1\} + A\,\{\exp[a(\sqrt{I_4} - 1)^2] - 1\}, \tag{2.18}$$

using four material parameters. Later however, it was determined that the myocardium is not a transversely isotropic material, and the models using this assumptions are thus inappropriate.

2.3.2 Orthotropic Models

In this section three main orthotropic models are presented. Several others have been proposed, but deemed inappropriate for modelling myocardial tissue, as they do not reflect the morphology of the myocardium. The models reviewed all share the basic foundation as they are partly structurally based, taking into account the fiber, sheet and normal material directions, and partly phenomenological, thus trying to replicate the myocardium behaviour without necessarily trying to explain it.

Strain-energy function proposed by Costa et al. [4]

Costa et al. [4] proposed a Fung-type exponential strain-energy function given by

$$\boldsymbol{\Psi} = \frac{1}{2}a(\exp Q - 1), \tag{2.19}$$

where

$$Q = b_{ff}E_{ff}^2 + b_{ss}E_{ss}^2 + b_{nn}E_{nn}^2 + 2b_{fs}E_{fs}^2 + 2b_{fn}E_{fn}^2 + 2b_{sn}E_{sn}^2, \tag{2.20}$$

with seven material parameters, a and b_{ij} where $i, j \in \{f, s, n\}$. Their proposed model is based on earlier work with a transversely isotropic, exponential strain energy function.

This is generalized and extended to material orthotropy by recognizing that the ventricular myofibers are organized into branching laminae, suggesting that the myocardium may be locally orthotropic with distinct cross-fiber stiffness within and across the sheet plane. They further note the difficulty with a constitutive model based only on biaxial tissue test, as there is uncertainty to how the biaxial properties of isolated tissue slices are related to the properties of the intact ventricular wall. Shear deformations are an important component of the mechanics in the intact heart, ant the then current biaxial protocol did not include the in-plane shear at the time when the model was proposed. [4]

Fung-type model proposed by Schmid et al. [27]

Schmid et al. [27] introduced another Fung-type model by separating the exponential terms for each component and thus decouple the effects of the material parameters in the single-exponential model of Costa et al. [4]. The model with 12 material parameters, is given by

$$\begin{aligned}\boldsymbol{\Psi} = &\frac{1}{2}a_{ff}[\exp(b_{ff}E_{ff}^2) - 1] + \frac{1}{2}a_{fn}[\exp(b_{fn}E_{fn}^2) - 1] + \frac{1}{2}a_{fs}[\exp(b_{fs}E_{fs}^2) - 1] \\ &+\frac{1}{2}a_{nn}[\exp(b_{nn}E_{nn}^2) - 1] + \frac{1}{2}a_{ns}[\exp(b_{ns}E_{ns}^2) - 1] + \frac{1}{2}a_{ss}[\exp(b_{ss}E_{ss}^2) - 1].\end{aligned} \tag{2.21}$$

The Pole-zero model

The pole-zero model is based on evidence from several biaxial tension tests on thin sections of ventricle myocardium which reveals a highly nonlinear, anisotropic stress-strain behaviour. Figure 2.7 depicts this typical stress-strain behaviour of the myocardium. Here the distinct difference in the properties along each of the micro strctural relevant directions are clearly seen, and Nash and Hunter [23] notes the large difference in the limiting strain for an elastic response of the three axes. This difference can be explained by the organisation of the extracellular connective matrix. High fiber stiffness can be attributed to the intracellular titin together with the tightly bound endomysial collagen coils surrounding the individual myocytes. The relatively low sheet-normal stiffness is most likely due to the sparse array of perimysial collagen links in the cleavage planes between myocardial sheets.

Since the stress-strain behaviour along one axis is nearly independent of lateral stretch, the contribution to the total strain energy from the stretch along one axis is nearly independent of the contribution from the two other axes. The small cross-axis coupling from the hydrostatic pressure is neglected since hydrostatic pressure is zero in a bi-axial tension test. Therefore the strain energy function is seperated into individual expressions for the stretch along each material axes. Further it is noted that, which is easily observed in Figure 2.7, that small axial strain gives very low axial stress, but stress rapidly increases as the strain approaches the strain limit for that axis. This material behaviour and micro structural observations are all included in the pole-zero strain-energy function first proposed by

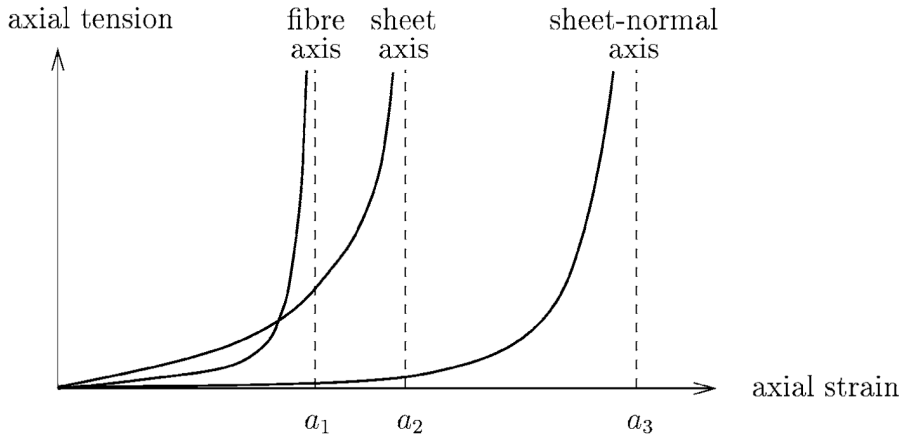


Figure 2.7: Typical nonlinear stress-strain properties of ventricular myocardium. [23]

Hunter et al. (1997):

$$\begin{aligned}\boldsymbol{\Psi} = &\frac{k_{ff}E_{ff}^2}{|a_{ff}-|E_{ff}||^{b_{ff}}} + \frac{k_{fn}E_{fn}^2}{|a_{fn}-|E_{fn}||^{b_{fn}}} + \frac{k_{nn}E_{nn}^2}{|a_{nn}-|E_{nn}||^{b_{nn}}} \\ &+\frac{k_{fs}E_{fs}^2}{|a_{fs}-|E_{fs}||^{b_{fs}}} + \frac{k_{ss}E_{ss}^2}{|a_{ss}-|E_{ss}||^{b_{ss}}} + \frac{k_{ns}E_{ns}^2}{|a_{nsf}-|E_{ns}||^{b_{ns}}},\end{aligned} \tag{2.22}$$

with 18 material parameters k_{ij}, a_{ij} and b_{ij}, where $i, j \in f, s, n$, and the different compoenents of E_{ij} are separated. Here a_{ij} represents the limiting strains or poles, b_{ij} are related to the curvature of the uniaxial stress-strain relasionship and lastly the k_{ij} parameters weight the contribution from each corresponding mode of deformation to the total strain energy of the material.

2.3.3 The Structurally Based Model Proposed by Holzapfel and Ogden [14]

This section outlines the basis for the constitutive model which is used for the remainder of this thesis. By using the fiber, sheet and sheet-normal directions specified, and the definition of the invariant I_4 in the first part of equation (2.10), the invariant I_4 associated with each direction is defined as

$$I_{4f} = \mathbf{f}_0 \cdot (\mathbf{C}\mathbf{f}_0), \quad I_{4s} = \mathbf{s}_0 \cdot (\mathbf{C}\mathbf{s}_0) \quad \text{and} \quad I_{4n} = \mathbf{n}_0 \cdot (\mathbf{C}\mathbf{n}_0). \tag{2.23}$$

By noting that

$$\sum_{i=f,s,n} I_{4i} = \mathbf{C} \colon (\mathbf{f}_0 \otimes \mathbf{f}_0 + \mathbf{s}_0 \otimes \mathbf{s}_0 + \mathbf{n}_0 \otimes \mathbf{n}_0) = \mathbf{C} \colon \mathbf{I} = I_1 \tag{2.24}$$

only three of the invariants I_{4f}, I_{4s}, I_{4n} and I_1 are independent, and thus one of these might be dropped. Similarly, for the second part of equation (2.10), the invariants I_{5f}, I_{5s} and I_{5n} may be defined for each direction. These are however not needed further as they are expressible in terms of the other invariants. Further, by using equation (2.12) and the here specified direction, we have

$$I_{8fs} = I_{8sf} = \mathbf{f}_0 \cdot (\mathbf{C}\mathbf{s}_0), \quad I_{8fn} = I_{8nf} = \mathbf{f}_0 \cdot (\mathbf{C}\mathbf{n}_0) \quad \text{and} \quad I_{8sn} = I_{8ns} = \mathbf{s}_0 \cdot (\mathbf{C}\mathbf{n}_0). \tag{2.25}$$

The data from Dokos et al. [5], which performed simple shear experiments on the myocardium, indicate that the shear response is stiffest when the fiber direction is extended, least stiff in the normal direction and intermediate stiff in the sheet direction. Further, the data indicate that there are differences between (fs) and (fn) and between the (sf) and (sn). The data can not distinguish responses for (nf) and (ns). The results of the shear experiments is presented in Figure 4.2. To capture these differences in a strain-energy function, one or more of the coupling invariants must be included. By choosing I_1, I_2, I_3, I_{4f}, I_{4s}, I_{8fs} and I_{8fn}, the Cauchy stress from equation (2.7) becomes

$$\begin{aligned} J\boldsymbol{\sigma} = 2\psi_1\mathbf{B} + 2\psi_2(I_1\mathbf{B} - \mathbf{B}^2) + 2I_3\psi_2\mathbf{I} + 2\psi_{4f}\mathbf{f}\otimes\mathbf{f} + 2\psi_{4s}\mathbf{s}\otimes\mathbf{s} \\ +\psi_{8fs}(\mathbf{f}\otimes\mathbf{s} + \mathbf{s}\otimes\mathbf{f}) + \psi_{8fn}(\mathbf{f}\otimes\mathbf{n} + \mathbf{n}\otimes\mathbf{f}), \end{aligned} \tag{2.26}$$

for a compressible material. The invariants I_{8fs} and I_{8fn} in equation (2.26) changes signs if the sense of one of the vectors $\mathbf{f}_0$, $\mathbf{s}_0$ and $\mathbf{n}_0$ is reversed. However, $\boldsymbol{\Psi}$ should be independent of this sense. By writing $\hat{\boldsymbol{\Psi}}(..., I^2_{8fs}, ...) = \boldsymbol{\Psi}(..., I_{8fs}, ...)$, then $\psi_{8fs} = 2\frac{\partial\hat{\boldsymbol{\Psi}}}{\partial(I^2_{8fs})}I_{8fs}$ and the shear in the (fs) plane becomes $I_{8fs} = \mathbf{f}\cdot\mathbf{s} = \gamma$ in either direction and vanishes in the reference configuration. As long as $\boldsymbol{\Psi}$ is well behaved, ψ_{8fs} also disappears in the reference state. This also holds for $I_{8fn} = \mathbf{f}\cdot\mathbf{n} = \gamma$ for shear in the (fn) plane and $I_{8sn} = \mathbf{s}\cdot\mathbf{n} = \gamma$ for shear in the (sn) plane. With these conditions in mind, equation (2.26) is reduced to

$$2(\psi_1 + 2\psi_2 + \psi_3)\mathbf{I} + 2\psi_{4f}\mathbf{f}_0\otimes\mathbf{f}_0 + 2\psi_{4s}\mathbf{s}_0\otimes\mathbf{s}_0 = \mathbf{0}, \tag{2.27}$$

in the reference configuration. This only holds if

$$\psi_1 + 2\psi_2 + \psi_3 = 0, \quad \psi_{4f} = 0 \quad \text{and} \quad \psi_{4s} = 0, \tag{2.28}$$

as well as

$$\psi_{8fs} = \psi_{8fn} = 0. \tag{2.29}$$

Similar, for an incompressible material

$$\begin{aligned} \boldsymbol{\sigma} = 2\psi_1\mathbf{B} + 2\psi_2(I_1\mathbf{B} - \mathbf{B}^2) - p\mathbf{I} + 2\psi_{4f}\mathbf{f}\otimes\mathbf{f} + 2\psi_{4s}\mathbf{s}\otimes\mathbf{s} \\ +\psi_{8fs}(\mathbf{f}\otimes\mathbf{s} + \mathbf{s}\otimes\mathbf{f}) + \psi_{8fn}(\mathbf{f}\otimes\mathbf{n} + \mathbf{n}\otimes\mathbf{f}), \end{aligned} \tag{2.30}$$

with only six inveriants I_1, I_2, I_{4f}, I_{4s}, I_{8fs} and I_{8fn}, which holds for the same conditions as before, except the first equation in (2.28) which is replaced by $2\psi_1 + 4\psi_2 - p_0 = 0$, where p_0 is the value of p in the reference configuration. The amount of shear stress versus

the amount of shear for the six simple shears are given by

$$(fs): \quad \sigma_{fs} = 2(\psi_1 + \psi_2 + \psi_{4f})\gamma + \psi_{8fs}, \tag{2.31}$$

$$(fn): \quad \sigma_{fn} = 2(\psi_1 + \psi_2 + \psi_{4f})\gamma + \psi_{8fn}, \tag{2.32}$$

$$(sf): \quad \sigma_{sf} = 2(\psi_1 + \psi_2 + \psi_{4s})\gamma + \psi_{8fs}, \tag{2.33}$$

$$(sn): \quad \sigma_{sn} = 2(\psi_1 + \psi_2 + \psi_{4s})\gamma, \tag{2.34}$$

$$(nf): \quad \sigma_{nf} = 2(\psi_1 + \psi_2)\gamma + \psi_{8fn}, \tag{2.35}$$

and $$(ns): \quad \sigma_{ns} = 2(\psi_1 + \psi_2)\gamma. \tag{2.36}$$

2.3.4 Holzapfel's Specific Model

With the structurally based model for the passive myocardium derived in the previous section, Holzapfel and Ogden [14] further derives a particular model by interpreting the different invariant and choosing which to include in the strain-energy function. The invariant I_1, the isotropic term, is included and can be regarded as associated with the underlying non-collagenous and non-muscular matrix. This can be modelled both as a neo-Hookean and exponential term. Figure 2.8 shows a schematic of the arrangement of muscle and

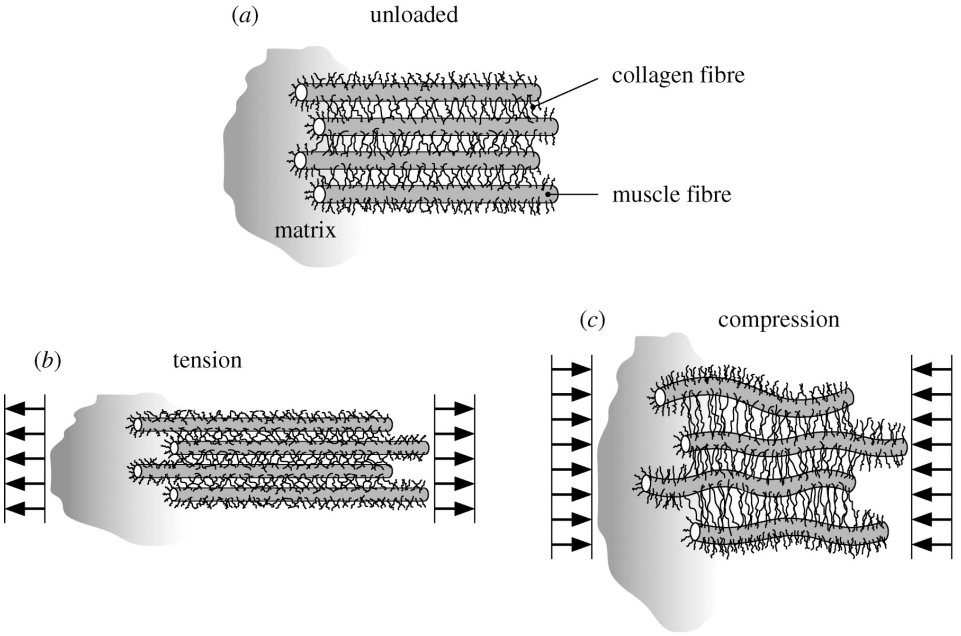


Figure 2.8: Schematic of the arrangement of muscle and collagen fibers and the surrounding matrix. [14]

collagen fibers for the unloaded configuration and both subjected to tension and compression. As seen, when muscle fibers are under tension in the fiber direction, the muscle fibers extend and the inter fiber distance decrease. The collagous network provide relatively little resistance to this deformation. For tension lateral to the fiber direction the

exponential stress behaviour can be attributed to the collagous network. Under compressive load in the fiber direction the muscle fibers buckle and thus stretching the collagous fibers. This stretching is thought to contribute to the relatively large compressive stiffness of the myocardium. The stiffening behaviour in the muscle fiber direction is included as an exponential function of the invariant I_{4f}. In the sheet direction as an exponential function of the invariant I_{4s}. These terms only contribute to the stored energy in tension, where as the contribution in compression is negligible, and thus the terms are only included in the energy function for $I_{4f} > 1$ or $I_{4s} > 1$. The invariant I_{4n} is not included as it, previously shown, depends on I_1, I_{4f} and I_{4s}, and thus the latter three invariant are sufficient to model the tension/compression behaviour. The invariant I_2 is also omitted. To completely capture the shear behaviour from Dokos et al. [5], and thus distiguish between the (fs) and (fn) and between the (sf) and (sn) responses, it is necessary to include an exponential function of the invariant I_{8fs}. This lead Holzapfel and Ogden [14] to propose the energy function

$$\boldsymbol{\Psi} = \frac{a}{2b}\exp[b(I_1-3]+\sum_{i=f,s}\frac{a_i}{2b_i}\{\exp[b_i(I_{4i}-1)^2]-1\}+\frac{a_{fs}}{2b_{fs}}[\exp(b_{fs}I_{8fs}^2)-1], \quad (2.37)$$

where a, b, a_f, a_s, b_f, b_s, a_{fs} and b_{fs} are positive material constants. All a parameters have dimensions stress and b parameters are dimensionless. Using equation 2.30, the Cauchy stress becomes

$$\begin{aligned}\boldsymbol{\sigma} = a\exp[b(I_1-3]\mathbf{B} - p\mathbf{I} + 2a_f(I_{4f}-1)\exp[b_f(I_{4f}-1)^2]\mathbf{f}\otimes\mathbf{f} \\ +2a_s(I_{4s}-1)\exp[b_s(I_{4s}-1)^2]\mathbf{s}\otimes\mathbf{s} + a_{fs}I_{8fs}\exp(b_{fs}I_{8fs}^2)(\mathbf{f}\otimes\mathbf{s}+\mathbf{s}\otimes\mathbf{f}).\end{aligned} \quad (2.38)$$

This constitutive model is at present considered to be the one to most accurately describe the material response of the myocardium and it is on this the full scale finite element model of the left ventricle in this thesis is based.

It is important for ensuring material stability and physical meaningful and unambiguous mechanical behaviour, that the strain-energy function shows convexity. In this context, strict local convexity means that the second-derivative of Ψ with respect to $\mathbf{E}$ is positive definite. In particular, when used for numerical computations, that no desirable instabilities may appear. Some of the earlier mentioned strain energy functions do not all fulfill the convexity requirement, something which the specific model proposed by Holzapfel and Ogden [14] does. [12]

2.4 Models for Active Cardiac Muscles

In this section we briefly outline some of the models proposed to model the active cardiac muscle.

2.4.1 HMT Model of Cardiac Mechanics

The Hunter-McCullock-terKeurs (HMT) model of cardiac mechanics is one of the most notable models intended for use in continuum mechanics. It is built on a fading memory

model of crossbridge kinetics and is developed in the following stages: (i) passive properties of cardiac muscle, (ii) the kinetics of Ca^{2+} binding to troponin-C, (iii) tropomyosin kinetics and (iv) crossbridge kinetics. The passive part used in their model is the earlier reviewed pole-zero law which is not further discussed here. By regarding the free calcium concentration $[Ca^{2+}]_i$ and the muscle fiber extension ratio λ as input to the system, the model can be summarized in four fundamental equations

$$[Ca^{2+}]_b = f_1([Ca^{2+}]_i, [Ca^{2+}]_b, T, T_0), \tag{2.39}$$

$$z = f_2(z, \lambda, [Ca^{2+}]_b, T), \tag{2.40}$$

$$T_0 = f_3(\lambda, z) \text{ and} \tag{2.41}$$

$$T = f_4(T_0, \lambda, t). \tag{2.42}$$

Here f_1 governs the binding kinetics of Ca^{2+} to troponin-C binding sites and is a function of both the concentration of free Ca^{2+} ($[Ca^{2+}]_i$), Ca^{2+} bound at the binding site ($[Ca^{2+}]_b$), as well as the actively developed tension in the muscle fiber (T). Further, f_2 models the tropomyosin kinetics where z is a non-dimensional parameter representing the proportion of actin sites available for cross-bridge binding. λ still denotes the extension ratio (stretch). f_3 governs the relationship between the muscle tension and the myofilament length under steady state conditions. f_4 incorporates the development of tension as a function of time. [16]

2.4.2 A Simplified Approach to Active Cardiac Mechanics

Zulliger et al. [30] presents a constitutive formulation for arterial mechanics including vascular smooth muscle (VSM) tone. They propose the term

$$\Psi_{active} = S_1 S_2 f_{VSM} \Psi_{VSM}, \tag{2.43}$$

where f_{VSM} is the cross-section area fraction of VSM and Ψ_{VSM} is the strain-energy function describing the VSM at maximum contraction. Here S_1 is a non dimensional function describing the level of VSM tone and S_2 governs the relationship between the stretch and the maximum force. S_1 is mathematically expressed through an error function as

$$S_1 = \begin{cases} 0 & \text{, fully relaxed} \\ 1 & \text{, maximum contraction} \\ S_{basal} + (1 - S_{basal})\frac{1}{2}[1 + Erf\,(\frac{Q-\mu}{\sqrt{2}\sigma})] & \text{, normal tone} \end{cases} \tag{2.44}$$

where S_{basal} represent the VSM basal tone contraction, which is the tone when there is no contraction, and Q is a function of the VSM deformation. The function is thus 0 at a fully relaxed state and 1 at maximum contraction with a smooth Gaussian distribution in between, using the error function Erf. Modelling the active muscle in the mitral valve, Skallerud et al. [28] simplified equation (2.44) further by taking the activation as a linear function of time,

$$\sigma_f = \frac{t - t_{start}}{t_{max} - t_{start}} \sigma_{max}, \quad t \in [t_{start}, t_{max}] \tag{2.45}$$

$$\sigma_f = 0, t < t_{start} \quad \text{and} \quad \sigma_f = \sigma_{max}, t > t_{max} \tag{2.46}$$

CHAPTER 3

MODELLING

3.1 Implementation of the Constitutive Model

In order to obtain solutions of nonlinear problems in computational finite elasticity an incremental/iterative solution technique is applied, solving a sequence of linearized problems. The constitutive law presented in equation (2.37) is implemented into Abaqus with the UMAT subroutine using the following form,

$$\begin{aligned}\Psi(\bar{I}_1, \bar{I}_{4_f}, \bar{I}_{4_s}, \bar{I}_{8_{fs}}) = c_{10}(\bar{I}_1 - 3) + \frac{a}{2b}(\exp[b(\bar{I}_1 - 3)] - 1) \\ + \sum_{i=f,s} \frac{a_i}{2b_i}(\exp[b_i(\bar{I}_{4i} - 1)^2] - 1) + \frac{a_{fs}}{2b_{fs}}(\exp[b_{fs}\bar{I}^2_{8_{fs}}] - 1) + \kappa(J-1)^2\end{aligned} \tag{3.1}$$

Here the parameter c_{10} includes a neo-Hookean part which is excluded in this thesis by setting c_{10} = 0 and κ is the positive penalty parameter governing the volumetric change.

We adopt a slightly different notation the previously. [28] Rather than dealing directly with $\mathbf{F}$, we perform a multiplicative decomposition of $\mathbf{F}$ into volume-changing and volume-conserving parts,

$$\mathbf{F} = (J^{1/3}\mathbf{1})\bar{\mathbf{F}} \quad \text{and} \tag{3.2}$$

$$\mathbf{C} = (J^{2/3}\mathbf{1})\bar{\mathbf{C}}, \tag{3.3}$$

where $\bar{\mathbf{F}}$ and $\bar{\mathbf{C}} = \bar{\mathbf{F}}^T\bar{\mathbf{F}}$ are called the modified deformation gradient and the modified right Cauchy-Green tensor. The modified left Cauchy-Green tensor is defined by $\bar{\mathbf{B}} = \bar{\mathbf{F}}\bar{\mathbf{F}}^T$. We also define the following vectors,

$$\bar{\mathbf{f}} = \bar{\mathbf{F}}\mathbf{f}_0 \text{ and } \bar{\mathbf{s}} = \bar{\mathbf{F}}\mathbf{s}_0, \tag{3.4}$$

corresponding to the push-forward of $\mathbf{f}_0$ and $\mathbf{s}_0$ through the volume preserving part of the deformation gradient. Based on this kinematic framework, the strain-energy function can

be presented on a decoupled form,

$$\Psi(\mathbf{C}) = U(J) + \Psi_{iso}(\bar{\mathbf{C}}), \tag{3.5}$$

where U and Ψ_{iso} are the volumetric and isochoric contributions of Ψ, respectively. In equation (3.1), the invariants are defined as

$$\bar{I}_1 = tr\ \bar{\mathbf{C}},\ \bar{I}_{4_f} = \mathbf{f}_0\dot{(}\bar{\mathbf{C}}\mathbf{f}_0),\ \bar{I}_{4_s} = \mathbf{s}_0\dot{(}\bar{\mathbf{C}}\mathbf{s}_0) \text{ and } \bar{I}_{8_{fs}} = \mathbf{f}_0\dot{(}\bar{\mathbf{C}}\mathbf{s}_0) \tag{3.6}$$

The second Piola-Kirchoff stress $\mathbf{S}$ on decoupled fotm, is derived from Ψ thorugh

$$\mathbf{S} = 2\frac{\partial \Psi}{\partial \mathbf{C}} = \mathbf{S}_{vol} + \mathbf{S}_{iso}. \tag{3.7}$$

The Cauchy stress tensor $\boldsymbol{\sigma}$ is then obtained by the push-forward operation of $\mathbf{S}$ to the configuration $\boldsymbol{\sigma} = \frac{1}{J}\mathbf{F}\mathbf{S}\mathbf{F}^T$, giving

$$\begin{aligned} \boldsymbol{\sigma} &= 2\kappa(J-1)\mathbf{1} + \frac{1}{J} dev\ \bar{\boldsymbol{\sigma}}, \\ \bar{\boldsymbol{\sigma}} &= 2\bar{\psi}_1\bar{\mathbf{B}} + 2\bar{\psi}_{4_f}\bar{\mathbf{f}}\otimes\bar{\mathbf{f}} + 2\bar{\psi}_{4_s}\bar{\mathbf{s}}\otimes\bar{\mathbf{s}} + \bar{\psi}_{8_{fs}}(\bar{\mathbf{f}}\otimes\bar{\mathbf{s}} + \bar{\mathbf{s}}\otimes\bar{\mathbf{f}}), \end{aligned} \tag{3.8}$$

where

$$dev\ [\bullet] = (\mathbb{I} - \frac{1}{3}\mathbf{1}\otimes\mathbf{1}) : (\bullet). \tag{3.9}$$

$\mathbb{I}$ denotes the fouth order identity tensor and reads

$$(\mathbb{I})_{ijkl} = \frac{1}{2}(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}) \tag{3.10}$$

In equation (3.8) we have adopted the notation

$$\bar{\psi}_i = \frac{\partial \Psi}{\partial \bar{I}_i}, \quad i = 1, 4_f, 4_s, 8_{fs}. \tag{3.11}$$

We further introduce the definition

$$\mathbb{C} = 2\frac{\partial \Psi}{\partial \mathbf{C}} = \mathbb{C}_{vol} + \mathbb{C}_{iso}, \tag{3.12}$$

where $\mathbb{C}$, in a material description, is the elasticity tensor which measures the change in stress from a change in strain. Combining equations (3.7) and (3.12), we arrive at the relation

$$\mathbb{C} = 4\frac{\partial^2 \mathbf{S}}{\partial \mathbf{C}\partial \mathbf{C}}, \tag{3.13}$$

The spatial description of the elasticity tensor is defined as the push-forward operation on $\mathbb{C}$ as

$$\mathbb{c} = \boldsymbol{\chi}_*(\mathbb{C}), \quad c_{ijkl} = \frac{1}{J}F_{iI}F_{jJ}F_{kK}F_{lL}C_{IJKL}. \tag{3.14}$$

3.2 Material Parameters based on Shear Data

Göktepe et al. [11] identified the material parameters a, b, a_f, b_f, a_s, b_s, a_{fs} and b_{fs} using the data from the simple shear experiments from Dokos et al. [5]. Their results are presented in Table 3.1.

a [kPa]	b [-]	a_f [kPa]	b_f [-]	a_s [kPa]	b_s [-]	a_{fs} [kPa]	b_{fs} [-]
0.496	7.209	15.193	20.417	3.283	11.176	0.662	9.499

Table 3.1: Material Parameters governing the constitutive law. [11]

Holzapfel and Ogden [14] obtained different material parameters using the same data set, and based on this Wang et al. [29] created another set of parameters. Considering that all parameter sets claim to appropriately represent the data, the values in Table 3.1 will be used throughout this analysis. Further, since the invariant I_{4s} and I_{4f} gives the major stress contribution and the parameters governing them, a_s and a_f, are relatively close when comparing the different parameter sets, the model should show similar results in either case. A last material parameters is also needed in the model. The volumetric penalty parameter κ, governing the volumetric contribution in the constitutive law, affects to what degree the materials exhibits volumetric changes during deformations. Given that the myocardium is considered a incompressible material, the penalty parameter $\kappa = 10^5$ MPa. This non-pysiological value is chosen to be much larger than the other material parameters in Table 3.1 and thus ensuring that the material will not exhibit any volumetric change.

3.3 Modelling of Active Contraction

The systolic contraction is modelled by defining the total second Piola-Kirchoff stress tensor $\mathbf{S}$ as the sum of the passive stress tensor $\mathbf{S}_p$ which is derived from the strain energy function and an active component $\mathbf{S}_a$ giving

$$\mathbf{S} = \mathbf{S}_p + \mathbf{S}_a. \tag{3.15}$$

There have been several proposed models to describe the activation of force in the myocardium. A rather complex and phenomenological approach is presented in Hunter et al. [17], where they in detail develop a model explaining the mechanics of the muscle contraction. In this thesis a much more simplified approach is chosen,

$$S_a = \begin{cases} \frac{t}{t_{max}} \times T_{max} & \text{, for } 0 < \mathrm{t} \leq \mathrm{t}_{max} \\ T_{max} & \text{, for } \mathrm{t}_{max} < \mathrm{t} < \mathrm{t}_{endsystole}. \end{cases} \tag{3.16}$$

Here the activation level increases linearly from t = 0 at end-diastole up to a maximum level at t_{max} and is kept there for the remainder of the analysis. t_{max} = 0.13s is chosen

to correspond with the time of maximum endocardial pressure implemented in the analysis. Different levels of activations will by studied to investigate at which level of T_{max} a realistic ejection fraction is achieved.

One could argue that the implementation of a more complex model of the active contraction would lead to a more realistic response in the left ventricle. However, as the constitutive model for the passive part of the material has not earlier been implemented here, a simpler approach is chosen. There are also upsides to this, as by limiting the number of variables in the model, a better understanding of the model can be obtained without the added uncertainty from a more complex active muscle formulation.

It is assumed that all the fibers activate simultaneously and that all fibers exhibit the same contraction behaviour. For the contraction to become realistic it must include features like wall thickening, longitudinal shortening, torsion and radial constriction. To achieve this realism, earlier work has shown that in addition to active stress in the fiber direction, components in the sheet normal direction and shear in the (sn)-plane is necessary Dorri et al. [6]. Only adding components in the fiber direction has not been able to create a realistic deformation pattern. The complete stress tensor implemented in the user subroutine UMAT will thus be on the following form:

$$\mathbf{S}_a = \begin{bmatrix} S_a(f,f) & 0 & 0 \\ 0 & 0 & S_a(s,n) \\ 0 & S_a(s,n) & S_a(n,n) \end{bmatrix} \tag{3.17}$$

It is noted in equation (3.17) that $S_a(s,n)$ = $S_a(n,s)$, preserving symmetry. Further, the level of activation in each direction is of some uncertainty. A study made on rabbit myocardium indicated that S_a(n,n) could be in the range of 20-60% of S_a(f,f) and based on the work of Dorri et al. [6], 60% is chosen as an estimate. S_a(s,n) is chosen to be 3% of S_a(f,f), but this assumption is of even more uncertainty since very little data exists to support it. This gives us the stress components,

$$S_a(n,n) = 0.60 \cdot S_a(f,f) \tag{3.18}$$

$$S_a(s,n) = 0.03 \cdot S_a(f,f) \tag{3.19}$$

The stress component S_a(s,s) is assumed to be negligible. Seeing as the sheet axis has a large component outwards normal to the endo-/epicardial surface (completely normal in the case of sheet angle, α = 0), a stress component in this direction would counteract the wall thickening, and further there is no experimental data indicating any significant component in the (s,s)-direction.

When implementing the active stress contribution into the material law, the Cuachy stress tensor $\boldsymbol{\sigma}$ is additively decomposed by

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}_p + \boldsymbol{\sigma}_a, \tag{3.20}$$

where $\boldsymbol{\sigma}_p$ is the passive part described in equation (3.8) and $\boldsymbol{\sigma}_a$ refers to the active part.

The components of the active stress tensor $\boldsymbol{\sigma}_a$ is found as

$$\sigma_a(f,f) = \frac{1}{J} S_a(f,f)(\bar{\mathbf{f}} \otimes \bar{\mathbf{f}}), \tag{3.21}$$

$$\sigma_a(n,n) = \frac{1}{J} S_a(n,n)(\bar{\mathbf{n}} \otimes \bar{\mathbf{n}}) \text{ and} \tag{3.22}$$

$$\sigma_a(n,s) = \sigma_a(s,n) = \frac{1}{J} S_{active}(s,n) \frac{1}{2}(\bar{\mathbf{s}} \otimes \bar{\mathbf{n}} + \bar{\mathbf{n}} \otimes \bar{\mathbf{s}}), \tag{3.23}$$

for the three different directions. The active stress tensor is implemented in UMAT by adding each of the active stress components to the already existing passive part. It is notable that the fiber direction vectors $\bar{\mathbf{f}}$, $\bar{\mathbf{s}}$ and $\bar{\mathbf{n}}$ are automatically normalized by Abaqus when they are introduced to the UMAT. [1]

3.4 Modelling the Laminar Structure of the Heart

To be able to produce a quantitative analysis of the heart function, the ventricular structure, and hereunder the myocardium architecture, must be properly represented. Legrice et al. [19] presents a mathematical model describing the cardiac micro structure by identifying the three axis of symmetry described and relating them to the ventricular geometry. This is used as a basis for implementing the fiber and sheet orientation in to the finite element model.

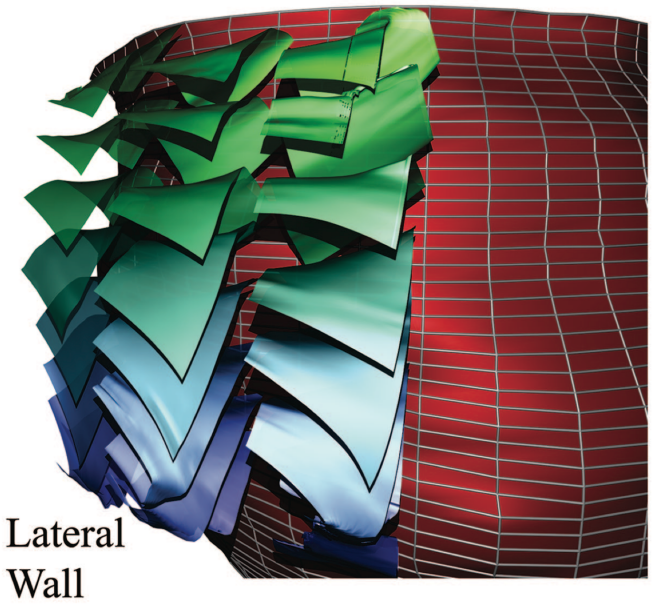


Figure 3.1: Fiber sheets stacked crossing the ventricular wall. [25]

Figure 2.6 gives a visual representation of how the fiber orientation varies transmural in the left ventricle. As seen in sub-figure c) the muscle fiber orientation change through the

wall, from +50° to +70° in the sub-epicardial region to -50° to -70° in the sub-endocardial region with respect to the circumferential direction. [14] The muscle fibers run from the apex to the base in a left-handed direction on the sub-epicardial side and in a right-handed direction on the sub-endocardial side. [22] The fibers in the heart form a 3 dimensional structure due to the arrangement of fibers in sheets. The laminar structure can be visualized as a twisting surface going across the wall and stacked from apex to base. The angle of the sheets is also reported to vary transmural with respect to the radial direction, which is illustrated in Figure 3.1. [25] Both the muscle fiber and the sheet orientation varies from the apical region to the basal region, but for simplicity these variations are excluded in this thesis. Work has also been done implementing DTMRI (Diffusion Tensor Magnetic Resonance Imaging) data in to a finite element model, by among others Dorri et al. [6], to get a more accurate description of the fiber orientation field, but such an approach is outside the scope of this thesis.

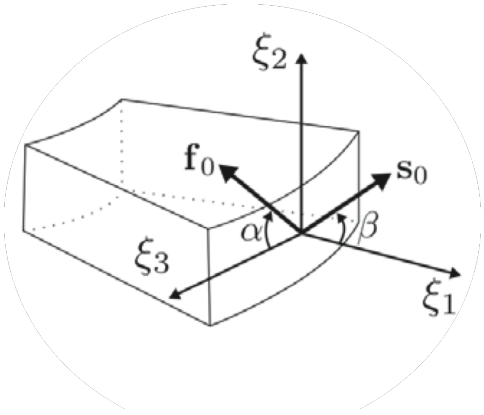


Figure 3.2: Fiber and sheet directions and their respective inclination angles from local element axes. [9]

3.4.1 Implementation of the Fiber Field

For each element of the Abaqus model, as shown in Figure 3.2, a local orthogonal coordinate (ξ_1, ξ_2, ξ_3) system is defined, where ξ_1 is the outward pointing normal to the endo-/epicaridal surface, ξ_3 is tangential to the endo-/epicardial surface, lying in a circumferential direction, and ξ_2 is orthogonal to the two former. This is achieved by arranging the local element numbering, referring to Figure 3.3, such that the 1-2-3-4 and 5-6-7-8 planes are parallel to and facing the epicardial and endocardial surfaces, respectively. Two vectors are created to define each of the planes and the vectors are averaged to find the average plane. The ξ_1 axis is defined as the cross product of the two averaged vectors. The angle ϕ is found as the angle ξ_1 makes with the global X-axis in the XY-plane. The ξ_3 axis is then defined as normal to the ξ_1-axis by rotating ϕ by 90° about the global Z-axis, lying in the XY-plane. Lastly, the ξ_2-axis is defined as the cross product of ξ_1 and ξ_3 axes.

As shown in Figure 3.2 the fiber inclination angle α is defined as the angle between the projection of fiber axis $\mathbf{f}_0$ on the (ξ_2, ξ_3) plane and the ξ_3-axis. Similar, the sheet angle

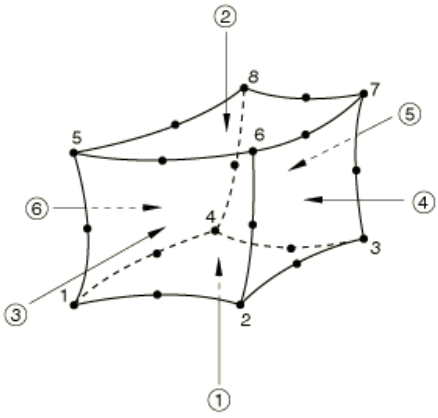


Figure 3.3: Node numbering convention in Abaqus. [1]

β is defined as the angle between the projection of the sheet axis $\mathbf{s}_0$ on the (ξ_1, ξ_2) plane and the ξ_1-axis. This is implemented by first rotating the (ξ_1, ξ_2, ξ_3) system about the ξ_3-axis, creating the sheet angle, and next about the ξ_1-axis creating the fiber angle. In doing this the sheet axis is pulled out of the plane and is consequently no longer normal to the surface. The fiber axis kept tangential to the epicardial surface and although this is not always physically correct, it is consistent with the general fiber direction.

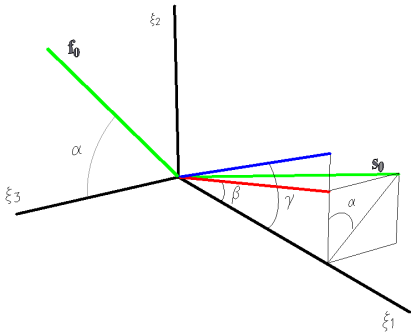


Figure 3.4: Definition of fiber and sheet angles for implementation of fiber field.

The out-of-plane angle (imbrication angle) is considered so small that neglecting it still gives the required accuracy. The choice of keeping $\mathbf{f}_0$ tangential is also justified by the fact that fibers are well represented by a vector. The sheets however, being surfaces, can not be represented with one vector alone and therefore the sheet axis needs only to lie in the sheet plane and thus will be rotated out of the (ξ_1, ξ_2) plane to keep the orthogonality. Because of the rotation of $\mathbf{s}_0$ out of the (ξ_1, ξ_2) plane, the initial rotation of $\mathbf{s}_0$ must be larger than β in order for the sheet angle projection to be correct. The calculation of this angle is found

by identifying the relevant angles in Figure 3.4,

$$\gamma = \arctan\left(\frac{\tan(\beta)}{\cos(\alpha)}\right), \tag{3.24}$$

where γ is the inital rotation of $\mathbf{s}_0$.

In Figure 3.4 the green axis represents the material axes $\mathbf{f}_0$ and $\mathbf{s}_0$, the blue axis is the inital rotation of the sheet axis with an angle γ and red axis is the projection of $\mathbf{s}_0$ on to the (ξ_1, ξ_2) plane. The procedure is implemented in a MATLAB script which is presented in Appendix B.

3.5 The Truncated Ellipsoid Model

In this section the different aspects surrounding the creation of the finite element model in Abaqus is presented.

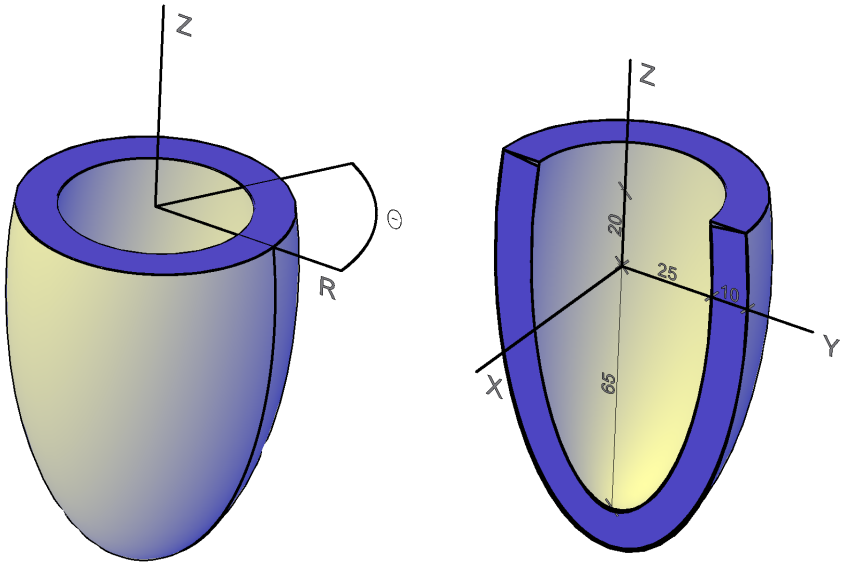
3.5.1 Left Ventricular Geometry

An idealized and simplified geometry is used to model the left ventricle and an ellipsoid truncated at the base is chosen. This approach has become the convention when modelling the heart and among others Eriksson et al. [9], Göktepe et al. [11] and Remme and Hunter [24] use variations of this geometry. As the objective of this thesis is to investigate the deformation of the human left ventricle in systole, the initial geometry is chosen to approximate the geometrical parameters of the end-diastolic geometry in a physiological heart. The geometric parameters are presented in Table 3.2 and Figure 3.5 depicts a cross section of the model geometry. In this thesis neither the mitral valve or the aortic valve are included in the left ventricular geometry. As they play an important part during the ventricular cycle, their inclusion is a natural continuation of this work.

	Theoretical Values	Model Values
Volume	120ml	123ml
Wall thickness	7-11mm	10mm
Internal diameter	25-56mm	50mm
Internal longitudinal length	-	85mm

Table 3.2: Left ventricle end-diastole geometry parameters. Theoretical taken from Levick [20].

The internal diameter of the model is valid for the equatorial region. The internal volume of the model is also dependent on the mesh size and the element type. For the majority of the analysis a linear cube element is used and since the geometry then is linearized, a finer mesh will produce a larger internal volume then a course. The volume of the model is however approximately 123ml for all mesh sizes. The wall thickness in the model is kept at a constant 10mm in all regions. Compared with a physiological heart this simplification. The septal wall is in general thinner than the rest of the ventricle and also the apex is shown to have a different thickness. The longitudinal length was chosen to give a volume of approximately 120ml, but is still with in the physiological range.



(a) Sketch of complete geometry with cylindrical coordinate system used to define boundary conditions.

(b) Cross-section of geometry with global Cartesian coordinate system.

Figure 3.5: Geometry of the truncated ellipsoid model.

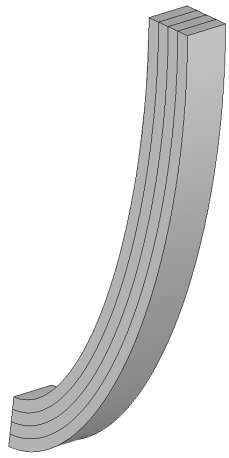


Figure 3.6: Schematic of transmural partitions on a slice of the ventricular wall.

The myocardial structure is modelled with a linear transmural variation of the fiber and sheet angle. To accommodate this in the model, the geometry is partitioned into layers of equal thickness, ensuring a correct representation of the fiber field. This is visually displayed in Figure 3.6. It is apparent that the number of layers may affect the result of the analysis. A course subdivision will yield a more discrete fiber distribution where as more layers yields a more uniform distribution, which must be considered as more physically correct. However, more layers consequently produces a finer mesh, which will affect the computational cost. A study will therefore be performed to investigate the affect of number of layers.

3.5.2 Boundary Conditions

Modelling the boundary conditions for biological soft tissue is somewhat challenging as the boundaries are not very well defined. Qualitative assumptions and approximations must therefore be carried out.

In this model every node on the basal surface has been restricted in the Z direction and thus the nodes are free to deform in the radial and Θ direction in an cylindrical coordinate system, shown in Figure 3.5a. An initial study showed that by restricting nodal displacement all together, a very large deformation gradient localized in the basal region and consequently also unnaturally high stresses occurred in the region. It is here noted, that if the model is not symmetrical, additional boundary conditions should be added. By constraining the apical nodes at X = Y = 0 from radial movement, the model becomes for secure against any rigid body movements. As this was not strictly needed here, the nodes are kept unrestrained.

Further to account for the tissue surrounding the epicardial surface, Göktepe et al. [11] applied linear springs with stiffness k_x=k_y=10^{-3} N/mm to every node on the epicardial surface. There is very little data regarding how the pericardium and other surrounding tissue affects the myocardial deformation. During contraction, given the model works as intended, the spring stiffness will also counteract the contraction, which is not necessarily correct. It is preferable not to add any extra uncertainty to the system and based on the lacking in supporting data, springs are not added to the model.

In systole the contraction creates a intracavital blood pressure acting on the endocardial surface. The pressure is a function of both time and location on the surface, but as the spatial distribution would have had to be implemented using computational fluid dynamics, which is outside the scope of this thesis, this is omitted. The systolic pressure is therefore implemented as a uniform pressure on the endocaridal suface with the function

$$P_{LV} = -944t^2 + 245t, 0 \leq t \leq 0.2s, \tag{3.25}$$

depicted in Figure 3.7.

This function in equation (3.25) has P_{max} = 16.0kPa at t = 0.13s which is in agreement with the normal systolic blood pressure for a healthy heart 120mmHg. [20] [6]. The phase prior to P_{max} is equivalent to the rapid ejection phase of the cardiac cycle. After P_{max} the pressure drops simulating the reduced ejection phase. Further the pressure from the right ventricle blood pressure acting on the septal region has been neglected. The inclusion of these forces will affect both the stress and strain distribution and the deformation pattern in the left ventricle, but this is not studied any further.

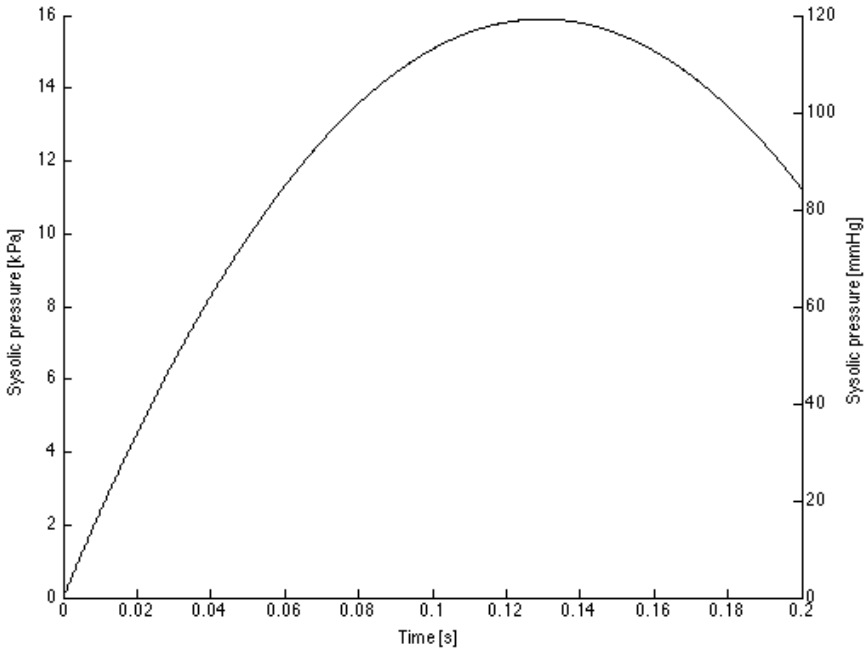


Figure 3.7: Sysolic pressure on endocardial surface.

In a real heart the active stress tensor will not keep constant level after the maximum pressure is reached, as is assumed in this model. However, since the blood pressure is included as an internal pressure, as opposed to through fluid dynamics, the decreasing pressure during the reduced ejection phase is serving as the reduction of active tension in the myocytes. [24]

3.5.3 Element Type

For every finite element analysis an appropriate element must be chosen to obtain the necessary accuracy, but also provide good convergence at a acceptable computational cost. In this work the eight node brick hybrid element C3D8H is used. Triangular and tetrahedral elements is convenient when meshing a complex geometry, but is known to be overly stiff and requiring a very fine mesh to provide accuracy. The first-order version of these elements are also known to exhibit volumetric locking when used in incompressible problems, hence they are considered inappropriate here. Further, quadrilateral elements, when provided with a good mesh provide equal accuracy at less computational cost.

When a material is incompressible, or nearly incompressible, very small displacements may cause very large changes in pressure, since the bulk modulus is much larger than the shear modulus. A displacement based solution may therefore cause numerical difficulties. When using a hybrid formulation the pressure is treated as a independent interpolated solu-

tion variable which solves any problems regarding volumetric locking. The elements, having more internal variables, are though computationally more expensive than non-hybrid formulations.

The Abaqus element C3D8H uses a first order element using a full integration scheme. Second-order elements with reduced integration can also be a good choice for this problem. Second-order elements are known to give a higher accuracy given the problem does not have severe element distortion. They are also better at modelling geometries as they can use fewer elements on a curved surface. A study using the element C3D20HR will be performed to assess any advantages of using second order elements. [1]

3.5.4 Damping Factor

A known issue with non-linear problems are instabilities. Specifically for the case of a hyperelastic material exhibiting large deformations, as is the case for the myocardium, local instabilities may occur and cause numerical problems. In the initial work with the model, the analysis exhibited convergence problems that may have been linked to some instability. To address this issue a constant damping factor with magnitude 10^{-8} was included to stabilize the problem. By doing this Abaqus includes a damping force in the global equilibrium equations. To ensure that the analysis still convergence towards the correct solution, it is important to compare the static dissipated energy with internal elastic energy. For all of the analysis this fraction was of the order 10^{-4}, which indicates that the added stabilization does not negatively affect the solution. [1]

3.5.5 Extraction of Results

This section describes the procedure for extracting the different results from the model. This is done to be able to better judge the validity of the results.

Ejection Fraction

The ejection fraction is calculated using the nodal coordinates of the nodes defining the endocardial surface at the end-diastole and end-systole configurations. The volumes of the two states are then calculated using the convhull function in MATLAB. This function returns the volume of a convex hull of a set of point in a 3 dimensional space. The procedure using MATLAB and Python scrips are found in Appendix D.

Torsion

The myocardial torsion of the Abaqus model is found by tracking the movement of the mesh between the end-diastole and end-systole configuration. It is noted that the values found are greatly affected by at which point on the geometry they are calculated. The rotation of the base is calculated from the four points on the epicardial surface and located on the global X=0 and Y=0 in the undeformed configuration. The rotation of the apex is calculated from the four nodes closest to the apex lying on the global X=0 and Y=0 in the undeformed configuration. The rotation is then calculated as the angle between the between the lines of the mesh at the start and end of the analysis. This gives four angles

for both apex and base, which are averaged to find the mean rotation. The torsion is found as the difference between the apex and base rotations. The rotation is found as about the global longitudinal axis and any movement in the global XY-plane of the apex point is accounted for. The procedure combined for calculating the torsion, using both Python and MATLAB scripts are found in Appendix F.

Wall Thickening

The wall thickening is found both at the apex and the equatorial region. The apex the wall thickening is found as the change of distance between the nodes on the endo and epicardial surfaces for global X=Y=0. At the equator the wall thickening is found as the change of distance between the nodes at Y=0 on the two surfaces. For the equator, only movement in the XY-plane is considered as the movement in the Z-direction must be considered to be a function of other mechanics than wall thickening. As the model is symmetric the wall thickening will be approximately equal for the same global Z-value and a single point on the equator will therefor be sufficient. For different positions along the longitudinal axis the wall thickness will however change.

Longitudinal and Radial Shortening

The change in relative longitudinal distance is found as the change in the distance between the node at the endocardial apex and the base, along the long axis. The change in radial distance is found as the change in the distance between the node on at the endocardial surface and Y=0, and the long axis.

CHAPTER 4

RESULTS

4.1 Validation of the Constitutive Model

The structural based constitutive model for passive myocardium is implemented in the Abaqus model using the user subroutine UMAT. The use of user subroutines allows for modelling with more specializes material models, but initial testing on single element models is required to confirm that expected behaviour is achieved. The UMAT used in this thesis was developed and provided by Assoc. Prof. Victorien Prot and is presented in Appendix C. Since the UMAT had not previously been used in a full scale model of the left ventricle, an initial validation analysis was performed. As a basis for this validation, the results from the shear tests by Dokos et al. [5] were used and recreated. The analysis

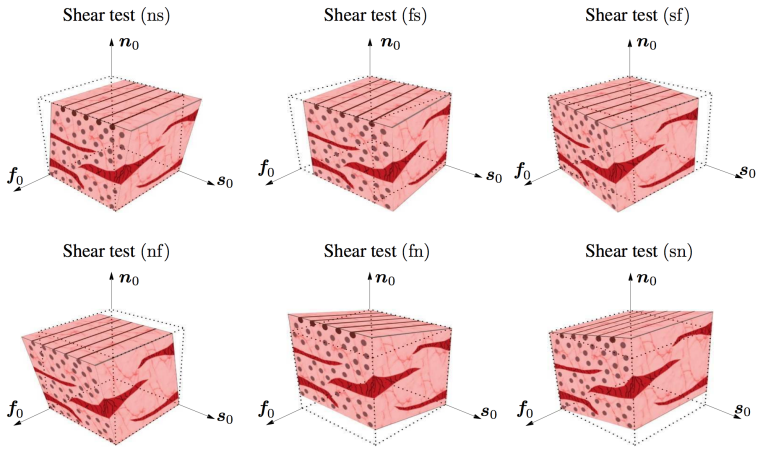


Figure 4.1: Sketches of the six modes of simple shear for myocardium with respect to the local material axis ($\mathbf{f}_0$, $\mathbf{s}_0$, $\mathbf{n}_0$) [11].

was performed by creating a 3x3x3mm cube and using one C3D8H element. For each of the six different shear modes, a displacement equivalent to simple shear $\gamma = 0.5$ was prescribed, consistent with that shown in Figure 4.1. The material parameters presented in Table 3.1 in combination with the implemented constitutive model represents the passive myocardium material.

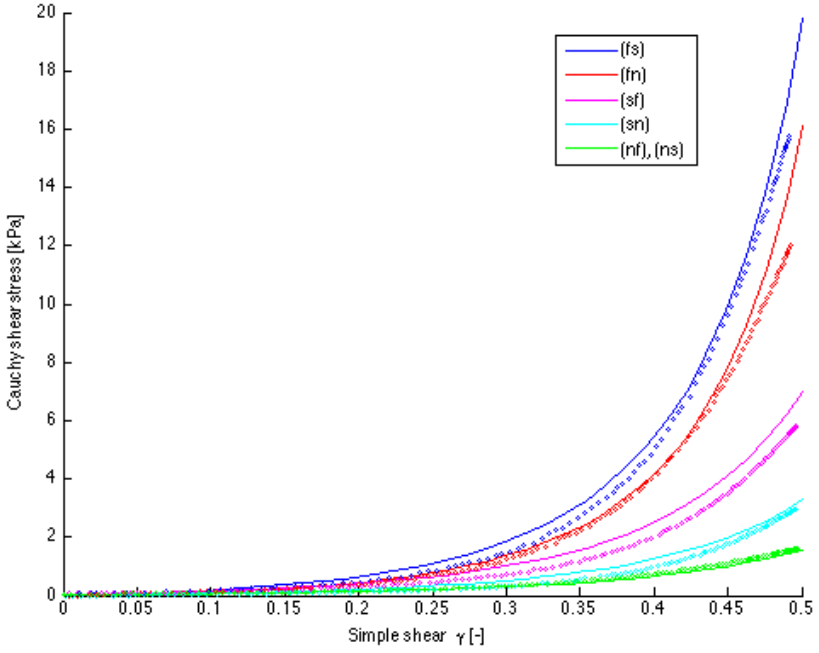


Figure 4.2: Comparison of simple shear experiments. Abaqus results (lines) and experimental data from Dokos et al. [5] (circles).

We observe in Figure 4.2 that the results of the Abaqus analysis agrees reasonably with the data from the shear test, however not to the degree of accuracy reported in Göktepe et al. [11]. Especially the result for (sf) shear gives slightly more resistance than the experimental results. The numerical result does predict the expected 5 distinguishable shear responses. This agrees with the number of invariant implemented in the model as the present research has not been able to find any significant difference in the (nf) and (ns) response. Further, the relative degree of stiffness in the different directions also is correct, with $F > S > N$, and thus the material implemented shows the wanted behaviour.

4.2 Validation of Active Stress Implementation

The active stress component was implemented into the finite element model thorough the UMAT user subroutine. Here the current activation level is found as a function of time in

the analysis and the different stress components is combined with the stress calculations for the passive response of the myocardium. The entire subroutine is reviewable in Appendix C. To validated correct behaviour a simple cube model of myocardium tissue was created. Here the local material axis is aligned with the global coordinate system. The cube has the boundary conditions: suppressed displacement in X direction for all X=0, displacement in Y direction for all Y=0 and displacement in Z direction for all Z=0, thus restricting any rigid body motion, but to allowing free deformation.

Considering the boundary conditions and the stress activation, it is instructive to view the setup as biaxial where the deformation is defined by

$$x_1 = \lambda_f X_1, \quad x_2 = \lambda_s X_2 \quad \text{and} \quad x_3 = \lambda_n X_3, \tag{4.1}$$

where λ_f, λ_s and λ_n are the principal stretches in the three material directions. Given the incompressible material, equation (4.1) must satisfy the condition

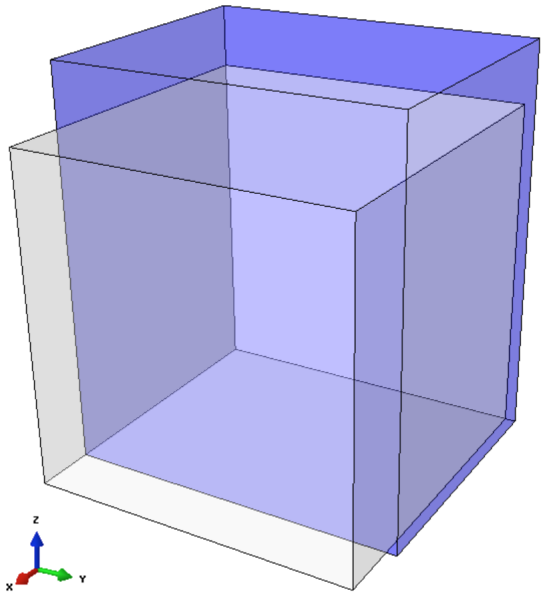
$$\lambda_f \lambda_s \lambda_n = 1 \tag{4.2}$$

and following the volume is conserved during deformation.

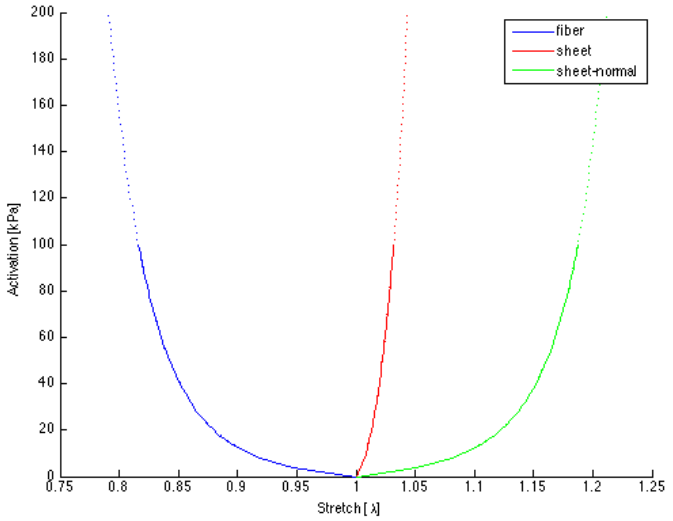
With activation only in the fiber direction, Figure 4.3, the cube exhibits a contraction in the fiber direction graphically represented as $\lambda_f < 1$. Consequently we then have $\lambda_s > 1$ and $\lambda_n > 1$. Further the stretch is larger in the $\mathbf{n}$ than $\mathbf{s}$ direction, consistent with the myocardium being less stiff in the $\mathbf{n}$ direction.

In Figure 4.4, a second stress component has been added in the sheet normal direction. Notably the contraction in the sheet normal direction is significantly larger than the fiber contraction. The lesser stiffness in this direction, than in the fiber direction, results in a relative large contraction although the stress is only 60% of that in the fiber direction. How this affects the behaviour of the full scale left ventricular model is to be investigated.

In Figure 4.5, a shear stress component, 3% of the fiber component, in the (ns) plane has been introduced. The inclusion of a shear component is evident in a much larger stretch in the sheet direction. It is clear that the contraction in the sheet normal direction is here more restricted which yields larger stretch in sheet direction to preserve the volume. While the two first cases has a stress free equilibrium, this is not the case here. However, for the purpose of comparing with the two former cases, and in order to get a better understanding of how the shear component affects the deformation, symmetry of the cube was kept. When not restricted the shear stress gave some shear deformation, which is to be expected. [9]

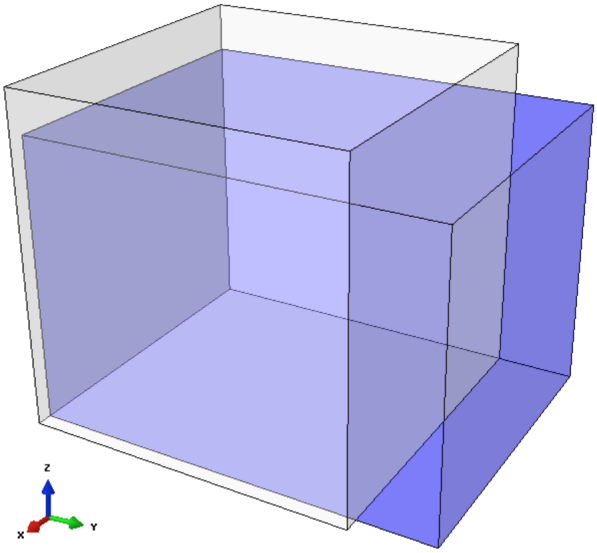


(a) Deformation of cube under activation. White and blue represents undeformed and deformed state with T_{max}=100kPa, respectively.

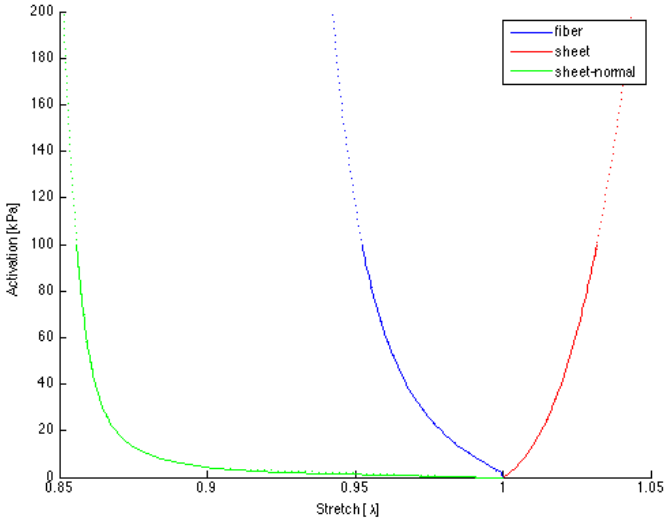


(b) Stretch as function of activation level. 100kPa and 200kPa illustrated by solid and dotted lines, respectively.

Figure 4.3: Response of cube with activation in fiber direction. Global (X, Y, Z) is equivalent to material ($\mathbf{f}_0$, $\mathbf{s}_0$, $\mathbf{n}_0$) system.

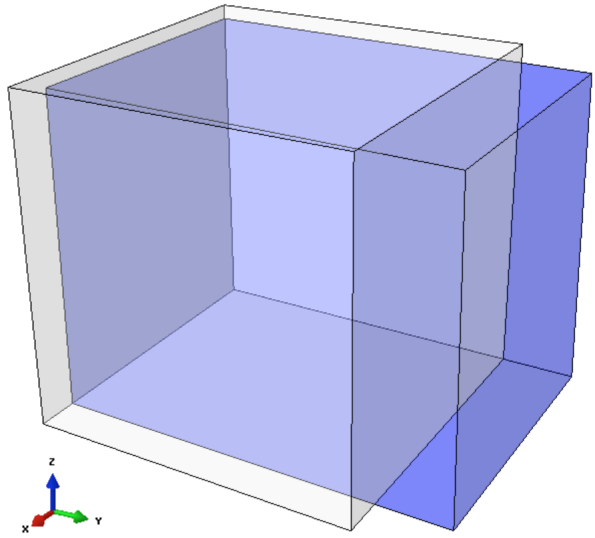


(a) Deformation of cube under activation. White and blue represents undeformed and deformed state T_{max}=100kPa, respectively.

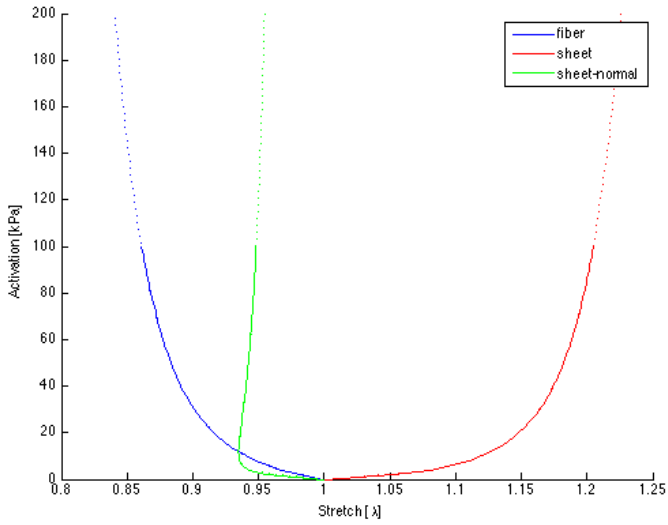


(b) Stretch as function of activation level. Analysis with 100kPa and 200kPa illustrated by solid and dotted lines, respectively.

Figure 4.4: Response of cube with activation in fiber and sheet normal direction. Global (X, Y, Z) system is equivalent to material ($\mathbf{f}_0$, $\mathbf{s}_0$, $\mathbf{n}_0$) system.



(a) Deformation of cube under activation. White and blue represents undeformed and deformed state T_{max}=100kPa, respectively.



(b) Stretch as function of activation level. Analysis with 100kPa and 200kPa illustrated by solid and dotted lines, respectively.

Figure 4.5: Response of cube with activation in fiber, sheet normal and shear (sn) direction. Global (X, Y, Z) system is equivalent to material ($\mathbf{f}_0$, $\mathbf{s}_0$, $\mathbf{n}_0$) system.

4.3 The Truncated Ellipsoid Model

To see how the truncated ellipsoid model behaves compared to a physiological heart, we choose a set of parameters describing the systolic deformation. These are the ejection fraction, myocardial torsion, wall thickening, longitudinal shortening and radial shortening. The physiological values are summarized in Table 4.1

EF [%]	AMT [°]	$\frac{\Delta h}{h}$ [%]	$\frac{\Delta L}{L}$ [%]	$\frac{\Delta R}{R}$ [%]
67	9-12	52	17-25	31-39

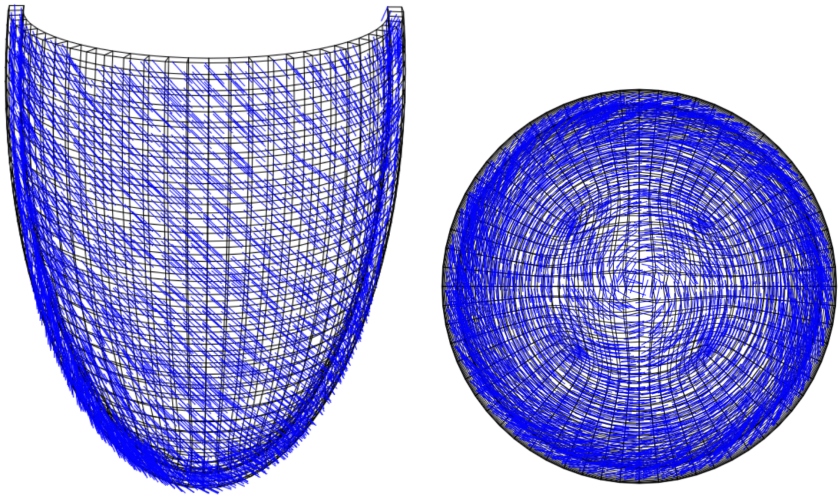
Table 4.1: Theoretical values describing systolic heart deformations. Values are taken from Levick [20] and Dumesnil and Shoucri [7]

In Table 4.1, both the longitudinal shortening $\frac{\Delta L}{L}$ and the radial shortening $\frac{\Delta R}{R}$ is measured as internal values in the left ventricle (hence from the endocardium). Dumesnil and Shoucri [7] does not specify where the internal radius is measured, however, as it is a relative value, it is still appropriate to use.

4.3.1 Validation of the Fiber Field Implementation

Figures 4.6 and 4.7 show the result of the implementation of the fiber field in Abaqus. The figures show the outermost layer of the model with fiber angle $\alpha = 45°$ and sheet angle $\beta = 0°$. We can clearly see the at the fiber axis form a left handed helix from the apex to the base and that the fibers are aligned with the circumferential direction. The sheet axis is confirmed as normal to the surface and is pointing in an outwards direction. The figures confirms that the implementation of the fiber field is correct.

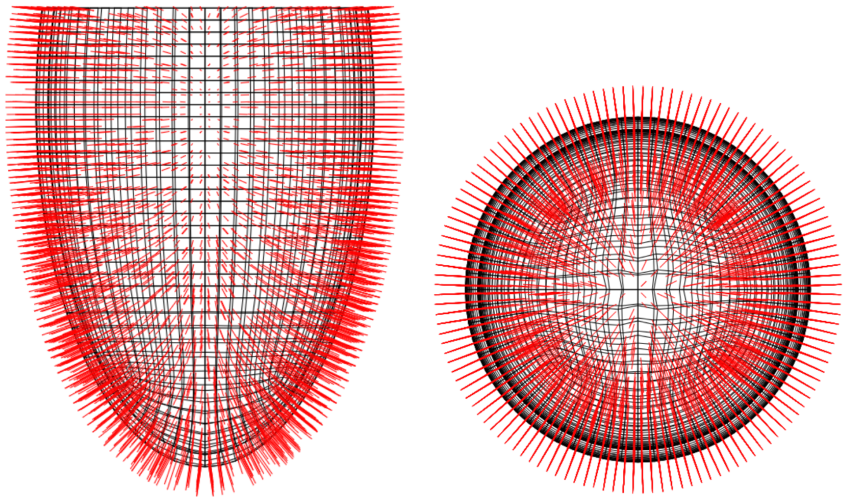
Figure 4.8 show the transmural variation of the fiber angle using $\alpha = 45°$. The fibers are varying from a left handed to a right handed helix when moving from the sub-epicardium to the sub-endocardium, which is in agreement with the literature. In Figure 4.9, the variation of the sheet angles for the different layers are shown for sheet angle $\beta = 45°$. It is also visible, when comparing with Figure 4.7a, that the sheet axis is no longer normal to the surfaces, which is as expected.



(a) Material direction $\mathbf{f}_0$ in the longitudinal direction.

(b) Material direction $\mathbf{f}_0$ in the global XY plane.

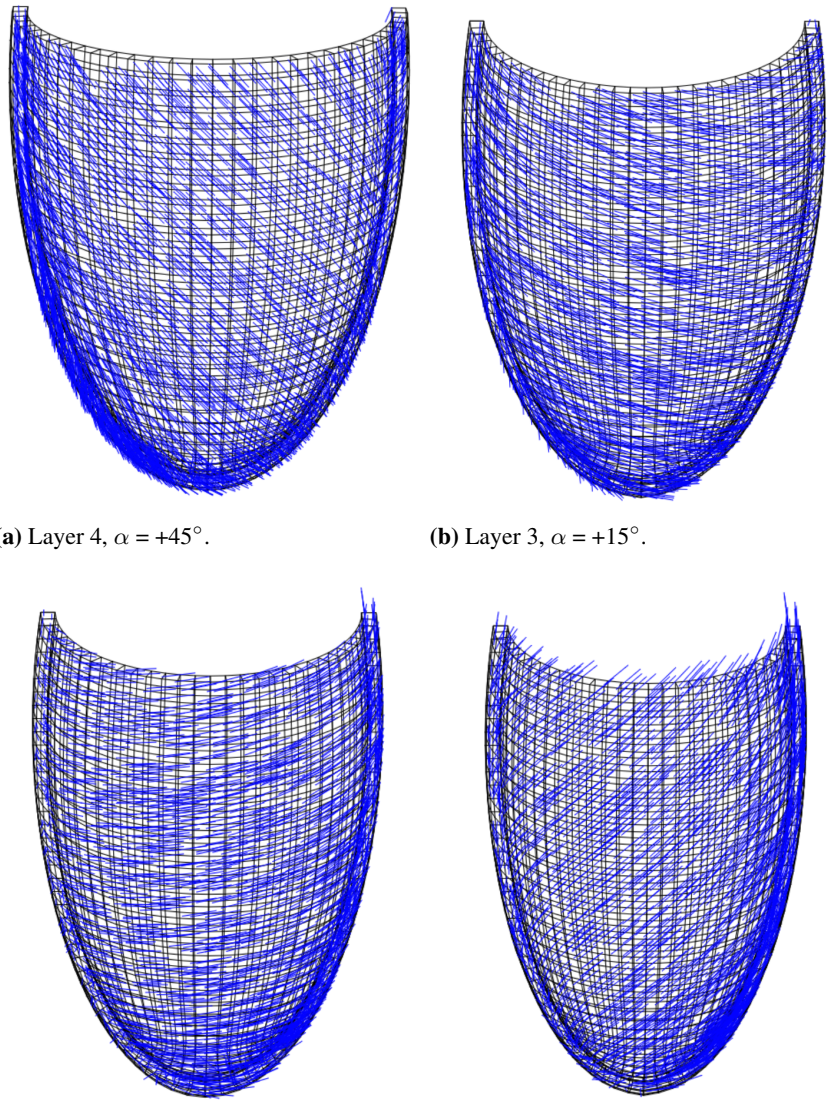
Figure 4.6: The local material $\mathbf{f}_0$ axis implemented in Abaqus with fiber angle $\alpha = 45°$. Figures show the outermost layer in the model.



(a) Material direction $\mathbf{s}_0$ in the global XZ plane.

(b) Material direction $\mathbf{s}_0$ in the global XY plane.

Figure 4.7: The local material $\mathbf{s}_0$ axis implemented in Abaqus with sheet angle $\beta = 0°$. Figures show the outermost layer in the model.



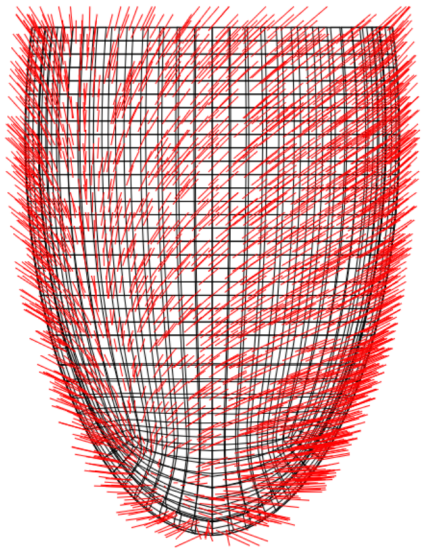
(a) Layer 4, α = +45°.

(b) Layer 3, α = +15°.

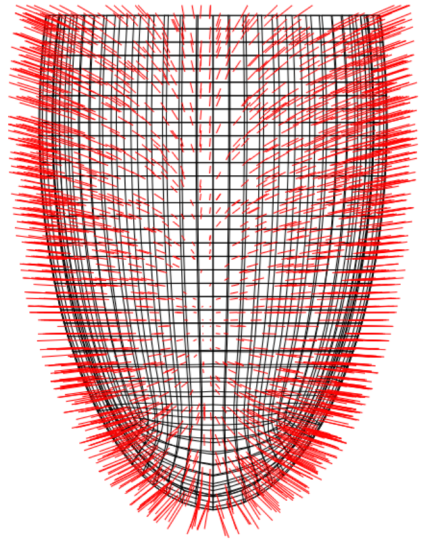
(c) Layer 2, α = -15°.

(d) Layer 1, α = -45°.

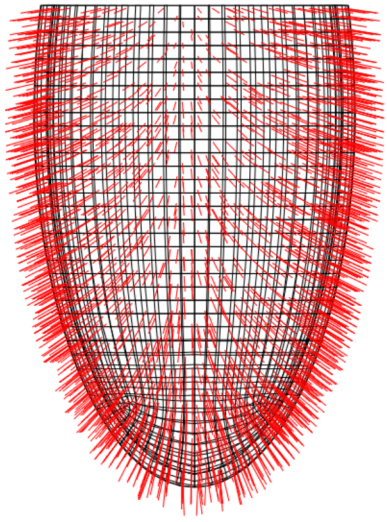
Figure 4.8: The local material $\mathbf{f}_0$ axis with fiber angle $\alpha = \pm 45°$, shown for the different layers through the left ventricular wall.



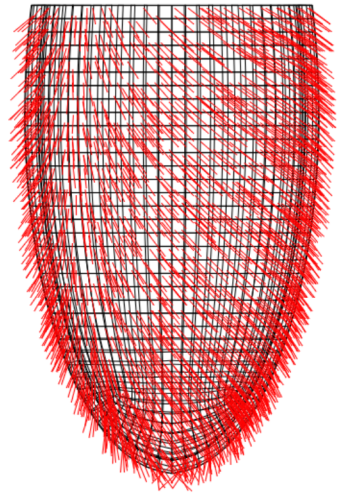
(a) Layer 4, β = +45°.



(b) Layer 3, α = +15°.



(c) Layer 2, α = -15°.



(d) Layer 1, α = -45°.

Figure 4.9: The local material $\mathbf{s}_0$ axis with sheet angle $\beta = \pm 45°$, shown for the different layers through the left ventricular wall.

4.3.2 Mesh Study

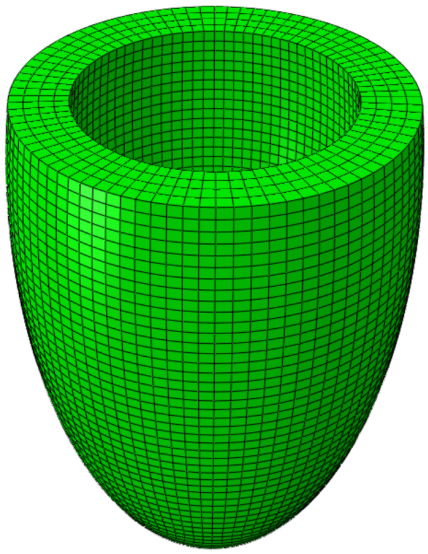
The mesh study was done on a model with fiber angle α = 45°, sheet angle β = 0° and a maximum activation T_{max} = 150kPa in (ff), (nn) and shear (sn) directions. Different combinations of angles and activations has shown different analysis time, this gives some uncertainty to how a model with different fiber angles may be affected by the mesh refinement. The mesh study using α = 45° will however serve as an indication of at what mesh size sufficient accuracy is reached. Further, since the models behaviour is also dependent on how many layers the ventricular wall is divided into, four layers will be used throughout the mesh study. A separate study of the effect of the number of layers is presented in the following section. Convergence will be checked for the parameters ejection fraction, torsion, longitudinal shortening, radial shortening and wall thickening.

N_{els} [°]	EF [%]	AMT [°]	$\frac{\Delta h}{h}_{equator}$ [%]	$\frac{\Delta h}{h}_{apex}$ [%]	$\frac{\Delta L}{L}$ [%]	$\frac{\Delta R}{R}$ [%]	CPU time [s]
15192	31.1	30.8	17.6	19.9	8.9	13.0	7712
8960	31.1	27.2	17.6	19.8	8.9	12.9	4888
2552	31.0	20.2	17.6	19.7	9.0	12.9	7502
1296	31.0	17.8	17.5	20.3	9.1	12.9	1592

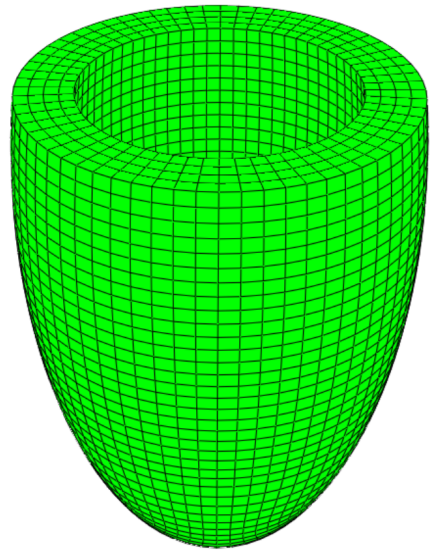
Table 4.2: Results of mesh study, with α = 45°, β = 0° and T_{max} = 150kPa with activation in (ff), (nn) and (sn) directions. N_{els} is the number of elements in the model.

Referring to Table 4.2, it is noticeable that the ejection fraction is almost equal for the different meshes. This also applies for wall thickening at equator, and longitudinal and radial shortening, where only minor changes are seen. A larger difference is observed in the wall thickening at the apex. Here the mesh using 1296 element gives a somewhat different result. Near the apex is also where we have largest curvature and it is clear that such a large mesh is not able to accurately describe this region. An important detail to bare in mind when evaluating the torsion results for the different meshes in Table 4.2, is how the torsion is calculated. The apex rotation is found by averaging the rotation of the 4 nodes closest to the apex and thus as the mesh changes, the distance from the nodes to the apex changes. Therefore the rotation is found at points in different locations on the geometry for the different meshes. A different approach, which perhaps would be better for this study, is to sample points at a specific geometric location. This would however not give the maximum rotation of the apex. Seeing as linear elements are used, the chosen method provides the absolute torsion at the apex and this approach seems consistent. A larger mesh produces a significant smaller torsion, but the different sampling points limits our ability to draw certain conclusions from this. The difference is however so large that the accuracy of the most course mesh at the apical region is considered insufficient.

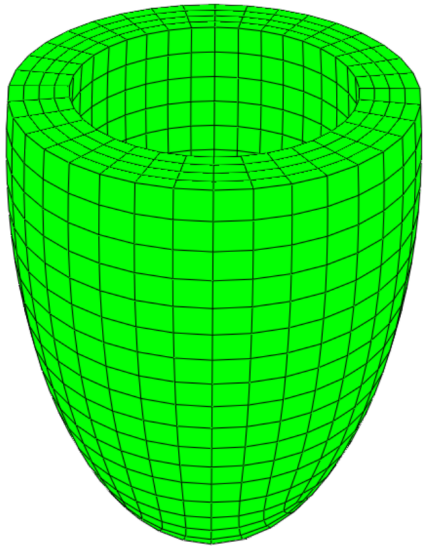
Apart from what the convergence of the chosen parameters shows. A more qualitative study of the mesh is appropriate. Near the apex, the curvature is quite large and thus the region requires a finer mesh than the rest of the geometry. Using a structured meshing technique, Abaqus insists on using two elements per partition in the radial direction.



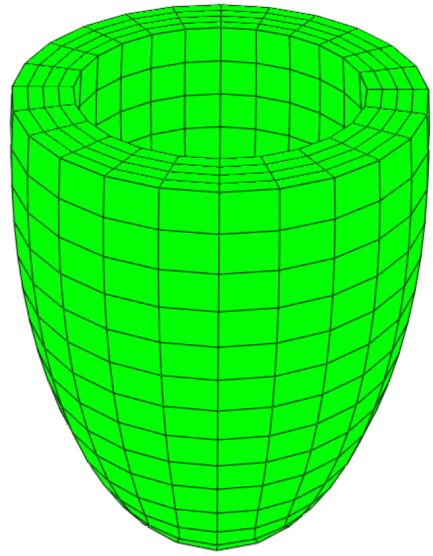
(a) Mesh with approximate size 2.0 mm, giving 15192 elements.



(b) Mesh with approximate size 2.5 mm, giving 8960 elements.



(c) Mesh with approximate size 5 mm, giving 2552 elements.



(d) Mesh with approximate size 7.5 mm, giving 1296 elements.

Figure 4.10: Different meshes used in the mesh study, in undeformed state.

Therefore, the thickness of each layers is not strictly forced to be equal. It is visually observable that when a courser mesh is used, the difference in element size, in the radial direction, is larger than for finer meshes. The change in fiber angles then no longer has a linear variation, as is assumed, and the contribution from the different fiber angles in the layers will therefore be different. This adds to the uncertainty of the accuracy of the apical deformation and highlights the need for a finer mesh in this region.

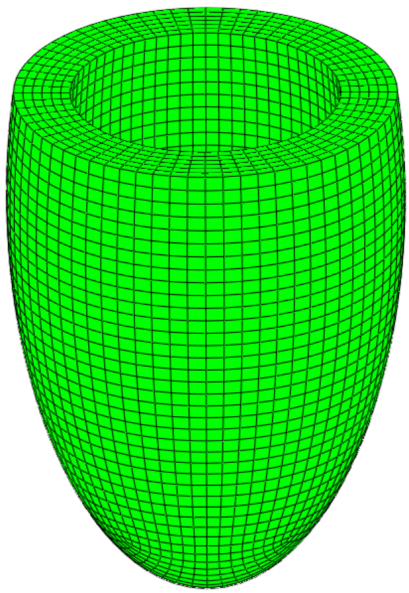
When looking at the total CPU time for the different meshes, some irregular numbers is encountered. For the analysis using 2552 element the analysis time is higher than those using a more refined mesh. A possible reason for this is that the analysis has encounter some sort of numerical issue, which has slowed the analysis. A typical feature of all the analysis in this thesis, is that Abaqus tends to use very small time increments, in the order 10^{-7}. This can be attributed to the exponential change of stiffness seen in the material, when smaller time steps are need. The analysis uses relatively longer using 15192 elements than 8960 elements, and there is no evidence that the results are significantly more accurate.

Based on the results, both quantitative and qualitative, it is clear that a mesh using 8960 elements is necessary to give enough accuracy in the apical region.

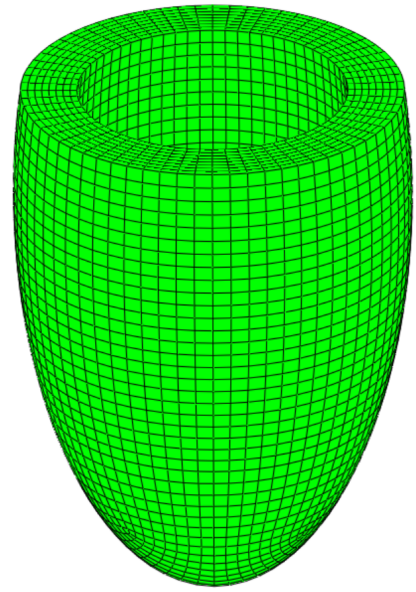
4.3.3 Study of Number of Layers

Using an otherwise equal model setup as in the previous section, a study of how different number of layers affect the results is performed. It is a reasonable assumption that a finer transmural partitioning gives a better representation of the fiber field. Increasing the number of layers will produce a more continuous change in the fiber angles and can be considered to give a more accurate implementation of the rule based fiber field. However, the increasing finer partitioning adds to the number of elements and then increasing the computational cost. As noted, when using cube elements and a structured mesh, Abaqus insists on assigning minimum two elements in the radial direction per partition thus significantly increasing the number of elements. This also makes it not so straight forward to compare the results based on the fiber distribution alone, as a finer mesh generally will increase the accuracy. One approach to counter this would be to increase the global element size to get approximately the same number of elements in the different cases. However, as shown in the results of the mesh study, larger elements compromises the accuracy in the apical region making comparison difficult. The global mesh size is therefore kept equal at 2.5mm and thus creating a similar mesh pattern in the circumferential direction, as shown in Figure 4.11. The added number of elements must of course be accounted for when analysis the results.

A trend which is apparent when looking at Table 4.3 is that for an increasing number of layers the model exhibit a decrease in longitudinal shortening and an increase in radial shortening. A quick geometric consideration is done by splitting unit fibers into circumferential and longitudinal components. The average components for a transmural cross-section is found as $\frac{1}{N_{layers}} \sum_{i=1}^{N_{layers}} \cos(\alpha_i)$ for the circuferetial direction and



(a) Mesh with 6 layers, giving 15192 elements.



(b) Mesh with 8 layers, giving 19984 elements.

Figure 4.11: Meshes with 6 and 8 layers, in undeformed state.

$\frac{1}{N_{layers}} \sum_{i=1}^{N_{layers}} \sin(\alpha_i)$ for the longitudinal direction, giving

$$(4layers)^{circum}: \quad \frac{1}{4} \cdot 2 \cdot (\cos(45°) + \cos(5°)) = 0.837$$

$$(4layers)^{long}: \quad \frac{1}{4} \cdot 2 \cdot (\sin(45°) + \sin(5°)) = 0.483$$

$$(6layers)^{circum}: \quad \frac{1}{6} \cdot 2 \cdot (\cos(45°) + \cos(27°) + \cos(9°)) = 0.862$$

$$(6layers)^{long}: \quad \frac{1}{6} \cdot 2 \cdot (\sin(45°) + \sin(27°) + \sin(9°)) = 0.440$$

Interpreting the different components we see that a increasing number of layers gives a larger circumferential component and smaller longitudinal component, which explains the global behaviour. The ejection fraction is increasing for an increase in number of layers. This indicates that the radial shortening and wall thickening plays a relatively more important part in creating the ejection fraction than the longitudinal shortening. Further, it is interesting that the torsion is decreasing for an increasing number of layers. The mesh pattern on the surface is approximately equal between the different meshes and thus the rotation at the apex can be compared with confidence. It is clear that a higher number of layers to a lesser degree is overestimating the myocardial left ventricular torsion.

The analysis time is approximately 5 times higher when using 8 layer compared with 4 layers. Individually the CPU time of 26892 seconds for the 8 layer analysis is not so

N_{layers} [ ]	N_{els} [ ]	EF [%]	AMT [°]	$\frac{\Delta h}{h}_{equator}$ [%]	$\frac{\Delta h}{h}_{apex}$ [%]	$\frac{\Delta L}{L}$ [%]	$\frac{\Delta R}{R}$ [%]	CPU time [s]
4	8960	31.1	27.2	17.6	19.8	8.9	12.9	4888
6	15192	31.3	25.1	17.9	20.5	8.0	13.6	7295
8	19984	31.5	22.6	18.2	21.3	7.7	14.0	26892

Table 4.3: Results from study of number of transmural layers. α = 45°, T_{max} = 150kPa in (ff), (nn) and (sn) directions.

much, but as previously mentioned the analysis time is much larger when using other fiber angles and the increasing number of layers makes the computational cost too large, at least for this thesis. A analysis was performed using 6 layers and fiber angle $\alpha = 60°$ where the analysis was terminated after a CPU time of 900350 seconds. The analysis was then only 71 finished and based on the computational cost of increasing the number of layers, only 4 layers will be used in further studies.

4.3.4 Study of Fiber Distribution

In this section different ranges of fiber angles are studied to see how the different distributions affect the systolic deformation. For practical reasons, the response using an active stress component in different directions will also be studied here. The different cases of fiber angles chosen are

1. α_{endo} = 0°, α_{epi} = 0°
2. α_{endo} = -45°, α_{epi} = 45°
3. α_{endo} = -60°, α_{epi} = 60°
4. α_{endo} = -70°, α_{epi} = 70°

The angles are chosen both to be within the realistic angles reported in literature, but also to show how the left ventricle would act without any fiber angle. During this section the sheet angle β = 0° for all cases. It is noted that the maximum activation level T_{max} refers to the maximum value in the fiber direction. Thus for T_{max} = 100kPa in the fiber direction, we have T_{max}(n,n) = 60kPa and T_{max}(s,n) = 3kPa when using equations (3.18) and (3.19).

Activation in fiber direction

As a first try, activation was only added in the fiber direction. With T_{max} = 100kPa, and for all ranges of fiber angles, the model did not exhibit any contraction and the volume increased during the analysis. There were a significant increase in longitudinal length and

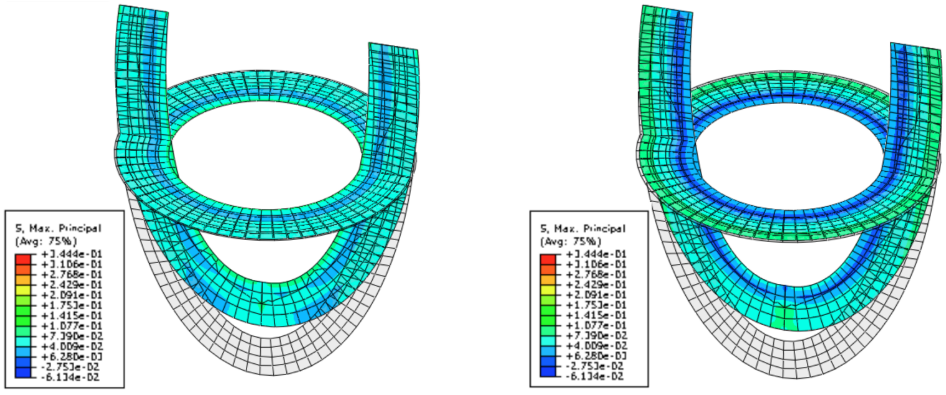
radius, and because the volume is constrained to constant, the walls became thinner. Attempts were made to increase the maximum activation level up to higher levels, but analysis with T_{max} = 300kPa still did not show any signs of a physiological correct contraction. This indicates, what already was found by Dorri et al. [6], that a realistic contraction can not be reproduced by using only an active stress component in the fiber direction.

Activation in fiber and sheet normal directions

In this section the results of the model using T_{max} = 100kPa in the fiber and sheet normal direction is presented. The myocardial torsion as defined by Carreras et al. [3], the difference between apical and basal rotation, has a physical value in the approximate range 9-12°. By comparing this with the values predicted by the models in Table 4.4, it is clear that the model has a significantly larger torsion than what is physiologically true. Further, the model rotates in the opposite direction, clockwise when looking from the apex to the base, then what it is supposed to. This can be account for in what directions the muscle fibers now are activated in.

α [°]	β [%]	EF [°]	AMT [%]	$\frac{\Delta h}{h}_{equator}$ [%]	$\frac{\Delta h}{h}_{apex}$ [%]	$\frac{\Delta L}{L}$ [%]	$\frac{\Delta R}{R}$ [%]
45	0	25.8	-23.2	18.4	19.3	18.0	5.6
60	0	27.2	-25.3	18.2	19.5	15.4	7.6
70	0	28.3	-23.8	18.0	19.7	13.6	9.0

Table 4.4: Response of left ventricular model with different fiber angles. β = 0 and T_{max} = 100kPa in fiber and sheet normal directions.



(a) T_{max} = 100kPa. **(b)** T_{max} = 150kPa.

Figure 4.12: First principal stress for deformed and undeformed states with α = 45°, T_{max} = 100 kPa and 150kPa in (ff) and (nn) directions. Cuts made for Y=0 and Z=-15 mm.

By examining Figure 4.4, we observe that the contraction in the sheet normal direction is considerably larger than in the fiber direction. The model, giving the results in Table 4.4, has no additional sheet angle and thus the sheet normal axis is directed in the normal direction from the fiber angle, in the circumferential plane. A larger contraction in the sheet normal direction therefore gives a clockwise rotation and activation only in fiber and sheet normal direction is then not able to reproduce a realistic torsion response. The results in Table 4.4 also reveals a very low internal radial displacement. The radial movement is function of both wall thickening and more rigid body movement. The longitudinal shortening however, displays, most significant for α=45°, more realistic values. What is not so evident from the numbers, but can be visually interpreted from Figure 4.12a is that the radial movement in the equatorial region is more as a consequence of the longitudinal displacement than actual radial displacement. Looking at the most basal region, we actually see a slight outwards movement in the radial direction at the epicardium, which indicates that the current model is not realistically reproducing the radial movement.

α [°]	EF [%]	AMT [°]	$\frac{\Delta h}{h}_{equator}$ [%]	$\frac{\Delta h}{h}_{apex}$ [%]	$\frac{\Delta L}{L}$ [%]	$\frac{\Delta R}{R}$ [%]
45	28.2	-24.6	19.6	20.3	17.6	7.5
60	29.5	-26.6	19.4	20.3	15.2	9.3
70	30.4	-26.3	19.4	20.5	13.7	10.6

Table 4.5: Response of left ventricular model with different fiber angles. β = 0 and T_{max} = 150kPa in fiber and sheet normal directions.

T_{max} was therefore increased to 150kPa to see if a more realistic radial contraction could be produced. By comparing the results in Tables 4.5 and 4.4, a larger radial displacement and wall thickness is in fact produced. As a result the ejection fractions are somewhat higher, although the longitudinal shortening is now lower. Again, by examining the epicardial, basal region in Figure 4.12b, it is clear that model has very little radial displacement here.

Activation in fiber, sheet normal and shear (sn) directions

The following presents the results of the analysis with active stress components in fiber, sheet normal and shear (sn) directions. The initial analysis using maximum activation level T_{max} = 100kPa showed a significant increase of the ventricular volume. This indicates that the active components is not able to overcome the internal pressure on the endocardial surface. The level was therefore increased to 150kPa, which is still within a realistic range, where more realistic deformations was achieved.

The inclusion of a shear component in the (sn) direction of the active stress tensor, has as expected produced torsion in the correct, counter-clockwise direction. The values of the torsion is still significantly larger than what is physically correct, but some of this can be accounted for in lack of right ventricle and the symmetrical geometry of the model.

α [°]	EF [%]	AMT [°]	$\frac{\Delta h}{h}_{equator}$ [%]	$\frac{\Delta h}{h}_{apex}$ [%]	$\frac{\Delta L}{L}$ [%]	$\frac{\Delta R}{R}$ [%]
0	34.2	7.5	18.7	21.0	3.9	17.4
45	31.1	27.2	17.6	19.8	8.9	12.9
60	29.0	31.5	17.5	20.3	11.9	10.2
70	27.7	16.5	17.6	20.4	13.4	9.0

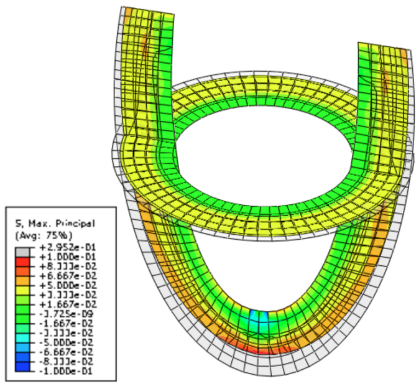
Table 4.6: Response of left ventricular model with different fiber angles. β = 0 and T_{max} = 150kPa in fiber, sheet normal and shear (sn) directions

The longitudinal shortening is somewhat lower now than without the shear activation. It is reasonable that the contraction in the fiber direction is the major component contributing to the longitudinal shortening. Seeing as both the fiber and sheet normal axis lies in the tangential plane to the wall surfaces, it is logical that contraction in sheet normal direction also contributes a little. This is supported by the increasing longitudinal shortening for increasing fiber angles, where the fiber axis gets a greater longitudinal components. By studying Figures 4.4 and 4.5, we see that the combined stretch in fiber and sheet normal directions is larger without activation in the shear direction. This fact may explain the reduced longitudinal shortening when adding the shear component to the active stress.

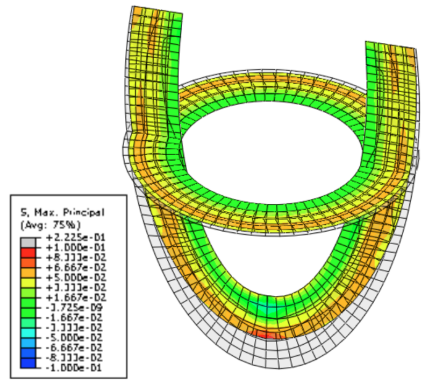
Although some slight differences, the fiber angle can not be said to have a great affect on the wall thickening, considering the values for both the apical and equatorial regions. The values are also significantly lower than physically correct.

Comparing Tables 4.5 and 4.6 it is clear that the active shear component to a larger degree is able to create a radial inwards motion. This is also visually observed in Figure 4.13. The magnitude from the more physically correct fiber angles are still quite small compared to over 30%, which is physiologically correct. What is evident, from studying Table 4.6 is that none of the different fiber angles are able to recreate the physical ejection fraction which is considered to be approximately 55-65%.

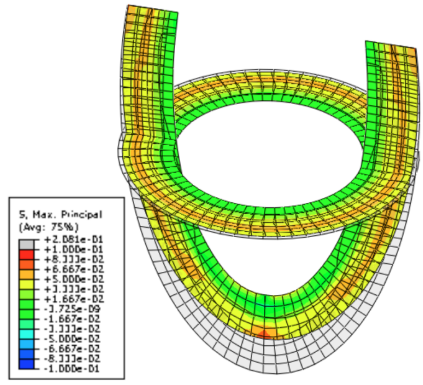
Figure 4.14 shows a typical rotation pattern seen through the analysis. It is easily observed that the rate of rotation is largest at the start of the systolic phase, which agrees with the material becoming exponentially stiffer under deformation. It is however in contradiction with the findings of Carreras et al. [3] where a more continuous rotation is seen throughout systole. As for any of the results from this model, the result at end-systole should be given more emphasise than results during the analysis. Seeing as the active contraction is implemented as a linear function, without much physiological meaning, response patterns during systole must me considered as a general trend rather than exact. The large rotation early in systole is however so significant, that its deviation from experimental results must be noted. Further, the rotation at the endocardium is in the opposite direction from the epicardium, for both basal and apical regions. This indicates a clockwise rotation for the endocardium which is consistent with what physically happens. In the earliest stage of the analysis, for both the base and apex, the endocardium first rotates counter clockwise before changing direction to clock wise. This is most predominant at



(a) $\alpha = 45^\circ$.



(b) $\alpha = 60^\circ$.



(c) $\alpha = 70^\circ$.

Figure 4.13: First principal stress for deformed and undeformed states T_{max} = 150 kPa in (ff), (nn) and (sn) directions. Cuts made for Y=0 and Z=-15 mm.

the apex. This can be explained by the larger rotation gradient at the epicardium dragging the endocardium at the start, before the the fiber tension at the endocardium overcomes this effect. The epicardium at the base is experimentally shown to exhibit a counterclockwise rotation is the first 30% of systole. [3] The model is not able to reproduce this behaviour, but this is not to be expected given the form of the active stress tensor.

4.3.5 Study of Activation Level

During the study of which directions of activation that needs to be included, a maximum activation stress of 100kPa was initally used, as this is roughly at the level indicated in literature. [6] When including activation also in shear (sn) direction, this activation level was no longer sufficient to overcome the pressure on the endocardium and the model showed an increase in volume rather than decrease. The level was therefore increased to 150kPa where contraction was achieved, but still the ejection fraction was no where

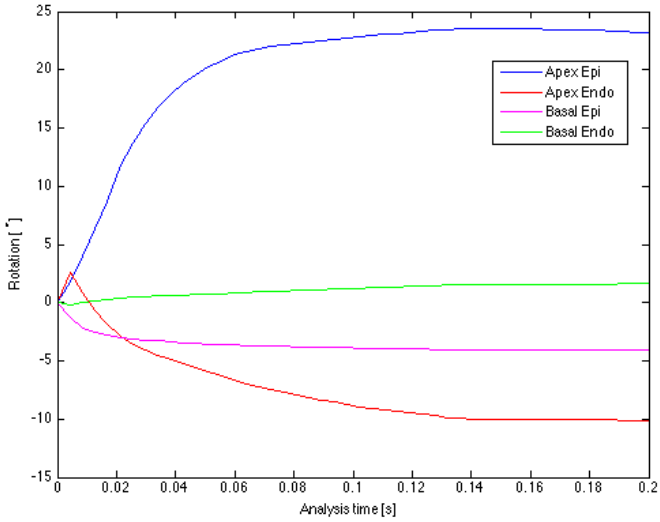


Figure 4.14: Rotation at apex and base, both for endocardium and epicardium, for $\alpha = 45°$, $\beta = 0°$, T_{max}=150kPa in (ff), (nn) and (sn) directions.

near the physical value. A study is therefore performed to find at what activation level a physically realistic ejection fraction is reached. From Table 4.6 we find that fiber angle $\alpha = 70°$ produced the highest ejection fraction, but since the computational cost is much lower, $\alpha = 45°$ is chosen for this study.

α [°]	T_{max} [%]	EF [°]	AMT [%]	$\frac{\Delta h}{h}_{equator}$ [%]	$\frac{\Delta h}{h}_{apex}$ [%]	$\frac{\Delta L}{L}$ [%]	$\frac{\Delta R}{R}$
45	150	31.1	27.2	17.6	19.8	8.9	12.9
45	200	33.1	27.7	19.3	20.3	8.4	14.8
45	300	35.1	28.6	21.0	21.0	7.9	16.6
45	500	36.9	29.9	22.3	22.1	7.6	18.0
45	1000	38.9	31.4	23.6	23.5	7.5	19.3

Table 4.7: Response of left ventricular model with increasing T_{max} for $\alpha = 45°$, $\beta = 0°$ and activation in fiber, sheet normal and shear (sn) directions.

Referring to Table 4.7, we see that as expect the ejection fraction increases for increasing T_{max}. There is also an increase in wall thickening and radial shortening, where as the longitudinal shortening is decreased. An ejection fraction of 38.9% when using T_{max}=1000kPa is still however far from a realistic value of 55-65%. It is therefore clear that despite using an active stress component well over what can be considered physiolog-

ical, the model is unable to produce a significantly larger ejection fraction.

A reason for this can be found by examining Figure 4.5. When increasing T_{max} to 200kPa for one cube element, a larger deformation is seen, but because of the exponential stiffening of the material, a increasingly larger stress is needed to produce a larger deformation. Further, this result strongly indicate that some mechanisms, which are not included in this mode, play a important part of the left ventricular function. Not surprisingly, an increasing T_{max} also produces an increasing ventricular torsion

4.3.6 Study of the Sheet Angle

Analysis were run using both sheet angle $\beta = \pm 45°$ and $\beta = \pm 85°$, which are both estimated to be within the physiological range. [9] Using activation in (ff), (nn) and (sn) directions with T_{max}=150kPa the model showed no contraction at all. To the contrary the volume increased during the analysis, showing large outwards radial displacement. As a result of this, the analysis it self became very slow, using very small time increments, and was terminated before completion. This was seen for all ranges of fiber angles. Attempts were made to increase T_{max}, but this did not affect the response noticeably.

For analysis using $\beta = 0°$, the sheet axis has a normal direction to the endo-/epicardial surface. The large stretch in the $\mathbf{s}$ direction, seen in Figure 4.5b, may be considered as the main component of creating both the radial motion and wall thickening. By varying β transmurally, a significantly smaller component is directed normal, which may to some degree explain why no realistic contraction is seen. However, the lack of contraction can at this point not fully be explained and further studies is needed here.

4.3.7 Volumetric Penalty Parameter Study

The volumetric penalty parameter κ is the parameter governing the volumetric change of the material. As the myocardium is considered to be nearly incompressible material, it has here been modelled as completely incompressible by setting $\kappa = 10^5$MPa. This arbitrary value is chosen to be much larger than the other material parameters governing the material law in Table 3.1. By choosing lower value, the volumetric change becomes less restricted.

κ [MPa]	Δ V [%]	EF [%]	AMT [°]	$\frac{\Delta h}{h}_{equator}$ [%]	$\frac{\Delta h}{h}_{apex}$ [%]	$\frac{\Delta L}{L}$ [%]	$\frac{\Delta R}{R}$ [%]
100000	0	31.1	27.2	17.6	19.9	8.9	13.0
1000	0.003	31.1	27.2	17.6	19.9	8.9	13.0
1	0.32	31.8	27	18.4	19.3	9	13.3
0.5	0.62	32.2	26.6	18.9	18.9	8.9	13.5
0.1	2.97	32.1	24.7	21	15.9	8.4	13.4
0.05	5.98	30.2	23.6	22.3	14.2	7.7	12.4

Table 4.8: Results from study of different values of κ, with $\alpha = 45°$, $\beta = 0$ and T_{max} = 150kPa in fiber, sheet normal and shear (sn) directions

Eriksson et al. [9] used a volumetric penalty parameter μ_k = 3333kPa in the volumetric function $U(J) = \mu_k ln(J)^2/2$. They experienced a small reduction of the ventricular wall volume during systole. The values in Table 4.8 show that a decreased κ gives an increased ventricular wall volume. Figure 4.15 shows a contour plot of the element volumes for κ=0.05MPa giving a increased volume of approximately 6 %. The increasing element volumes are predominantly seen on the sub-epicardial apical region, where as the sub-endocardial elements has decreased volumes.

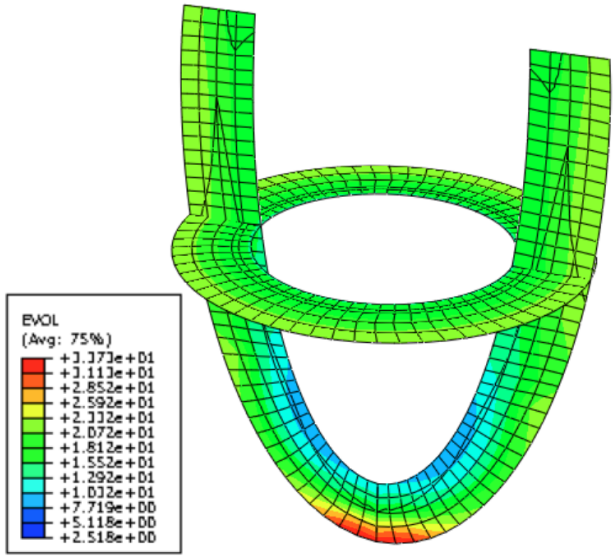


Figure 4.15: Contour plot of element volume for κ = 0.05MPa.

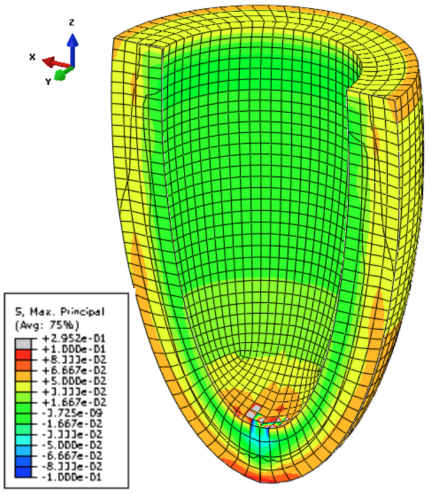
It is also interesting, that a part until a certain point, a more relaxed volumetric constraint produces a higher ejection fraction. This may be because the change in volume allows for a larger radial displacement without forcing a lower longitudinal shortening.

4.3.8 Stress and Strain Distribution

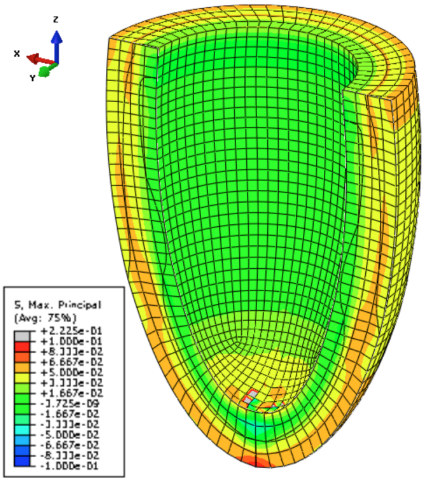
Figure 4.16 shows the distribution of the first principal stress for models with fiber angle α = 45°, 60° and 70° using T_{max} = 150kPa in (ff), (nn) and (sn) directions. The highest values of stresses are found at the apex. Here it is clear that some very high non-physical stresses also occur, which may be considered as a byproduct of the numerical method. All three models show in general a very similar stress pattern. In the region from mid-wall to sub-endocardium, the stresses are slightly negative as result of the pressure on the endocardial surface. From mid-wall to the sub-epicardial region the model shows stresses up to approximately 50kPa. There is thus a stress gradient from lower to higher transmurally from the endocardium to the epicardium.

When looking at the distribution of the first principal strains in Figure 4.17, the different fiber angles show a relatively similar pattern. For the majority of the left ventricle

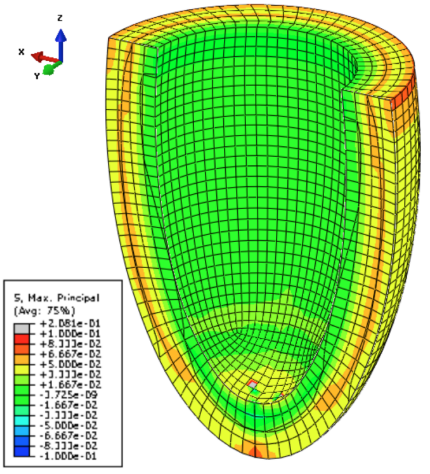
the models show strains in average range 0.20-0.25, with the exception of the endocardial region at the apex.



(a) $\alpha = 45°$.

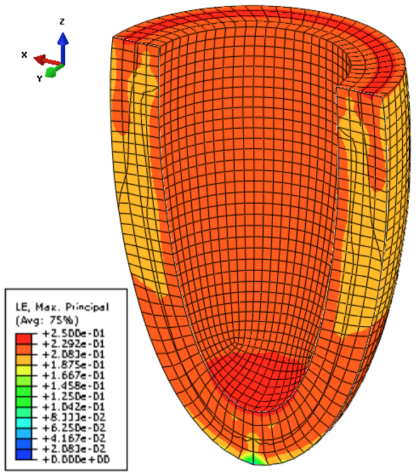


(b) $\alpha = 60°$.

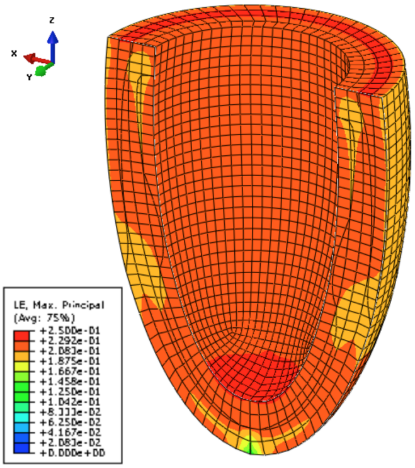


(c) $\alpha = 70°$.

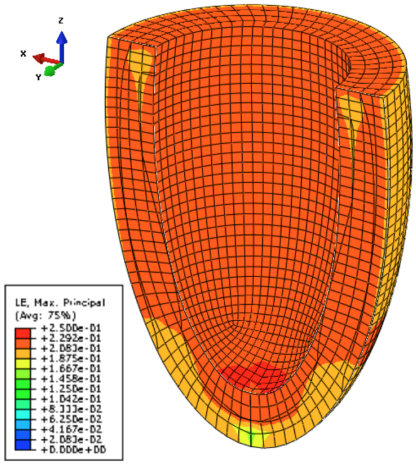
Figure 4.16: First principal stresses [MPa] for fiber angles $\alpha = 45°$, $60°$ and $70°$ at end-systole.



(a) $\alpha = 45^\circ$.



(b) $\alpha = 60^\circ$.



(c) $\alpha = 70^\circ$.

Figure 4.17: First principal strains for fiber angles $\alpha = 45^\circ$, 60° and 70° at end-systole.

CHAPTER 5
DISCUSSION

The results have to some extent already been discussed in the previous chapter. Here this is extended upon and the results are further weighed up against the simplifications and assumptions made in the modelling.

5.1 Torsion

In general, when looking at the analysis with fiber angles which are physiological reasonable, the model significantly overestimates the torsion. There may be several reasons for this. First of all the simplified geometry, which is very symmetrical, is bound to increase the torsion as non-symmetry should add to the resistance to torsion. Further, the model excludes the right ventricle, which should restrain torsion as the septal region would be more restricted. A study done by Eriksson et al. [9], comparing rule based fiber angle distribution and a more heterogeneous distribution, gave more realistic torsion for the heterogeneous distribution, indicating that the simplified approach for implementing the fiber distribution has some drawbacks. Further, it is unknown how well the chosen boundary conditions replicate those which are physically correct. In the model, the boundary conditions were chosen to avoid possible rigid body motions, but allow deformation and rotation. In a real heart both the aorta and pulmonary arteries, as well as surrounding tissues, will play a part in restraining the heart deformation. Little data exist concerning the effects, but it is reasonable to believe that they will to some extent restrain the rotation.

5.2 Wall Thickening

A feature which the model have problems of recreating is the wall thickening. Some explanation of why this is can be found by examining what the model does not include. It is believed that the cleavage planes in the myocardium are an important function to facilitate the wall thickening. The muscle fibers are here able to rearrange during thickening

trough shear deformation. This sliding of sheets along the cleavage planes, predominant in the inner third of the wall, is thought to play a significant part in the thickening. [19] The cleavage planes and sliding mechanics is in no way explicitly included in the model. However as the the material parameters governing the constitutive material law is based on shear experiments, some of this may be captured in these data. Considering this, it is important to bare in mind that these test were done on passive myocardium ex vivo, in a small scale. The sliding of sheets along cleavage planes is very much a global mechanical function present during the contraction of the muscle, and thus it is unlikely that this is captured in the experimental shear tests.

5.3 Fiber and Sheet Angles

As noted in the implementation of the fiber and sheet angles, the imbrication angle (out-of-plane angle) is not included and following the fiber axis is always parallel to the endo-/epicardial surfaces. This is assumed to have little affect in the major part of the ventricle, as the imbrication angle is mostly $< 5^\circ$. However, as noted by Legrice et al. [19], the imbrication angle is not small in the apical and basal region and will definitely affect the local mechanics. Using a homogeneous fiber field as in this model is of course a simplification of what is physiological correct, but it gives us an opportunity to analyse how different fiber angles affect the left ventricular contraction. The results presented in Table 4.6 show that a fiber field varying from -45° to +45°, from the endocardium to the epicardium, gives a larger radial shortening and higher wall thickening, than a fiber field with $\pm 70^\circ$. At the same time, longitudinal shortening increases when comparing increasing the maximum fiber angle. It therefore seems that fibers which are more circumferential aligned adds to the radial movement and the more longitudinally aligned fibers contributes to the longitudinal shortening. However, it is not apparent why the circumferential aligned fiber should add to the radial shortening, as they are still tangential to the wall surfaces when the sheet angle is kept at 0°. Bearing in mind the incompressible nature of the material, the radial shortening may actually become larger because of the lack of longitudinal shortening. Since the material must conserve its volume, a large longitudinal shortening acts as a constraint on the the radial movement, and on the wall thickening. This is most predominant in the case of fiber angle $\alpha = 0^\circ$. Here all fibers are aligned circumferential throughout the ventricle, something which is physically unrealistic. This produces almost no longitudinal shortening, but as can be seen in Table 4.6, both the radial movement and the wall thickening are considerably larger than any other case. It is also interesting to note that the case using $\alpha = 0^\circ$ is actually the one with the highest ejection fraction. This cannot be interpreted to have any physical meaning, but it highlights how important the radial movement and wall thickening in producing the ejection fraction, and thus also for the effectiveness of the heart.

5.4 The Passive Constitutive Model

The validation of the constitutive model of the passive myocardium, confirms that the model is able to show a correct response. As earlier noted, there is not a perfect match

between the Abaqus results and the experimental results. This slight stiffer behaviour might to some degree affect the behaviour of the model. If the ratio of stiffness between the different directions is different in the model, it will deform easier in one direction than is physiological correct. This might have some affect the global response. However, the slight stiffer behaviour itself can not account for the relatively low ejection fraction seen in the model. During the analysis of activation level, the stress component were increased well beyond physiological values without the ejection fraction coming close to that of a real heart. Had the stiffness been an important factor, the added stress would have created relatively larger ejection fraction than it did. This leads to suggest that features which are not included in the model, may play a big part in creating the physiological correct ejection fraction. Further, there is a need for further experiments on the myocardium. Current models still use the work Dokos et al. [5] as the only source of experimental data. Though this work was a breakthrough in the understanding of the orthotropic nature of the myocardium, more experiments should be performed to further validate the mechanical properties of the myocardium.

Considering the left ventricle as only consisting of myocardium is also somewhat inaccurate. Tissue in both the endocardial and epicardial regions are different from the myocardium and thus will have different mechanical properties. This will definitively have an some effect on the systolic deformation which is not included in the model.

In this model the myocardium is modelled as incompressible by setting the volumetric penalty parameter in the constitutive law, $\kappa = 10^5$MPa. It is however noted that during systole there is a spatially and time varying changing coronary blood pressure. This change in fluid volume affects the hydrostatic pressure and therefore introduces an effective compressibility. [23] The modelling of this variation over the course of the systolic phase needs a fluid interaction which is not within the scope of this thesis.

5.5 Active Contraction

In this work a very simplified model of the active stress components have been used. This approach is sufficient to create a contraction in the left ventricle and by using relatively physiological stress levels, gives a good approximation. However, since the variation of active stress through systole is only time dependant, and with limited physical meaning, only end-systolic results should be considered. The next step in this work would be to implement a more physiological correct active stress model. Models, such as the HMT-model, take in parameters as Ca^{2+} concentrations and fiber stretch, which gives a more realistic evolution of the active stress component.

Further, in this work the size of the different components of the active stress tensor have been based on findings in previous work, and not investigated thoroughly. It is clear, by observing the different principal stretches in Figures 4.4 and 4.5, that the adding of a small shear component has a large effect. The size of this component includes a great deal of uncertainty, as little work has been done to investigate it. It is though very likely that different values would affect the deformation seen in the left ventricular model, and this is something which may be of interest for further studies.

Lastly, a known problem within finite element analysis, is that discontinues loads can cause numerical difficulties. Both the HMT-model and a model using an error function,

has a smooth, continuous stress development, which could prove to be beneficial from a numerical standpoint.

CHAPTER 6

CONCLUDING REMARKS

This works has shown that the invariant based constitutive model proposed by Holzapfel and Ogden [14] is to successfully able to reproduce the mechanical response of a full scale, left ventricular model. By using a rule based implementation of the cardiac muscle fiber field and a simplified model for the active muscle, the model is able to reproduce some of the main features describing ventricular contraction. The results of the work show that in able to correctly produce contraction, the active stress must have components in fiber, sheet normal and shear (sn) directions.

The model is not able to reproduce a realistic ejection fraction, even when using non-physiological, high active stress components. The work has here shown that it is challenging to properly model the radial shortening and wall thickening, two of the most important features of the ventricular contraction. This strongly indicates that mechanical features, such as sliding along cleavage planes, which are not included in the present model is necessary to achieve more realistic results. Further, the inclusion of a sheet angle did not give any realistic contraction, even when the active stress components were significantly increased. It is finally clear that a model of only the left ventricle, using a simplified, symmetrical geometry, overestimates the torsion.

Numerically the analysis has proven to be somewhat unstable considering rate convergence and computational cost. Several factors, as fiber angle and active stress components, is shown to greatly affect the convergence and force the analysis to use very small time increments.

BIBLIOGRAPHY

[1] Abaqus, 2014. Abaqus 6.14. Abaqus Analysis User's Guide. Dassault Systemes.

[2] Barnett, V., 2009. Cellular myocytes. In: Iaizzo, P. A. (Ed.), Handbook of Cardiac Anatomy, Physiology, and Devices. Humana Press, pp. 147–158.

[3] Carreras, F., Garcia-Barnes, J., Gil, D., Pujadas, S., Li, C., Suarez-Arias, R., Leta, R., Alomar, X., Ballester, M., Pons-Llado, G., 2012. Left ventricular torsion and longitudinal shortening: two fundamental components of myocardial mechanics assessed by tagged cine-mri in normal subjects. The International Journal of Cardiovascular Imaging 28 (2), 273–284.

[4] Costa, K. D., Holmes, J. W., McCulloch, A. D., 2001. Modelling cardiac mechanical properties in three dimensions. Philosophical Transactions: Mathematical, Physical and Engineering Sciences 359 (1783), pp. 1233–1250.

[5] Dokos, S., Smaill, B. H., Young, A. A., LeGrice, I. J., 2002. Shear properties of passive ventricular myocardium. American Journal of Physiology - Heart and Circulatory Physiology 283 (6), H2650–H2659.

[6] Dorri, F., Niederer, P. F., Lunkenheimer, P. P., 2006. A finite element model of the human left ventricular systole. Computer Methods in Biomechanics and Biomedical Engineering 9 (5), 319–341.

[7] Dumesnil, J. G., Shoucri, R. M., 1991. Quantitative relationships between left ventricular ejection and wall thickening and geometry. Journal of Applied Physiology 70 (1), 48–54.

[8] Eggen, M., Swingen, C., 2009. Cardiovascular magnetic resonance imaging. In: Iaizzo, P. A. (Ed.), Handbook of Cardiac Anatomy, Physiology, and Devices. Humana Press, pp. 341–362.

[9] Eriksson, T., Prassl, A., Plank, G., Holzapfel, G., 2013. Influence of myocardial fiber/sheet orientations on left ventricular mechanical contraction. Mathematics and Mechanics of Solids 18 (6), 592–606.

[10] Gilbert, S. H., Benson, A. P., Li, P., Holden, A. V., 2007. Regional localisation of left ventricular sheet structure: integration with current models of cardiac fibre, sheet and band structure. European Journal of Cardio-Thoracic Surgery 32 (2), 231–249.

[11] Göktepe, S., Acharya, S. N. S., Wong, J., Kuhl, E., 2011. Computational modeling of passive myocardium. International Journal for Numerical Methods in Biomedical Engineering 27 (1), 1–12.

[12] Holzapfel, G., Gasser, T., Ogden, R., 2000. A new constitutive framework for arterial wall mechanics and a comparative study of material models. Journal of elasticity and the physical science of solids 61 (1-3), 1–48.

[13] Holzapfel, G. A., 2005. Nonlinear Solid Mechanics A Continuum Approach For Engineering. John Wiley and Sons Ltd.

[14] Holzapfel, G. A., 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.

[15] Humphrey, J., Yin, F., 1987. A new constitutive formulation for characterizing the mechanical behavior of soft tissues. Biophysical Journal 52 (4), 563 – 570.

[16] Hunter, P., McCulloch, A., ter Keurs, H., 1998. Modelling the mechanical properties of cardiac muscle. Progress in Biophysics and Molecular Biology 69 (23), 289 – 331.

[17] Hunter, P., McCulloch, A., ter Keurs, H., 1998. Modelling the mechanical properties of cardiac muscle. Progress in Biophysics and Molecular Biology 69 (23), 289 – 331.

[18] Iaizzo, P., 2005. General features of the cardiovascular system. In: Iaizzo, P. (Ed.), Handbook of Cardiac Anatomy, Physiology, and Devices. Humana Press, pp. 3–11.

[19] Legrice, I. J., Hunter, P. J., Smaill, B. H., 1997. Laminar structure of the heart: a mathematical model. American Journal of Physiology - Heart and Circulatory Physiology 272 (5), H2466–H2476.

[20] Levick, J., 2010. An Introduction to Cardiovascular Physiology, 5Tth edition. Hodder Arnold.

[21] Nakatani, S., 2011. Left ventricular rotation and twist: Why should we learn. Journal of Cardiovascular Ultrasound 19 (1), 1–6.

[22] Nakatani, S., 2011. Left ventricular rotation and twist: why should we learn. Journal of Cardiovascular Ultrasound 19 (1), 1–6.

[23] Nash, M., Hunter, P., 2000. Computational mechanics of the heart. Journal of elasticity and the physical science of solids 61 (1-3), 113–141.

[24] Remme, E., Hunter, P., 2004. The influence of material properties on left ventricular deformation in an elliptical model.

[25] Rohmer, D., Sitek, A., Gullberg, G. T., 2007. Reconstruction and visualization of fiber and laminar structure in the normal human heart from ex vivo diffusion tensor magnetic resonance imaging (dtmri) data. Investigative Radiology 42 (1), 777–789.

[26] Rssel, I. K., Gtte, M. J., Bronzwaer, J. G., Knaapen, P., Paulus, W. J., van Rossum, A. C., 2009. Left ventricular torsion: An expanding role in the analysis of myocardial dysfunction. JACC: Cardiovascular Imaging 2 (5), 648 – 655.

[27] Schmid, H., Nash, M. P., Young, A. A., Hunter, P. J., 2006. Myocardial material parameter estimationa comparative study for simple shear. Journal of Biomechanical Engineering 128 (5), 742–750.

[28] Skallerud, B., Prot, V., Nordrum, I., 2011. Modeling active muscle contraction in mitral valve leaflets during systole: a first approach. Biomechanics and Modeling in Mechanobiology 10 (1), 11–26.

[29] Wang, H. M., Gao, H., Luo, X. Y., Berry, C., Griffith, B. E., Ogden, R. W., Wang, T. J., 2013. Structure-based finite strain modelling of the human left ventricle in diastole. International Journal for Numerical Methods in Biomedical Engineering 29 (1), 83–103.

[30] Zulliger, M. A., Rachev, A., Stergiopulos, N., 2004. A constitutive formulation of arterial mechanics including vascular smooth muscle tone. American Journal of Physiology - Heart and Circulatory Physiology 287 (3), H1335–H1343.

Appendix

A Create Abaqus Model

```
#This python script recreates the 4 layer Truncated Ellipsoid
    model used in the
#greater part of this thesis.


#----------------------Initialize-----------------------

from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *


#----------------------Create model----------------------

mdb.Model(name='Model-1')
mdb.models['Model-1'].setValues(noPartsInputFile=ON)


#----------------------Create part-----------------------

mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
    sheetSize=200.0)
mdb.models['Model-1'].sketches['__profile__'].ConstructionLine(
    point1=(0.0,
    -100.0), point2=(0.0, 100.0))
mdb.models['Model-1'].sketches['__profile__'].FixedConstraint(
    entity=
```

```
    mdb.models['Model-1'].sketches['__profile__'].geometry[2])
mdb.models['Model-1'].sketches['__profile__'].
    EllipseByCenterPerimeter(
    axisPoint1=(0.0, 65.0), axisPoint2=(25.0, 0.0), center=(0.0,
        0.0))
mdb.models['Model-1'].sketches['__profile__'].
    EllipseByCenterPerimeter(
    axisPoint1=(0.0, 75.0), axisPoint2=(35.0, 0.0), center=(0.0,
        0.0))
mdb.models['Model-1'].sketches['__profile__'].Line(point1=(0.0,
    -75.0000000018626), point2=(0.0, -65.0))
mdb.models['Model-1'].sketches['__profile__'].VerticalConstraint(
    addUndoState=
    False, entity=mdb.models['Model-1'].sketches['__profile__'].
        geometry[7])
mdb.models['Model-1'].sketches['__profile__'].ParallelConstraint(
    addUndoState=
    False, entity1=mdb.models['Model-1'].sketches['__profile__'].
        geometry[2],
    entity2=mdb.models['Model-1'].sketches['__profile__'].geometry
        [7])
mdb.models['Model-1'].sketches['__profile__'].CoincidentConstraint
    (
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices[5],
        entity2=
    mdb.models['Model-1'].sketches['__profile__'].geometry[2])
mdb.models['Model-1'].sketches['__profile__'].CoincidentConstraint
    (
    addUndoState=False, entity1=
    mdb.models['Model-1'].sketches['__profile__'].vertices[6],
        entity2=
    mdb.models['Model-1'].sketches['__profile__'].geometry[2])
mdb.models['Model-1'].sketches['__profile__'].Line(point1=(0.0,
    20.0), point2=(
    40.0, 20.0))
mdb.models['Model-1'].sketches['__profile__'].HorizontalConstraint
    (
    addUndoState=False, entity=
    mdb.models['Model-1'].sketches['__profile__'].geometry[8])
mdb.models['Model-1'].sketches['__profile__'].autoTrimCurve(curve1
    =
    mdb.models['Model-1'].sketches['__profile__'].geometry[3],
        point1=(
    -10.6040496826172, -57.5122756958008))
mdb.models['Model-1'].sketches['__profile__'].autoTrimCurve(curve1
    =
    mdb.models['Model-1'].sketches['__profile__'].geometry[5],
        point1=(
    -19.4768447875977, -62.0470886230469))
```

```
mdb.models['Model-1'].sketches['__profile__'].autoTrimCurve(curve1
    =
    mdb.models['Model-1'].sketches['__profile__'].geometry[8],
        point1=(
    11.0368957519531, 20.6593246459961))
mdb.models['Model-1'].sketches['__profile__'].autoTrimCurve(curve1
    =
    mdb.models['Model-1'].sketches['__profile__'].geometry[9],
        point1=(
    20.5588989257813, 34.0478363037109))
mdb.models['Model-1'].sketches['__profile__'].autoTrimCurve(curve1
    =
    mdb.models['Model-1'].sketches['__profile__'].geometry[10],
        point1=(
    30.2973175048828, 34.0478363037109))
mdb.models['Model-1'].sketches['__profile__'].autoTrimCurve(curve1
    =
    mdb.models['Model-1'].sketches['__profile__'].geometry[11],
        point1=(
    37.2224273681641, 20.2274398803711))
mdb.models['Model-1'].Part(dimensionality=THREE_D, name='TEM',
    type=
    DEFORMABLE_BODY)
mdb.models['Model-1'].parts['TEM'].BaseSolidRevolve(angle=360.0,
    flipRevolveDirection=OFF, sketch=
    mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']


#----------------------Partition part----------------------

mdb.models['Model-1'].parts['TEM'].PartitionCellByPlaneThreePoints
    (cells=
    mdb.models['Model-1'].parts['TEM'].cells.getSequenceFromMask((
        '[#1 ]', ), )
    , point1=mdb.models['Model-1'].parts['TEM'].InterestingPoint(
    mdb.models['Model-1'].parts['TEM'].edges[1], MIDDLE), point2=
    mdb.models['Model-1'].parts['TEM'].InterestingPoint(
    mdb.models['Model-1'].parts['TEM'].edges[3], MIDDLE), point3=
    mdb.models['Model-1'].parts['TEM'].vertices[0])
mdb.models['Model-1'].parts['TEM'].PartitionCellByPlaneThreePoints
    (cells=
    mdb.models['Model-1'].parts['TEM'].cells.getSequenceFromMask((
        '[#2 ]', ), )
    , point1=mdb.models['Model-1'].parts['TEM'].InterestingPoint(
    mdb.models['Model-1'].parts['TEM'].edges[9], MIDDLE), point2=
    mdb.models['Model-1'].parts['TEM'].InterestingPoint(
    mdb.models['Model-1'].parts['TEM'].edges[8], MIDDLE), point3=
    mdb.models['Model-1'].parts['TEM'].vertices[1])
```

```
mdb.models['Model-1'].parts['TEM'].PartitionCellByPlaneThreePoints
    (cells=
    mdb.models['Model-1'].parts['TEM'].cells.getSequenceFromMask((
        '[#2 ]', ), )
    , point1=mdb.models['Model-1'].parts['TEM'].InterestingPoint(
    mdb.models['Model-1'].parts['TEM'].edges[14], MIDDLE), point2=
    mdb.models['Model-1'].parts['TEM'].InterestingPoint(
    mdb.models['Model-1'].parts['TEM'].edges[13], MIDDLE), point3=
    mdb.models['Model-1'].parts['TEM'].vertices[1])
mdb.models['Model-1'].ConstrainedSketch(gridSpacing=5.06, name='
    __profile__',
    sheetSize=202.46, transform=
    mdb.models['Model-1'].parts['TEM'].MakeSketchTransform(
    sketchPlane=mdb.models['Model-1'].parts['TEM'].faces[8],
    sketchPlaneSide=SIDE1,
    sketchUpEdge=mdb.models['Model-1'].parts['TEM'].edges[14],
    sketchOrientation=TOP, origin=(23.339913, -28.854688, 0.0)))
mdb.models['Model-1'].parts['TEM'].projectReferencesOntoSketch(
    filter=
    COPLANAR_EDGES, sketch=mdb.models['Model-1'].sketches['
        __profile__'])
mdb.models['Model-1'].sketches['__profile__'].sketchOptions.
    setValues(
    gridOrigin=(-23.3399130000917, 28.8546879999259))
mdb.models['Model-1'].sketches['__profile__'].
    EllipseByCenterPerimeter(
    axisPoint1=(-23.3399130000917, 98.8546879999259), axisPoint2=(
    6.6600869999083, 28.8546879999259), center=(-23.3399130000917,
    28.8546879999259))
mdb.models['Model-1'].parts['TEM'].PartitionFaceBySketch(faces=
    mdb.models['Model-1'].parts['TEM'].faces.getSequenceFromMask((
        '[#100 ]', ),
    ), sketch=mdb.models['Model-1'].sketches['__profile__'],
        sketchOrientation=
    TOP, sketchUpEdge=mdb.models['Model-1'].parts['TEM'].edges
        [14])
del mdb.models['Model-1'].sketches['__profile__']
mdb.models['Model-1'].parts['TEM'].PartitionCellBySweepEdge(cells=
    mdb.models['Model-1'].parts['TEM'].cells.getSequenceFromMask((
        '[#4 ]', ), )
    , edges=(mdb.models['Model-1'].parts['TEM'].edges[0], ),
        sweepPath=
    mdb.models['Model-1'].parts['TEM'].edges[9])
mdb.models['Model-1'].parts['TEM'].PartitionCellBySweepEdge(cells=
    mdb.models['Model-1'].parts['TEM'].cells.getSequenceFromMask((
        '[#2 ]', ), )
    , edges=(mdb.models['Model-1'].parts['TEM'].edges[0], ),
        sweepPath=
    mdb.models['Model-1'].parts['TEM'].edges[16])
mdb.models['Model-1'].parts['TEM'].PartitionCellBySweepEdge(cells=
```

```
    mdb.models['Model-1'].parts['TEM'].cells.getSequenceFromMask((
        '[#20 ]', ),
    ), edges=(mdb.models['Model-1'].parts['TEM'].edges[0], ),
        sweepPath=
    mdb.models['Model-1'].parts['TEM'].edges[28])
mdb.models['Model-1'].parts['TEM'].PartitionCellBySweepEdge(cells=
    mdb.models['Model-1'].parts['TEM'].cells.getSequenceFromMask((
        '[#10 ]', ),
    ), edges=(mdb.models['Model-1'].parts['TEM'].edges[0], ),
        sweepPath=
    mdb.models['Model-1'].parts['TEM'].edges[31])
mdb.models['Model-1'].parts['TEM'].DatumPlaneByPrincipalPlane(
    offset=0.0,
    principalPlane=XZPLANE)
mdb.models['Model-1'].parts['TEM'].DatumPlaneByPrincipalPlane(
    offset=-20.0,
    principalPlane=XZPLANE)
mdb.models['Model-1'].parts['TEM'].DatumPlaneByPrincipalPlane(
    offset=-40.0,
    principalPlane=XZPLANE)
mdb.models['Model-1'].parts['TEM'].PartitionCellByDatumPlane(cells
    =
    mdb.models['Model-1'].parts['TEM'].cells.getSequenceFromMask((
        '[#ff ]', ),
    ), datumPlane=mdb.models['Model-1'].parts['TEM'].datums[10])
mdb.models['Model-1'].parts['TEM'].PartitionCellByDatumPlane(cells
    =
    mdb.models['Model-1'].parts['TEM'].cells.getSequenceFromMask((
        '[#1cc7 ]',
    ), ), datumPlane=mdb.models['Model-1'].parts['TEM'].datums
        [11])
mdb.models['Model-1'].parts['TEM'].PartitionCellByDatumPlane(cells
    =
    mdb.models['Model-1'].parts['TEM'].cells.getSequenceFromMask((
        '[#18017c ]',
    ), ), datumPlane=mdb.models['Model-1'].parts['TEM'].datums
        [12])
mdb.models['Model-1'].parts['TEM'].Surface(name='endoSurf',
    side1Faces=
    mdb.models['Model-1'].parts['TEM'].faces.getSequenceFromMask((
    '[#300c000 #330 #33 #3042 ]', ), ))


#----------------------Create mesh----------------------

mdb.models['Model-1'].parts['TEM'].setElementType(elemTypes=(
    ElemType(
    elemCode=C3D8H, elemLibrary=STANDARD), ElemType(elemCode=C3D6,
    elemLibrary=STANDARD), ElemType(elemCode=C3D4, elemLibrary=
        STANDARD)),
```

```
    regions=(mdb.models['Model-1'].parts['TEM'].cells.
        getSequenceFromMask((
    '[#ffffffff ]', ), ), ))
mdb.models['Model-1'].parts['TEM'].seedPart(deviationFactor=0.1,
    minSizeFactor=
    0.1, size=2.5)
mdb.models['Model-1'].parts['TEM'].generateMesh()
mdb.models['Model-1'].parts['TEM'].assignStackDirection(cells=
    mdb.models['Model-1'].parts['TEM'].cells.getSequenceFromMask((
    '[#ffffffff ]', ), ), referenceRegion=
    mdb.models['Model-1'].parts['TEM'].faces[72])
mdb.models['Model-1'].parts['TEM'].PartFromMesh(copySets=True,
    name=
    'TEM-mesh-1')
mdb.models['Model-1'].parts.changeKey(fromName='TEM-mesh-1',
    toName=
    'TEM-4layers')


#----------------------Create sets----------------------

mdb.models['Model-1'].parts['TEM-4layers'].Set(elements=
    mdb.models['Model-1'].parts['TEM-4layers'].elements.
        getSequenceFromMask(
    mask=('[#ffffffff:5 #ffff #0:16 #ff000000 #ffffffff:9 #0:5 #
        ffff0000',
    ' #ffffffff:3 #ffff #0:33 #ffffffff:3 #ffffff #0:38 #ffffffff
        :5',
    ' #ffff #0:5 #ffffffff:4 #0:7 #ff000000 #ffffffff:3 #ffff',
    ' #0:5 #ffffffff:9 #ffff #0:3 #ffff0000 #ffffffff:3 #ffff',
    ' #0:7 #ffffff00 #ffffffff:3 #0:65 #ffffffff:3 #ffffff #0:7',
    ' #ffff0000 #ffffffff:3 #ffff:8 #5555ffff #55555555:7 ]', ), )
        , name=
    'layer1')
mdb.models['Model-1'].parts['TEM-4layers'].Set(elements=
    mdb.models['Model-1'].parts['TEM-4layers'].elements.
        getSequenceFromMask(
    mask=('[#0:5 #ffff0000 #ffffffff:5 #0:8 #ffffffff:3 #ffffff
        #0:9',
    ' #ffffffff:5 #ffff #0:3 #ffff0000 #ffffffff:3 #ffff #0:32',
    ' #ff000000 #ffffffff:3 #ffff #0:39 #ffff0000 #ffffffff:5 #0:4
        ',
    ' #ffffffff:7 #ffffff #0:3 #ffff0000 #ffffffff:5 #0:9 #
        ffff0000',
    ' #ffffffff:3 #ffff #0:3 #ffff0000 #ffffffff:7 #ff #0:71',
    ' #ff000000 #ffffffff:7 #ffff #0:3 #ffff0000:8 #aaaa0000 #
        aaaaaaaa:7 ]', ),
    ), name='layer2')
mdb.models['Model-1'].parts['TEM-4layers'].Set(elements=
```

```
    mdb.models['Model-1'].parts['TEM-4layers'].elements.
        getSequenceFromMask(
    mask=('[#0:15 #ffffffff:4 #0:32 #ffffffff:5 #ffff #0:5 #
        ffffffff:5',
    ' #ffff #0:3 #ffffff00 #ffffffff:3 #0:11 #ffff0000 #ffffffff:3
        ',
    ' #ffff #0:5 #ffffffff:5 #ffff #0:3 #ffff0000 #ffffffff:3',
    ' #ffff #0:3 #ffffff00 #ffffffff:3 #0:61 #55555555:8 #aaaaaaaa
        :7',
    ' #aaaa #0:3 #ffffff00 #ffffffff:3 #0:4 #ffffffff:4 #ffff:8',
    ' #c0007fff #f0001fff #fc0007ff #ff0001ff #ffc0007f #fff0001f
        #fffc0007',
    ' #1 #0:3 #ffffff00 #ffffffff:3 #0:5 #ffff0000 #ffffffff:5 ]',
         ), ), name=
    'layer3')
mdb.models['Model-1'].parts['TEM-4layers'].Set(elements=
    mdb.models['Model-1'].parts['TEM-4layers'].elements.
        getSequenceFromMask(
    mask=('[#0:11 #ffffffff:4 #0:30 #ffff0000 #ffffffff:5 #0:5 #
        ffff0000',
    ' #ffffffff:5 #0:5 #ffff0000 #ffffffff:3 #ff #0:10 #ffff0000',
    ' #ffffffff:3 #ffff #0:3 #ffff0000 #ffffffff:5 #0:5 #ffff0000'
        ,
    ' #ffffffff:3 #ffff #0:3 #ffff0000 #ffffffff:3 #ff #0:64',
    ' #aaaaaaaa:8 #55555555:7 #ffff5555 #ffffffff:3 #ff #0:3 #
        ffffffff:4',
    ' #0:4 #ffff0000:8 #3fff8000 #fffe000 #3fff800 #fffe00 #3fff80
        ',
    ' #fffe0 #3fff8 #fffffffe #ffffffff:3 #ff #0:3 #ffffffff:5', '
         #ffff ]', ),
    ), name='layer4')
mdb.models['Model-1'].parts['TEM-4layers'].Set(name='basalNodes',
    nodes=
    mdb.models['Model-1'].parts['TEM-4layers'].nodes.
        getSequenceFromMask(mask=(
    '[#0 #7afe0 #0:25 #ffc00000 #fffc000f #80010180 #bfffffff',
    ' #1fff80 #0 #ffff8000 #f0003fff #80ffffff #ffe03fff #fffff',
    ' #0:138 #c0000000 #fff #0:6 #7fff000 #0:3 #fffc0000',
    ' #0:7 #ffc00000 #1f #0:5 #c0000000 #fff #0:20',
    ' #fffffffe0 #7 #0:5 #fffc000 ]', ), ))
mdb.models['Model-1'].parts['TEM-4layers'].Set(name='endoNodes',
    nodes=
    mdb.models['Model-1'].parts['TEM-4layers'].nodes.
        getSequenceFromMask(mask=(
    '[#e1f0007 #fffe0198 #7fff7fff #0:5 #fffc0000 #8fffffff #
        fffffff',
    ' #fffff800 #ffff #0:4 #fc000000 #ffffffff #f #0:3',
    ' #fffff800 #7fff #fdffffff0 #fffeffff #0 #fffff800 #7f',
    ' #0:5 #ff000000 #ff9fffff #ffffffff:5 #3fff #0:16 #fff80000',
```

```
    ' #ffffffff:2 #1fffff #80000000 #ffffffff:4 #1ffffff #0:5 #
        ffff0000',
    ' #ffffffff:2 #1ffffff #0:32 #c0000000 #ffffffff:3 #0:24 #
        fffc0000',
    ' #ffffffff:4 #fc000fff #ffffffff:3 #7 #0:2 #f8000000 #
        ffffffff:2',
    ' #1fffffff #fffc0000 #ffffffff:7 #1fffff #fffff800 #ffffffff
        :2 #fffff',
    ' #0:6 #ffffc000 #ffffffff:2 #ffff #0:41 #fffff000 #ffffffff:5
        ',
    ' #7fffff #0 #fffffff8 #ffffffff:5 #3fff ]', ), ))
mdb.models['Model-1'].parts['TEM-4layers'].Set(name='apexNodes',
    nodes=
    mdb.models['Model-1'].parts['TEM-4layers'].nodes.
        getSequenceFromMask(mask=(
    '[#800024 #0:2 #1 #0:10 #10000000 ]', ), ))


#----------------------Create material----------------------

mdb.models['Model-1'].Material(name='orthotropic')
mdb.models['Model-1'].materials['orthotropic'].Depvar(n=10)
mdb.models['Model-1'].materials['orthotropic'].UserMaterial(
    mechanicalConstants=(0.0, 0.000496, 7.209, 0.015193, 20.417,
        0.003283,
    11.176, 0.000662, 9.466, 100000.0))


#----------------------Create section----------------------

mdb.models['Model-1'].HomogeneousSolidSection(material='
    orthotropic', name=
    'Section-1', thickness=None)
mdb.models['Model-1'].parts['TEM-4layers'].SectionAssignment(
    offset=0.0,
    offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
    elements=mdb.models['Model-1'].parts['TEM-4layers'].elements.
        getSequenceFromMask(
    mask=('[#ffffffff:274 ]', ), )), sectionName='Section-1',
    thicknessAssignment=FROM_SECTION)


#----------------------Assign material orientation
    ----------------------

mdb.models['Model-1'].parts['TEM-4layers'].MaterialOrientation(
    additionalRotationType=ROTATION_NONE, fieldName='', localCsys=
        None,
    orientationType=USER, region=Region(
```

```
    elements=mdb.models['Model-1'].parts['TEM-4layers'].elements.
        getSequenceFromMask(
    mask=('[#ffffffff:274 ]', ), )), stackDirection=STACK_3)


#----------------------Create assembly----------------------

mdb.models['Model-1'].rootAssembly.DatumCsysByDefault(CARTESIAN)
mdb.models['Model-1'].rootAssembly.Instance(dependent=ON, name='
    TEM-4layers-1',
    part=mdb.models['Model-1'].parts['TEM-4layers'])
mdb.models['Model-1'].rootAssembly.rotate(angle=90.0,
    axisDirection=(1.0, 0.0,
    0.0), axisPoint=(0.0, 0.0, 0.0), instanceList=('TEM-4layers-1'
        , ))


#----------------------Create step----------------------


mdb.models['Model-1'].StaticStep(adaptiveDampingRatio=None,
    continueDampingFactors=False, initialInc=0.001, maxInc=0.2,
        maxNumInc=20000
    , minInc=1e-15, name='static', nlgeom=ON, previous='Initial',
    stabilizationMagnitude=1e-08, stabilizationMethod=
        DAMPING_FACTOR,
    timePeriod=0.2)
mdb.models['Model-1'].steps['static'].control.setValues(
    allowPropagation=OFF,
    resetDefaultValues=OFF, timeIncrementation=(4.0, 8.0, 9.0,
        16.0, 10.0, 4.0,
    12.0, 10.0, 6.0, 3.0, 50.0))


#----------------------Request output----------------------

mdb.models['Model-1'].fieldOutputRequests['F-Output-1'].setValues(
    numIntervals=
    50, timeMarks=OFF, variables=('S', 'PE', 'PEEQ', 'PEMAG', 'LE'
        , 'U', 'RF',
    'CF', 'SDV'))
mdb.models['Model-1'].historyOutputRequests['H-Output-1'].
    setValues(
    numIntervals=50, timeMarks=OFF)


#----------------------Apply loads and BCs----------------------

mdb.models['Model-1'].TabularAmplitude(data=((0.0, 0.0), (0.01,
    2.3556), (0.02,
```

```
    4.5224), (0.03, 6.5004), (0.04, 8.2896), (0.05, 9.89), (0.06,
        11.3016), (
    0.07, 12.5244), (0.08, 13.5584), (0.09, 14.4036), (0.1, 15.06)
        , (0.11,
    15.5276), (0.12, 15.8064), (0.13, 15.8964), (0.14, 15.7976),
        (0.15, 15.51),
    (0.16, 15.0336), (0.17, 14.3684), (0.18, 13.5144), (0.19,
        12.4716), (0.2,
    11.24)), name='systolicAmp', smooth=SOLVER_DEFAULT, timeSpan=
        TOTAL)
mdb.models['Model-1'].Pressure(amplitude='systolicAmp',
    createStepName='static'
    , distributionType=UNIFORM, field='', magnitude=0.001, name=
    'systolicPressure', region=
    mdb.models['Model-1'].rootAssembly.instances['TEM-4layers-1'].
        surfaces['endoSurf'])
mdb.models['Model-1'].rootAssembly.DatumCsysByThreePoints(
    coordSysType=
    CYLINDRICAL, name='CyldricalCSYS', origin=(0.0, 0.0, 20.0),
        point1=(1.0,
    0.0, 20.0), point2=(0.0, 1.0, 20.0))
mdb.models['Model-1'].DisplacementBC(amplitude=UNSET,
    createStepName='static',
    distributionType=UNIFORM, fieldName='', fixed=OFF, localCsys=
    mdb.models['Model-1'].rootAssembly.datums[4], name='basalBC',
        region=
    mdb.models['Model-1'].rootAssembly.instances['TEM-4layers-1'].
        sets['basalNodes']
    , u1=UNSET, u2=UNSET, u3=0.0, ur1=UNSET, ur2=UNSET, ur3=UNSET)

#The following BC, restrcting radial movement in the apex node can
     be used if,
#problems with rigid body motion occurs.
mdb.models['Model-1'].DisplacementBC(amplitude=UNSET,
    createStepName='static',
    distributionType=UNIFORM, fieldName='', fixed=OFF, localCsys=
    mdb.models['Model-1'].rootAssembly.datums[4], name='apexBC',
        region=
    mdb.models['Model-1'].rootAssembly.instances['TEM-4layers-1'].
        sets['apexNodes']
    , u1=0.0, u2=UNSET, u3=UNSET, ur1=UNSET, ur2=UNSET, ur3=UNSET)
```

B Write ORIENT Subroutine

```
clear all
close all

%This MATLAB script writes an ORIENT user subroutine for Abaqus
    used
%in the left ventricle truncated ellipsoid model based on the
    Abaqus input file.
%It assumes that the Abaqus model is partitioned transmurally and
    that
%the element sets defined containing the element for each layer.
    Using this
%information different fiber and sheet angles are given to each
    layer.

%Defining number of layers, fiber and sheet angles

nLayers = 4;

angleAmp = 60*pi()/180;
dAngle = 2*angleAmp/(nLayers-1);

l1angle = -angleAmp;
l2angle = -angleAmp + dAngle;
l3angle = -angleAmp + 2*dAngle;
l4angle = -angleAmp + 3*dAngle;

sheetProjection = 45*pi()/180;
dSheetProjection = 2*sheetProjection/(nLayers-1);

sheetl1angle = atan2(tan(-sheetProjection),cos(l1angle));
sheetl2angle = atan2(tan(-sheetProjection+dSheetProjection),cos(
    l2angle));
sheetl3angle = atan2(tan(-sheetProjection+2*dSheetProjection),cos(
    l3angle));
sheetl4angle = atan2(tan(-sheetProjection+3*dSheetProjection),cos(
    l4angle));

%Finding the line numbers for the relevant lines in the input file
inputFile = fopen('TEM_4layers_active_fn100_f60_2.inp','rt');
counter = 0;
while 1
    tline = fgetl(inputFile);
    counter = counter + 1;
    if ischar(tline)
        U = strfind(tline, '*Node');
        if isfinite(U) == 1;
            nodeLine = counter;
```

```
        end
        U = strfind(tline, '*Element');
        if isfinite(U) == 1;
            elsLine = counter;
        end
        U = strfind(tline, 'elset=Layer1');
        if isfinite(U) == 1;
            layer1line = counter;
        end
        U = strfind(tline, 'elset=Layer2');
        if isfinite(U) == 1;
            layer2line = counter;
        end
        U = strfind(tline, 'elset=Layer3');
        if isfinite(U) == 1;
            layer3line = counter;
        end
        U = strfind(tline, 'elset=Layer4');
        if isfinite(U) == 1;
            layer4line = counter;
        end
        U = strfind(tline, 'nset=basalNodes');
        if isfinite(U) == 1;
            layer4stop = counter;
            break
        end
    end
end
fclose(inputFile);

nnodes=elsLine-nodeLine-1;
noel=layer1line-elsLine-1;


%Reading the data from the input file
nodeCoor = csvread('TEM_4layers_active_fn100_f60_2.inp',nodeLine,
    1, [nodeLine,1,nodeLine+nnodes-1,3]);

%Rotating coordinates the same way the instance is rotated in
    Abaqus, 90
%degrees about x-axis
tempY = nodeCoor(:,2);
tempZ = nodeCoor(:,3);

nodeCoor(:,2) = cos(pi()/2)*tempY(:)-sin(pi()/2)*tempZ(:);
nodeCoor(:,3) = sin(pi()/2)*tempY(:)+cos(pi()/2)*tempZ(:);

elements = csvread('TEM_4layers_active_fn100_f60_2.inp',elsLine,
    1, [elsLine,1,elsLine+noel-1,8]);
```

```
layer1 = csvread('TEM_4layers_active_fn100_f60_2.inp',layer1line,
    0, [layer1line,0,layer2line-2,15]);
layer1 = sort(layer1(:));
layer1(layer1==0) = [];

layer2 = csvread('TEM_4layers_active_fn100_f60_2.inp',layer2line,
    0, [layer2line,0,layer3line-2,15]);
layer2 = sort(layer2(:));
layer2(layer2==0) = [];

layer3 = csvread('TEM_4layers_active_fn100_f60_2.inp',layer3line,
    0, [layer3line,0,layer4line-2,15]);
layer3 = sort(layer3(:));
layer3(layer3==0) = [];

layer4 = csvread('TEM_4layers_active_fn100_f60_2.inp',layer4line,
    0, [layer4line,0,layer4stop-2,15]);
layer4 = sort(layer4(:));
layer4(layer4==0) = [];

%Ordering the relevant angles into lists according to the element
    numbering

elementAngleList = zeros(noel,1);
elementSheetAngleList = zeros(noel,1);

for i=1:numel(layer1)
    elementAngleList(layer1(i)) = l1angle;
    elementSheetAngleList(layer1(i)) = sheetl1angle;
end
for i=1:numel(layer2)
    elementAngleList(layer2(i)) = l2angle;
    elementSheetAngleList(layer2(i)) = sheetl2angle;
end
for i=1:numel(layer3)
    elementAngleList(layer3(i)) = l3angle;
    elementSheetAngleList(layer3(i)) = sheetl3angle;
end
for i=1:numel(layer4)
    elementAngleList(layer4(i)) = l4angle;
    elementSheetAngleList(layer4(i)) = sheetl4angle;
end

%Declaring variables
originalCoor = zeros(8,3);
elsCenteroid = zeros(noel,3);
f = zeros(noel,3);
s = zeros(noel,3);
n = zeros(noel,3);
T = zeros(3,3,noel);
```

```
phi = zeros(noel);
theta = zeros(noel);
R1= zeros(3,3);
R2= zeros(3,3);

for i=1:noel
    for j=1:8
        originalCoor(j,:) = nodeCoor(elements(i,j),:);

    end
    %Centeroids are only calculated for plotting purposes
    for k=1:8
        elsCenteroid(i,1) = elsCenteroid(i,1)+originalCoor(k,1);
        elsCenteroid(i,2) = elsCenteroid(i,2)+originalCoor(k,2);
        elsCenteroid(i,3) = elsCenteroid(i,3)+originalCoor(k,3);
    end
    elsCenteroid(i,1) = elsCenteroid(i,1)/8;
    elsCenteroid(i,2) = elsCenteroid(i,2)/8;
    elsCenteroid(i,3) = elsCenteroid(i,3)/8;

    %Defining epi/endocardium plane by two vectors in the 1-2-3-4
        plane
    v1=[originalCoor(1,1)-originalCoor(3,1) originalCoor(1,2)-
        originalCoor(3,2) originalCoor(1,3)-originalCoor(3,3)];
    v2=[originalCoor(2,1)-originalCoor(4,1) originalCoor(2,2)-
        originalCoor(4,2) originalCoor(2,3)-originalCoor(4,3)];


    %The vector defining the s0-axis is the normalvector to the
        plane and
    %thus the cross product of the two vectors
    s0temp=cross(v1,v2);

    s0(1)=s0temp(1)/sqrt(s0temp(1)^2+s0temp(2)^2+s0temp(3)^2);
    s0(2)=s0temp(2)/sqrt(s0temp(1)^2+s0temp(2)^2+s0temp(3)^2);
    s0(3)=s0temp(3)/sqrt(s0temp(1)^2+s0temp(2)^2+s0temp(3)^2);

    %Calculate relevant angles
    phi(i) = atan2(s0(2),s0(1));

    %Define inital v0-axis as a 90 degree rotation of s0 with no
        fiberangle
    f0(1)=cos(phi(i)+pi()/2);
    f0(2)=sin(phi(i)+pi()/2);
    f0(3)=0;

    %Define inital n0 axis as cross-product of s0 and v0
    f0temp=[f0(1) f0(2) f0(3)];

    n0temp=cross(f0temp,s0temp);
```

```
n0(1)=n0temp(1);
n0(2)=n0temp(2);
n0(3)=n0temp(3);

%Inital coordinate system

Tinital = [f0(1) f0(2) f0(3); s0(1) s0(2) s0(3); n0(1) n0(2)
    n0(3)];

%Rotation matrix to rotate inital coordinate system about f0
    axis with
%an angle alpha
alpha = elementSheetAngleList(i);
R1 = [cos(alpha)+f0(1)^2*(1-cos(alpha)) f0(1)*f0(2)*(1-cos(
    alpha))-f0(3)*sin(alpha) f0(1)*f0(3)*(1-cos(alpha))+f0(2)*
    sin(alpha);
    f0(2)*f0(1)*(1-cos(alpha))+f0(3)*sin(alpha) cos(alpha)+f0
        (2)^2*(1-cos(alpha)) f0(2)*f0(3)*(1-cos(alpha))-f0(1)*
        sin(alpha);
    f0(3)*f0(1)*(1-cos(alpha))-f0(2)*sin(alpha) f0(3)*f0(2)
        *(1-cos(alpha))+f0(1)*sin(alpha) cos(alpha)+f0(3)
        ^2*(1-cos(alpha))];

%Rotate coordinate system

T1=Tinital*R1;

%Rotation matrix to rotate inital coordinate system about s0
    axis with
%an angle beta
beta = elementAngleList(i);
R2 = [cos(beta)+s0(1)^2*(1-cos(beta)) s0(1)*s0(2)*(1-cos(beta)
    )-s0(3)*sin(beta) s0(1)*s0(3)*(1-cos(beta))+s0(2)*sin(beta
    );
    s0(2)*s0(1)*(1-cos(beta))+s0(3)*sin(beta) cos(beta)+s0(2)
        ^2*(1-cos(beta)) s0(2)*s0(3)*(1-cos(beta))-s0(1)*sin(
        beta);
    s0(3)*s0(1)*(1-cos(beta))-s0(2)*sin(beta) s0(3)*s0(2)*(1-
        cos(beta))+s0(1)*sin(beta) cos(beta)+s0(3)^2*(1-cos(
        beta))];

%Rotate coordinate system

T(:,:,i)=T1*R2;

%Extract vectors from T-matrix for plotting
f(i,1)=T(1,1,i);
f(i,2)=T(1,2,i);
f(i,3)=T(1,3,i);
```

```
    s(i,1)=T(2,1,i);
    s(i,2)=T(2,2,i);
    s(i,3)=T(2,3,i);
    n(i,1)=T(3,1,i);
    n(i,2)=T(3,2,i);
    n(i,3)=T(3,3,i);

end

%WRITE FORTRAN SUBROUTINE
orientFile = fopen('ORIENT_4layers_f60_s45_NOTVALID.for','w');
fprintf(orientFile, '%s\n', '      SUBROUTINE ORIENT(T,NOEL,NPT,
    LAYER,KSPT,COORDS,BASIS,');
fprintf(orientFile, '%s\n', '     1 NNODES,CNODES,JNNUM)');
fprintf(orientFile, '%s\n', 'C');
fprintf(orientFile, '%s\n', '       INCLUDE ''ABA_PARAM.INC''');
fprintf(orientFile, '%s\n', 'C');
fprintf(orientFile, '%s\n', '       CHARACTER*80 ORNAME');
fprintf(orientFile, '%s\n', 'C');
fprintf(orientFile, '%s\n', '       DIMENSION T(3,3),COORDS(3),
    BASIS(3,3),CNODES(3,NNODES)');
fprintf(orientFile, '%s\n', '       DIMENSION JNNUM(NNODES)');

for i=1:noel
    fprintf(orientFile, '%s%d%s\n', '       IF (NOEL == ',i,') THEN
        ');

    fprintf(orientFile, '%s%d\n', '         T(1,1) = ',T(1,1,i));
    fprintf(orientFile, '%s%d\n', '         T(2,1) = ',T(1,2,i));
    fprintf(orientFile, '%s%d\n', '         T(3,1) = ',T(1,3,i));
    fprintf(orientFile, '%s%d\n', '         T(1,2) = ',T(2,1,i));
    fprintf(orientFile, '%s%d\n', '         T(2,2) = ',T(2,2,i));
    fprintf(orientFile, '%s%d\n', '         T(3,2) = ',T(2,3,i));
    fprintf(orientFile, '%s%d\n', '         T(1,3) = ',T(3,1,i));
    fprintf(orientFile, '%s%d\n', '         T(2,3) = ',T(3,2,i));
    fprintf(orientFile, '%s%d\n', '         T(3,3) = ',T(3,3,i));

    fprintf(orientFile, '%s\n', '       ENDIF');
end

fprintf(orientFile, '%s\n', '       RETURN');
fprintf(orientFile, '%s\n', '       END');


%The rest of the code are only plotting for the purpose of visual
%confirmation of correct implementation of fiber angle field.

%Arrange all the coordinates for the element centeroid in theire
    respective
%layers
```

```
layer1Coor=zeros(numel(layer1),3);
layer2Coor=zeros(numel(layer2),3);
layer3Coor=zeros(numel(layer3),3);
layer4Coor=zeros(numel(layer4),3);

for i=1:numel(layer1)
layer1Coor(i,1)=elsCenteroid(layer1(i),1);
layer1Coor(i,2)=elsCenteroid(layer1(i),2);
layer1Coor(i,3)=elsCenteroid(layer1(i),3);
end
for i=1:numel(layer2)
layer2Coor(i,1)=elsCenteroid(layer2(i),1);
layer2Coor(i,2)=elsCenteroid(layer2(i),2);
layer2Coor(i,3)=elsCenteroid(layer2(i),3);
end
for i=1:numel(layer3)
layer3Coor(i,1)=elsCenteroid(layer3(i),1);
layer3Coor(i,2)=elsCenteroid(layer3(i),2);
layer3Coor(i,3)=elsCenteroid(layer3(i),3);
end
for i=1:numel(layer4)
layer4Coor(i,1)=elsCenteroid(layer4(i),1);
layer4Coor(i,2)=elsCenteroid(layer4(i),2);
layer4Coor(i,3)=elsCenteroid(layer4(i),3);
end

for i=1:numel(layer4)
layer3fvector(i,1)=s(layer3(i),1);
layer3fvector(i,2)=s(layer3(i),2);
layer3fvector(i,3)=s(layer3(i),3);
end

for i=1:numel(layer4)
layer3svector(i,1)=s(layer3(i),1);
layer3svector(i,2)=s(layer3(i),2);
layer3svector(i,3)=s(layer3(i),3);
end

%Plotting  the s and f vectors for layer3
figure()
hold on
quiver3(layer3Coor(:,1), layer3Coor(:,2), layer3Coor(:,3),
    layer3fvector(:,1),layer3fvector(:,2),layer3fvector(:,3),'r',
    'AutoScaleFactor', 2)
xlabel('X')
ylabel('Y')
zlabel('Z')
hold off

figure()
```

```
hold on
quiver3(layer3Coor(:,1), layer3Coor(:,2), layer3Coor(:,3),
    layer3svector(:,1),layer3svector(:,2),layer3svector(:,3),'b',
    'AutoScaleFactor', 2)
xlabel('X')
ylabel('Y')
zlabel('Z')
hold off
```

C UMAT Subroutine

```
C This Abaqus UMAT user subroutine implements Holzapfels
    structurally based
C constitutive material law. The passive part was implemented by
    Victorien Prot
C and is is unchanged in this thesis. The implentation of the
    active contraction
C is done by adding to the vairable PSI4f and the addition of the
    new vairables
C PSI6 and PSI6sn. These are futher used in the stress calculation
    .

         SUBROUTINE UMAT(STRESS,STATEV,DDSDDE,SSE,SPD,SCD,
     1 RPL,DDSDDT,DRPLDE,DRPLDT,STRAN,DSTRAN,
     2 TIME,DTIME,TEMP,DTEMP,PREDEF,DPRED,MATERL,NDI,NSHR,NTENS,
     3 NSTATV,PROPS,NPROPS,COORDS,DROT,PNEWDT,CELENT,
     4 DFGRD0,DFGRD1,NOEL,NPT,KSLAY,KSPT,KSTEP,KINC)
C
      INCLUDE 'ABA_PARAM.INC'
C
      CHARACTER*8 MATERL
      DIMENSION STRESS(NTENS),STATEV(NSTATV),
     1 DDSDDE(NTENS,NTENS),DDSDDT(NTENS),DRPLDE(NTENS),
     2 STRAN(NTENS),DSTRAN(NTENS),DFGRD0(3,3),DFGRD1(3,3),
     3 TIME(2),PREDEF(1),DPRED(1),PROPS(NPROPS),COORDS(3),DROT
        (3,3)
C

C    LOCAL ARRAYS
C ----------------------------------------------------------------
C    BBAR   - DEVIATORIC RIGHT CAUCHY-GREEN TENSOR
C    DISTGR - DEVIATORIC DEFORMATION GRADIENT (DISTORTION TENSOR)
C ----------------------------------------------------------------
C
      DIMENSION BBAR(6),DISTGR(3,3), BB(3,3), bf0(3), bf(3),
          SIGBAR(3,3)
      DIMENsION CB(3,3), SIGISO(3,3),devB(3,3), devff(3,3)
      DIMENSION devss(3,3), devfs(3,3),bs0(3), bs(3), bn0(3), bn
          (3)
C
      PARAMETER(ZERO=0.D0, ONE=1.D0, TWO=2.D0, THREE=3.D0, FOUR=4.
          D0)
C
C ----------------------------------------------------------------

C ----------------------------------------------------------------
C
C    ELASTIC PROPERTIES
```

```
C

        STATEV(1)=0.
        STATEV(2)=0.
        STATEV(3)=0.
        STATEV(4)=0.
        STATEV(5)=0.
        STATEV(6)=0.
        STATEV(7)=0.
        STATEV(8)=0.

        STATEV(10)=0.




      C10=PROPS(1)

      a=PROPS(2)
        b=PROPS(3)

        af=PROPS(4)
        bff=PROPS(5)

        as=PROPS(6)
        bss=PROPS(7)

        afs=PROPS(8)
        bfs=PROPS(9)


        D11 =PROPS(10)


C
C     JACOBIAN AND DISTORTION TENSOR
C
      DET=DFGRD1(1, 1)*DFGRD1(2, 2)*DFGRD1(3, 3)
     1   -DFGRD1(1, 2)*DFGRD1(2, 1)*DFGRD1(3, 3)
      IF(NSHR.EQ.3) THEN
        DET=DET+DFGRD1(1, 2)*DFGRD1(2, 3)*DFGRD1(3, 1)
     1         +DFGRD1(1, 3)*DFGRD1(3, 2)*DFGRD1(2, 1)
     2         -DFGRD1(1, 3)*DFGRD1(3,1)*DFGRD1(2, 2)
     3         -DFGRD1(2, 3)*DFGRD1(3, 2)*DFGRD1(1, 1)
      END IF
```

```
      SCALE=DET**(-ONE/THREE)
      DO K1=1, 3
        DO K2=1, 3
          DISTGR(K2, K1)=SCALE*DFGRD1(K2, K1)
        END DO
      END DO

        STATEV(1)=DET
        STATEV(5)=DISTGR(1,2)-DISTGR(2,1)
        STATEV(6)=DISTGR(1, 3)-DISTGR(3,1)
        STATEV(7)=DISTGR(2,3)-DISTGR(3,2)
        STATEV(8)=DISTGR(1,1)
        STATEV(9)=DISTGR(2,2)
        STATEV(10)=DISTGR(3,3)




C
C     CALCULATE LEFT CAUCHY-GREEN TENSOR
C
      BBAR(1)=DISTGR(1, 1)**2+DISTGR(1, 2)**2+DISTGR(1, 3)**2
      BBAR(2)=DISTGR(2, 1)**2+DISTGR(2, 2)**2+DISTGR(2, 3)**2
      BBAR(3)=DISTGR(3, 3)**2+DISTGR(3, 1)**2+DISTGR(3, 2)**2
      BBAR(4)=DISTGR(1, 1)*DISTGR(2, 1)+DISTGR(1, 2)*DISTGR(2, 2)
     1       +DISTGR(1, 3)*DISTGR(2, 3)
      IF(NSHR.EQ.3) THEN
        BBAR(5)=DISTGR(1, 1)*DISTGR(3, 1)+DISTGR(1, 2)*DISTGR(3,
            2)
     1         +DISTGR(1, 3)*DISTGR(3, 3)
        BBAR(6)=DISTGR(2, 1)*DISTGR(3, 1)+DISTGR(2, 2)*DISTGR(3,
            2)
     1         +DISTGR(2, 3)*DISTGR(3, 3)
      END IF
C
C     CALCULATE THE STRESS
C
      TRBBAR=(BBAR(1)+BBAR(2)+BBAR(3))/THREE
      EG=TWO*C10/DET
      EK=TWO*D11*(TWO*DET-ONE)
      PR=TWO*D11*(DET-ONE)
c       EK=D1*Log(sqrt(DET))+0.5*D1
c       PR=D1*Log(sqrt(DET))



      DO K1=1,NDI
        STRESS(K1)=EG*(BBAR(K1)-TRBBAR)+PR
      END DO
```

```
      DO K1=NDI+1,NDI+NSHR
        STRESS(K1)=EG*BBAR(K1)
      END DO
C
C     CALCULATE THE STIFFNESS
C
      EG23=EG*TWO/THREE

      DDSDDE(1, 1)= EG23*(BBAR(1)+TRBBAR)+EK
      DDSDDE(2, 2)= EG23*(BBAR(2)+TRBBAR)+EK
      DDSDDE(3, 3)= EG23*(BBAR(3)+TRBBAR)+EK
      DDSDDE(1, 2)=-EG23*(BBAR(1)+BBAR(2)-TRBBAR)+EK
      DDSDDE(1, 3)=-EG23*(BBAR(1)+BBAR(3)-TRBBAR)+EK
      DDSDDE(2, 3)=-EG23*(BBAR(2)+BBAR(3)-TRBBAR)+EK
      DDSDDE(1, 4)= EG23*BBAR(4)/TWO
      DDSDDE(2, 4)= EG23*BBAR(4)/TWO
      DDSDDE(3, 4)=-EG23*BBAR(4)
      DDSDDE(4, 4)= EG*(BBAR(1)+BBAR(2))/TWO
      IF(NSHR.EQ.3) THEN
       DDSDDE(1, 5)= EG23*BBAR(5)/TWO
        DDSDDE(2, 5)=-EG23*BBAR(5)
        DDSDDE(3, 5)= EG23*BBAR(5)/TWO
        DDSDDE(1, 6)=-EG23*BBAR(6)
        DDSDDE(2, 6)= EG23*BBAR(6)/TWO
        DDSDDE(3, 6)= EG23*BBAR(6)/TWO
        DDSDDE(5, 5)= EG*(BBAR(1)+BBAR(3))/TWO
        DDSDDE(6, 6)= EG*(BBAR(2)+BBAR(3))/TWO
        DDSDDE(4,5)= EG*BBAR(6)/TWO
        DDSDDE(4,6)= EG*BBAR(5)/TWO
        DDSDDE(5,6)= EG*BBAR(4)/TWO
      END IF


CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC



      BB(1,1) = DISTGR(1,1) ** 2 + DISTGR(1,2) ** 2 + DISTGR(1,3)
          ** 2

      BB(1,2) = DISTGR(1,1) * DISTGR(2,1) + DISTGR(1,2) * DISTGR
          (2,2) +
     # DISTGR(1,3) * DISTGR(2,3)

      BB(1,3) = DISTGR(1,1) * DISTGR(3,1) + DISTGR(1,2) * DISTGR
          (3,2) +
     # DISTGR(1,3) * DISTGR(3,3)

      BB(2,1) = DISTGR(1,1) * DISTGR(2,1) + DISTGR(1,2) * DISTGR
          (2,2) +
```

```
     # DISTGR(1,3) * DISTGR(2,3)

      BB(2,2) = DISTGR(2,1) ** 2 + DISTGR(2,2) ** 2 + DISTGR(2,3)
          ** 2

      BB(2,3) = DISTGR(2,1) * DISTGR(3,1) + DISTGR(2,2) * DISTGR
          (3,2) +
     # DISTGR(2,3) * DISTGR(3,3)

      BB(3,1) = DISTGR(1,1) * DISTGR(3,1) + DISTGR(1,2) * DISTGR
          (3,2) +
     # DISTGR(1,3) * DISTGR(3,3)

      BB(3,2) = DISTGR(2,1) * DISTGR(3,1) + DISTGR(2,2) * DISTGR
          (3,2) +
     # DISTGR(2,3) * DISTGR(3,3)

      BB(3,3) = DISTGR(3,1) ** 2 + DISTGR(3,2) ** 2 + DISTGR(3,3)
          ** 2

CCCCCC

        CB(1,1) = DISTGR(1,1) ** 2 + DISTGR(2,1) ** 2 + DISTGR
            (3,1) ** 2

      CB(1,2) = DISTGR(1,1) * DISTGR(1,2) + DISTGR(2,1) * DISTGR
          (2,2) +
     # DISTGR(3,1) * DISTGR(3,2)

      CB(1,3) = DISTGR(1,1) * DISTGR(1,3) + DISTGR(2,1) * DISTGR
          (2,3) +
     # DISTGR(3,1) * DISTGR(3,3)

      CB(2,1) = DISTGR(1,1) * DISTGR(1,2) + DISTGR(2,1) * DISTGR
          (2,2) +
     # DISTGR(3,1) * DISTGR(3,2)

      CB(2,2) = DISTGR(1,2) ** 2 + DISTGR(2,2) ** 2 + DISTGR(3,2)
          ** 2

      CB(2,3) = DISTGR(1,2) * DISTGR(1,3) + DISTGR(2,2) * DISTGR
          (2,3) +
     # DISTGR(3,2) * DISTGR(3,3)

      CB(3,1) = DISTGR(1,1) * DISTGR(1,3) + DISTGR(2,1) * DISTGR
          (2,3) +
     # DISTGR(3,1) * DISTGR(3,3)

      CB(3,2) = DISTGR(1,2) * DISTGR(1,3) + DISTGR(2,2) * DISTGR
          (2,3) +
```

```
     # DISTGR(3,2) * DISTGR(3,3)

      CB(3,3) = DISTGR(1,3) ** 2 + DISTGR(2,3) ** 2 + DISTGR(3,3)
          ** 2


CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

C INVARIANTS

        bf0(1)=1
        bf0(2)=0
        bf0(3)=0

        bf(1)=bf0(1)*DISTGR(1,1)+bf0(2)*DISTGR(1,2)+bf0(3)*DISTGR
            (1,3)
        bf(2)=bf0(1)*DISTGR(2,1)+bf0(2)*DISTGR(2,2)+bf0(3)*DISTGR
            (2,3)
        bf(3)=bf0(1)*DISTGR(3,1)+bf0(2)*DISTGR(3,2)+bf0(3)*DISTGR
            (3,3)

        BI1=CB(1,1)+CB(2,2)+CB(3,3)

        BI4f=bf0(1) * (CB(1,1) * bf0(1)+ CB(1,2)*bf0(2) + CB(1,3)*
             bf0(3)
     #) + bf0(2) * (CB(2,1) * bf0(1) + CB(2,2)*bf0(2) + CB(2,3)*
         bf0(3))
     #+ bf0(3) * (CB(3,1) * bf0(1) + CB(3,2)*bf0(2) + CB(3,3)*bf0
         (3))

        bs0(1)=0
        bs0(2)=1
        bs0(3)=0

        bs(1)=bs0(1)*DISTGR(1,1)+bs0(2)*DISTGR(1,2)+bs0(3)*DISTGR
            (1,3)
        bs(2)=bs0(1)*DISTGR(2,1)+bs0(2)*DISTGR(2,2)+bs0(3)*DISTGR
            (2,3)
        bs(3)=bs0(1)*DISTGR(3,1)+bs0(2)*DISTGR(3,2)+bs0(3)*DISTGR
            (3,3)

        BI4s=bs0(1) * (CB(1,1) * bs0(1) + CB(1,2)*bs0(2) + CB(1,3)
            *bs0(3)
     #) + bs0(2) * (CB(2,1) * bs0(1) + CB(2,2)*bs0(2) + CB(2,3)*
         bs0(3))
     #+ bs0(3) * (CB(3,1) * bs0(1) + CB(3,2) * bs0(2) + CB(3,3)*
         bs0(3))

        BI8=bf0(1) * (CB(1,1) * bs0(1) + CB(1,2) * bs0(2) + CB
            (1,3) * bs
```

```
    #0(3)) + bf0(2) * (CB(2,1) * bs0(1) + CB(2,2) * bs0(2) + CB
        (2,3) *
    #bs0(3)) + bf0(3) * (CB(3,1) * bs0(1) + CB(3,2) * bs0(2) + CB
        (3,3)
    #* bs0(3))


       STATEV(2)=ACOSD((bs(1)*bf(1)+bs(2)*bf(2)+bs(3)*bf(3))
    #/(SQRT(bs(1)*bs(1)+bs(2)*bs(2)+bs(3)*bs(3))*
    #SQRT(bf(1)*bf(1)+bf(2)*bf(2)+bf(3)*bf(3))))

       STATEV(3)=BI4f
       STATEV(4)=BI4s

       bn0(1)=0
       bn0(2)=0
       bn0(3)=1

       bn(1)=bn0(1)*DISTGR(1,1)+bn0(2)*DISTGR(1,2)+bn0(3)*DISTGR
           (1,3)
       bn(2)=bn0(1)*DISTGR(2,1)+bn0(2)*DISTGR(2,2)+bn0(3)*DISTGR
           (2,3)
       bn(3)=bn0(1)*DISTGR(3,1)+bn0(2)*DISTGR(3,2)+bn0(3)*DISTGR
           (3,3)


CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

C Material law

c       a=0.496
c       b=7.209
c       af=15.193
c       bff=20.417
c       as=3.283
c       bss=11.176
c       afs=0.662
c       bfs=9.466

C Active contraction
c Set Tm=100kPa=0.1MPa
      TM=0.1
      DMAXTIME=0.2
c Function governing stress at time
      IF (TIME(2)<DMAXTIME) THEN
        F=(TIME(2)+DTIME)/DMAXTIME
      ELSE
        F=1
      ENDIF
```

```
      PSI1=a * exp(dble(b * (BI1 - 3))) / 0.2D1
      PSI4f=af * dble(BI4f - 1) * exp(dble(bff * (BI4f - 1) **
         2)) +
      1(TM*F)/0.2D1
      STATEV(10)=PSI4F
      PSI4s=as * dble(BI4s - 1) * exp(dble(bss * (BI4s - 1) **
         2))
      PSI8=afs * BI8 * exp(bfs * BI8 ** 2)

      PSI11=a * dble(b) * exp(dble(b * (BI1 - 3))) / 0.2D1


      PSI14f=0
      PSI14s=0
      PSI18=0


      PSI44f=af*exp(dble(bff * (BI4f - 1) ** 2)) + 0.2D1 * af *
         dble((BI
     #4f - 1) ** 2) * dble(bff) * exp(dble(bff * (BI4f - 1) ** 2))

      PSI44s=as*exp(dble(bss * (BI4s - 1) ** 2)) + 0.2D1 * as *
         dble((BI
     #4s - 1) ** 2) * dble(bss) * exp(dble(bss * (BI4s - 1) ** 2))


      PSI88=afs*exp(bfs * BI8 ** 2) + 0.2D1 * afs * BI8 ** 2 *
         bfs * e
     #xp(bfs * BI8 ** 2)

    PSI6 = 0.6*(TM*F)/0.2D1

    PSI6sn = 0.03*(TM*F)/0.2D1

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC



CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

C STRESS CALCULATION

      SIGBAR(1,1) = 2 * PSI1 * BB(1,1) + 2 * PSI4f*bf(1)**2 +2*
         PSI4s*bs(
     #1) ** 2 + 2 * PSI8 * bf(1) * bs(1) + 2 * PSI6 * bn(1) * bn
        (1) +
     #0.2D1 * 0.2D1 * PSI6sn * bs(1) * bn(1)
```

```
      STATEV(11) = SIGBAR(1,1)
      STATEV(12) = PSI1
      STATEV(13) = PSI4s
      STATEV(14) = PSI8


      SIGBAR(1,2) = dble(2*PSI1 * BB(1,2))+dble(2 * PSI4f*bf(1)*bf
          (2))+
     # dble(2 * PSI4s * bs(1) * bs(2)) + 0.2D1 * PSI8 * (dble(bf
         (1) * bs
     #(2)) / 0.2D1 + dble(bs(1) * bf(2)) / 0.2D1) + 0.2D1 * PSI6 *
          bn(1)
     # * bn(2) + 0.2D1 * PSI6sn * (dble(bs(1) * bn(2)) / 0.2D1 +
         dble(bn
     #(1) * bs(2)) / 0.2D1)

      SIGBAR(1,3) = dble(2 *PSI1*BB(1,3)) + dble(2*PSI4f*bf(1)* bf
          (3)) +
     # dble(2 * PSI4s * bs(1) * bs(3)) + 0.2D1 * PSI8 * (dble(bf
         (1) * bs
     #(3)) / 0.2D1 + dble(bs(1) * bf(3)) / 0.2D1) + 0.2D1 * PSI6 *
          bn(1)
     # * bn(3) + 0.2D1 * PSI6sn * (dble(bs(1) * bn(3)) / 0.2D1 +
         dble(bn
     #(1) * bs(3)) / 0.2D1)

      SIGBAR(2,1) = dble(2 * PSI1*BB(2,1)) + dble(2*PSI4f*bf(1)*bf
          (2)) +
     # dble(2 * PSI4s * bs(1) * bs(2)) + 0.2D1 * PSI8 * (dble(bf
         (1) * bs
     #(2)) / 0.2D1 + dble(bs(1) * bf(2)) / 0.2D1) + 0.2D1 * PSI6 *
          bn(2)
     # * bn(1) + 0.2D1 * PSI6sn * (dble(bs(2) * bn(1)) / 0.2D1 +
         dble(bn
     #(2) * bs(1)) / 0.2D1)

      SIGBAR(2,2) =2 * PSI1*BB(2,2) + 2*PSI4f*bf(2) ** 2 + 2*PSI4s
           * bs
     #(2) ** 2 + 2 * PSI8 * bf(2) * bs(2) + 2 * PSI6 * bn(2) * bn
         (2) + 2
     # * PSI6sn * bs(2) * bn(2)

      STATEV(15) = SIGBAR(2,2)

      SIGBAR(2,3) = dble(2 * PSI1*BB(2,3)) + dble(2*PSI4f*bf(2)*bf
          (3)) +
     # dble(2 * PSI4s * bs(2) * bs(3)) + 0.2D1 * PSI8 * (dble(bf
         (2) * bs
     #(3)) / 0.2D1 + dble(bs(2) * bf(3)) / 0.2D1) + 0.2D1 * PSI6 *
          bn(2)
```

```
# * bn(3) + 0.2D1 * PSI6sn * (dble(bs(2) * bn(3)) / 0.2D1 +
    dble(bn
#(2) * bs(3)) / 0.2D1)

 SIGBAR(3,1) =dble(2*PSI1*BB(3,1)) + dble(2 * PSI4f*bf(3)*bf
     (1)) +
# dble(2 * PSI4s * bs(3) * bs(1)) + 0.2D1 * PSI8 * (dble(bf
    (3) * bs
#(1)) / 0.2D1 + dble(bs(3) * bf(1)) / 0.2D1) + 0.2D1 * PSI6 *
     bn(3)
# * bn(1) + 0.2D1 * PSI6sn * (dble(bs(3) * bn(1)) / 0.2D1 +
    dble(bn
#(1) * bs(3)) / 0.2D1)

 SIGBAR(3,2) =dble(2 * PSI1*BB(3,2))+dble(2*PSI4f*bf(2)*bf(3)
     ) +
# dble(2 * PSI4s * bs(2) * bs(3)) + 0.2D1 * PSI8 * (dble(bf
    (2) * bs
#(3)) / 0.2D1 + dble(bs(2) * bf(3)) / 0.2D1) + 0.2D1 * PSI6 *
     bn(3)
# * bn(2) + 0.2D1 * PSI6sn * (dble(bs(3) * bn(2)) / 0.2D1 +
    dble(bn
#(3) * bs(2)) / 0.2D1)

 SIGBAR(3,3) =2 * PSI1*BB(3,3) + 2*PSI4f*bf(3) ** 2 +2*PSI4s
     * bs
#(3) ** 2 + 2 * PSI8 * bf(3) * bs(3) + 0.2D1 * PSI6 * bn(3) *
     bn(3)
# + 0.2D1 * PSI6sn * bs(3) * bn(3)

 STATEV(16) = SIGBAR(3,3)


 SIGISO(1,1) = 1 / DET*(0.2D1/0.3D1*SIGBAR(1,1)-SIGBAR(2,2)/
     0
#.3D1 - SIGBAR(3,3) / 0.3D1)

 STATEV(17) = SIGISO(1,1)

 SIGISO(1,2) = 1 / DET * SIGBAR(1,2)

 SIGISO(1,3) = 1 / DET * SIGBAR(1,3)

 SIGISO(2,1) = 1 / DET * SIGBAR(2,1)

 SIGISO(2,2) = 1/DET *(0.2D1/0.3D1*SIGBAR(2,2)-SIGBAR(1,1)/ 0
#.3D1 - SIGBAR(3,3) / 0.3D1)

 SIGISO(2,3) = 1 / DET * SIGBAR(2,3)
```

```
      SIGISO(3,1) = 1 / DET * SIGBAR(3,1)

      SIGISO(3,2) = 1 / DET * SIGBAR(3,2)

      SIGISO(3,3) = 1 /DET *(0.2D1/0.3D1*SIGBAR(3,3)-SIGBAR(1,1)/
          0
     #.3D1 - SIGBAR(2,2) / 0.3D1)


c       STATEV(10)=SIGISO(1,2)

        STRESS(1)=STRESS(1)+SIGISO(1,1)
        STRESS(2)=STRESS(2)+SIGISO(2,2)
        STRESS(3)=STRESS(3)+SIGISO(3,3)
        STRESS(4)=STRESS(4)+SIGISO(1,2)
        STRESS(5)=STRESS(5)+SIGISO(1,3)
        STRESS(6)=STRESS(6)+SIGISO(2,3)


        TRSIGB=SIGBAR(1,1) + SIGBAR(2,2) + SIGBAR(3,3)
CCC     definition devB devff,ss,fs

      devB(1,1)=0.2D1 / 0.3D1*BB(1,1)- BB(2,2)/0.3D1 - BB(3,3) /
          0.3D1
        devB(1,2)=BB(1,2)
        devB(1,3)=BB(1,3)
        devB(2,1)=BB(2,1)
        devB(2,2)=0.2D1 / 0.3D1*BB(2,2)- BB(1,1)/0.3D1 - BB(3,3) /
             0.3D1
        devB(2,3)=BB(2,3)
        devB(3,1)=BB(3,1)
        devB(3,2)=BB(3,2)
        devB(3,3)=0.2D1 / 0.3D1*BB(3,3)- BB(1,1)/0.3D1 - BB(2,2) /
             0.3D1
C
    ------------------------------------------------------------------------

        devff(1,1)=0.2D1/0.3D1 * bf(1)**2 - bf(2) ** 2 / 0.3D1 -
            bf(3) **
     #2 / 0.3D1

        devff(1,2)=bf(1) * bf(2)
        devff(1,3)=bf(1) * bf(3)
        devff(2,1)=bf(2) * bf(1)
        devff(2,2)=0.2D1/0.3D1 * bf(2)**2 - bf(1) ** 2 / 0.3D1 -
            bf(3) **
     #2 / 0.3D1

        devff(2,3)=bf(2) * bf(3)
        devff(3,1)=bf(3) * bf(1)
```

```
      devff(3,2)=bf(3) * bf(2)
      devff(3,3)=0.2D1/0.3D1 * bf(3)**2 - bf(1) ** 2 / 0.3D1 -
          bf(2) **
     #2 / 0.3D1

C
     ------------------------------------------------------------------

      devss(1,1)=0.2D1/0.3D1 * bs(1)**2 - bs(2) ** 2 / 0.3D1 -
          bs(3) **
     #2 / 0.3D1

      devss(1,2)=bs(1) * bs(2)
      devss(1,3)=bs(1) * bs(3)
      devss(2,1)=bs(2) * bs(1)
      devss(2,2)=0.2D1/0.3D1 * bs(2)**2 - bs(1) ** 2 / 0.3D1 -
          bs(3) **
     #2 / 0.3D1

      devss(2,3)=bs(2) * bs(3)
      devss(3,1)=bs(3) * bs(1)
      devss(3,2)=bs(3) * bs(2)
      devss(3,3)=0.2D1/0.3D1 * bs(3)**2 - bs(1) ** 2 / 0.3D1 -
          bs(2) **
     #2 / 0.3D1


C
     ------------------------------------------------------------------

      devfs(1,1)=0.2D1/0.3D1 * bf(1)*bs(1) - bf(2) * bs(2) / 0.3
          D1 - bf(
     #3) * bs(3) / 0.3D1

      devfs(1,2)=bf(1) * bs(2) / 0.2D1 + bs(1) * bf(2) / 0.2D1
      devfs(1,3)=bf(1) * bs(3) / 0.2D1 + bs(1) * bf(3) / 0.2D1
      devfs(2,1)=bf(1) * bs(2) / 0.2D1 + bs(1) * bf(2) / 0.2D1
      devfs(2,2)=0.2D1/0.3D1 * bf(2)*bs(2) - bf(1) * bs(1) / 0.3
          D1 - bf(
     #3) * bs(3) / 0.3D1

      devfs(2,3)=bf(2) * bs(3) / 0.2D1 + bs(2) * bf(3) / 0.2D1
      devfs(3,1)=bf(3) * bs(1) / 0.2D1 + bs(3) * bf(1) / 0.2D1
      devfs(3,2)=bf(2) * bs(3) / 0.2D1 + bs(2) * bf(3) / 0.2D1
      devfs(3,3)=0.2D1/0.3D1 * bf(3)*bs(3) - bf(1) * bs(1) / 0.3
          D1 - bf(
     #2) * bs(2) / 0.3D1
```

```
CCC     Tangent stiffness


        DDSDDE(1,1)=DDSDDE(1,1)+
     #dble(4 * PSI11* devB(1,1) ** 2) + dble(8 * PSI14f * devB
         (1,1)
     # * devff(1,1)) + dble(8 * PSI14s * devB(1,1) * devss(1,1)) +
          dble(
     #8 * PSI18 * devB(1,1) * devfs(1,1)) + dble(4 * PSI44f *
         devff(1,1)
     # ** 2) + dble(4 * PSI44s * devss(1,1) ** 2) + dble(4 * PSI88
          * dev
     #fs(1,1) ** 2) - 0.4D1 / 0.3D1 * SIGBAR(1,1) + 0.8D1 / 0.9D1
         * TRSI
     #GB

        DDSDDE(2,2)=DDSDDE(2,2)+
     #dble(4 * PSI11* devB(2,2) ** 2) + dble(8 * PSI14f * devB(2,2
     #) * devff(2,2)) + dble(8 * PSI14s * devB(2,2) * devss(2,2))
         + dble
     #(8 * PSI18 * devB(2,2) * devfs(2,2)) + dble(4 * PSI44f *
         devff(2,2
     #) ** 2) + dble(4 * PSI44s * devss(2,2) ** 2) + dble(4 *
         PSI88 * de
     #vfs(2,2) ** 2) - 0.4D1 / 0.3D1 * SIGBAR(2,2) + 0.8D1 / 0.9D1
          * TRS
     #IGB

        DDSDDE(3,3)=DDSDDE(3,3)+
     #dble(4 * PSI11* devB(3,3) ** 2) + dble(8 * PSI14f * devB(3,3
     #) * devff(3,3)) + dble(8 * PSI14s * devB(3,3) * devss(3,3))
         + dble
     #(8 * PSI18 * devB(3,3) * devfs(3,3)) + dble(4 * PSI44f *
         devff(3,3
     #) ** 2) + dble(4 * PSI44s * devss(3,3) ** 2) + dble(4 *
         PSI88 * de
     #vfs(3,3) ** 2) - 0.4D1 / 0.3D1 * SIGBAR(3,3) + 0.8D1 / 0.9D1
          * TRS
     #IGB

        DDSDDE(4,4)=DDSDDE(4,4)+
     #dble(4 * PSI11* devB(1,2) ** 2) + dble(8 * PSI14f * devB(1,2
     #) * devff(1,2)) + dble(8 * PSI14s * devB(1,2) * devss(1,2))
         + dble
     #(8 * PSI18 * devB(1,2) * devfs(1,2)) + dble(4 * PSI44f *
         devff(1,2
     #) ** 2) + dble(4 * PSI44s * devss(1,2) ** 2) + dble(4 *
         PSI88 * de
     #vfs(1,2) ** 2) + TRSIGB / 0.3D1
```

```
   DDSDDE(5,5)=DDSDDE(5,5)+
#dble(4 * PSI11* devB(1,3) ** 2) + dble(8 * PSI14f * devB(1,3
#) * devff(1,3)) + dble(8 * PSI14s * devB(1,3) * devss(1,3))
    + dble
#(8 * PSI18 * devB(1,3) * devfs(1,3)) + dble(4 * PSI44f *
    devff(1,3
#) ** 2) + dble(4 * PSI44s * devss(1,3) ** 2) + dble(4 *
    PSI88 * de
#vfs(1,3) ** 2) + TRSIGB / 0.3D1

   DDSDDE(6,6)=DDSDDE(6,6)+
#dble(4 * PSI11* devB(2,3) ** 2) + dble(8 * PSI14f * devB(2,3
#) * devff(2,3)) + dble(8 * PSI14s * devB(2,3) * devss(2,3))
    + dble
#(8 * PSI18 * devB(2,3) * devfs(2,3)) + dble(4 * PSI44f *
    devff(2,3
#) ** 2) + dble(4 * PSI44s * devss(2,3) ** 2) + dble(4 *
    PSI88 * de
#vfs(2,3) ** 2) + TRSIGB / 0.3D1

   DDSDDE(1,2)=DDSDDE(1,2)+
#dble(4 * PSI11* devB(1,1) * devB(2,2)) + dble(4 * PSI14f * (
#devB(1,1) * devff(2,2) + devff(1,1) * devB(2,2))) + dble(4 *
     PSI14
#s * (devB(1,1) * devss(2,2) + devss(1,1) * devB(2,2))) +
    dble(4 *
#PSI18 * (devB(1,1) * devfs(2,2) + devfs(1,1) * devB(2,2))) +
     dble(
#4 * PSI44f * devff(1,1) * devff(2,2)) + dble(4 * PSI44s *
    devss(1,
#1) * devss(2,2)) + dble(4 * PSI88 * devfs(1,1) * devfs(2,2))
     - 0.2
#D1 / 0.3D1 * SIGBAR(2,2) - 0.2D1 / 0.3D1 * SIGBAR(1,1) + 0.2
    D1 / 0
#.9D1 * TRSIGB

   DDSDDE(1,3)=DDSDDE(1,3)+
#dble(4 * PSI11* devB(1,1) * devB(3,3)) + dble(4 * PSI14f * (
#devB(1,1) * devff(3,3) + devff(1,1) * devB(3,3))) + dble(4 *
     PSI14
#s * (devB(1,1) * devss(3,3) + devss(1,1) * devB(3,3))) +
    dble(4 *
#PSI18 * (devB(1,1) * devfs(3,3) + devfs(1,1) * devB(3,3))) +
     dble(
#4 * PSI44f * devff(1,1) * devff(3,3)) + dble(4 * PSI44s *
    devss(1,
#1) * devss(3,3)) + dble(4 * PSI88 * devfs(1,1) * devfs(3,3))
     - 0.2
```

```
#D1 / 0.3D1 * SIGBAR(3,3) - 0.2D1 / 0.3D1 * SIGBAR(1,1) + 0.2
    D1 / 0
#.9D1 * TRSIGB

   DDSDDE(1,4)=DDSDDE(1,4)+
#dble(4 * PSI11* devB(1,1) * devB(1,2)) + dble(4 * PSI14f * (
#devB(1,1) * devff(1,2) + devff(1,1) * devB(1,2))) + dble(4 *
     PSI14
#s * (devB(1,1) * devss(1,2) + devss(1,1) * devB(1,2))) +
    dble(4 *
#PSI18 * (devB(1,1) * devfs(1,2) + devfs(1,1) * devB(1,2))) +
     dble(
#4 * PSI44f * devff(1,1) * devff(1,2)) + dble(4 * PSI44s *
    devss(1,
#1) * devss(1,2)) + dble(4 * PSI88 * devfs(1,1) * devfs(1,2))
     - 0.2
#D1 / 0.3D1 * SIGBAR(1,2)

   DDSDDE(1,5)=DDSDDE(1,5)+
#dble(4 * PSI11* devB(1,1) * devB(1,3)) + dble(4 * PSI14f * (
#devB(1,1) * devff(1,3) + devff(1,1) * devB(1,3))) + dble(4 *
     PSI14
#s * (devB(1,1) * devss(1,3) + devss(1,1) * devB(1,3))) +
    dble(4 *
#PSI18 * (devB(1,1) * devfs(1,3) + devfs(1,1) * devB(1,3))) +
     dble(
#4 * PSI44f * devff(1,1) * devff(1,3)) + dble(4 * PSI44s *
    devss(1,
#1) * devss(1,3)) + dble(4 * PSI88 * devfs(1,1) * devfs(1,3))
     - 0.2
#D1 / 0.3D1 * SIGBAR(1,3)

   DDSDDE(1,6)=DDSDDE(1,6)+
#dble(4 * PSI11* devB(1,1) * devB(2,3)) + dble(4 * PSI14f * (
#devB(1,1) * devff(2,3) + devff(1,1) * devB(2,3))) + dble(4 *
     PSI14
#s * (devB(1,1) * devss(2,3) + devss(1,1) * devB(2,3))) +
    dble(4 *
#PSI18 * (devB(1,1) * devfs(2,3) + devfs(1,1) * devB(2,3))) +
     dble(
#4 * PSI44f * devff(1,1) * devff(2,3)) + dble(4 * PSI44s *
    devss(1,
#1) * devss(2,3)) + dble(4 * PSI88 * devfs(1,1) * devfs(2,3))
     - 0.2
#D1 / 0.3D1 * SIGBAR(2,3)

   DDSDDE(2,3)=DDSDDE(2,3)+
#dble(4 * PSI11* devB(2,2) * devB(3,3)) + dble(4 * PSI14f *
#(devB(2,2) * devff(3,3) + devff(2,2) * devB(3,3))) + dble(4
    * PSI1
```

```
#4s * (devB(2,2) * devss(3,3) + devss(2,2) * devB(3,3))) +
    dble(4 *
# PSI18 * (devB(2,2) * devfs(3,3) + devfs(2,2) * devB(3,3)))
    + dble
#(4 * PSI44f * devff(2,2) * devff(3,3)) + dble(4 * PSI44s *
    devss(2
#,2) * devss(3,3)) + dble(4 * PSI88 * devfs(2,2) * devfs(3,3)
    ) - 0.
#2D1 / 0.3D1 * SIGBAR(3,3) - 0.2D1 / 0.3D1 * SIGBAR(2,2) +
    0.2D1 /
#0.9D1 * TRSIGB

   DDSDDE(2,4)=DDSDDE(2,4)+
#dble(4 * PSI11* devB(2,2) * devB(1,2)) + dble(4 * PSI14f *
#(devB(2,2) * devff(1,2) + devff(2,2) * devB(1,2))) + dble(4
    * PSI1
#4s * (devB(2,2) * devss(1,2) + devss(2,2) * devB(1,2))) +
    dble(4 *
# PSI18 * (devB(2,2) * devfs(1,2) + devfs(2,2) * devB(1,2)))
    + dble
#(4 * PSI44f * devff(2,2) * devff(1,2)) + dble(4 * PSI44s *
    devss(2
#,2) * devss(1,2)) + dble(4 * PSI88 * devfs(2,2) * devfs(1,2)
    ) - 0.
#2D1 / 0.3D1 * SIGBAR(1,2)

   DDSDDE(2,5)=DDSDDE(2,5)+
#dble(4 * PSI11* devB(2,2) * devB(1,3)) + dble(4 * PSI14f *
#(devB(2,2) * devff(1,3) + devff(2,2) * devB(1,3))) + dble(4
    * PSI1
#4s * (devB(2,2) * devss(1,3) + devss(2,2) * devB(1,3))) +
    dble(4 *
# PSI18 * (devB(2,2) * devfs(1,3) + devfs(2,2) * devB(1,3)))
    + dble
#(4 * PSI44f * devff(2,2) * devff(1,3)) + dble(4 * PSI44s *
    devss(2
#,2) * devss(1,3)) + dble(4 * PSI88 * devfs(2,2) * devfs(1,3)
    ) - 0.
#2D1 / 0.3D1 * SIGBAR(1,3)

   DDSDDE(2,6)=DDSDDE(2,6)+
#dble(4 * PSI11* devB(2,2) * devB(2,3)) + dble(4 * PSI14f *
#(devB(2,2) * devff(2,3) + devff(2,2) * devB(2,3))) + dble(4
    * PSI1
#4s * (devB(2,2) * devss(2,3) + devss(2,2) * devB(2,3))) +
    dble(4 *
# PSI18 * (devB(2,2) * devfs(2,3) + devfs(2,2) * devB(2,3)))
    + dble
#(4 * PSI44f * devff(2,2) * devff(2,3)) + dble(4 * PSI44s *
    devss(2
```

```
#,2) * devss(2,3)) + dble(4 * PSI88 * devfs(2,2) * devfs(2,3)
    ) - 0.
#2D1 / 0.3D1 * SIGBAR(2,3)


   DDSDDE(3,4)=DDSDDE(3,4)+
#dble(4 * PSI11* devB(3,3) * devB(1,2)) + dble(4 * PSI14f *
#(devB(3,3) * devff(1,2) + devff(3,3) * devB(1,2))) + dble(4
    * PSI1
#4s * (devB(3,3) * devss(1,2) + devss(3,3) * devB(1,2))) +
    dble(4 *
# PSI18 * (devB(3,3) * devfs(1,2) + devfs(3,3) * devB(1,2)))
    + dble
#(4 * PSI44f * devff(3,3) * devff(1,2)) + dble(4 * PSI44s *
    devss(3
#,3) * devss(1,2)) + dble(4 * PSI88 * devfs(3,3) * devfs(1,2)
    ) - 0.
#2D1 / 0.3D1 * SIGBAR(1,2)


   DDSDDE(3,5)=DDSDDE(3,5)+
#dble(4 * PSI11* devB(3,3) * devB(1,3)) + dble(4 * PSI14f *
#(devB(3,3) * devff(1,3) + devff(3,3) * devB(1,3))) + dble(4
    * PSI1
#4s * (devB(3,3) * devss(1,3) + devss(3,3) * devB(1,3))) +
    dble(4 *
# PSI18 * (devB(3,3) * devfs(1,3) + devfs(3,3) * devB(1,3)))
    + dble
#(4 * PSI44f * devff(3,3) * devff(1,3)) + dble(4 * PSI44s *
    devss(3
#,3) * devss(1,3)) + dble(4 * PSI88 * devfs(3,3) * devfs(1,3)
    ) - 0.
#2D1 / 0.3D1 * SIGBAR(1,3)


   DDSDDE(3,6)=DDSDDE(3,6)+
#dble(4 * PSI11* devB(3,3) * devB(2,3)) + dble(4 * PSI14f *
#(devB(3,3) * devff(2,3) + devff(3,3) * devB(2,3))) + dble(4
    * PSI1
#4s * (devB(3,3) * devss(2,3) + devss(3,3) * devB(2,3))) +
    dble(4 *
# PSI18 * (devB(3,3) * devfs(2,3) + devfs(3,3) * devB(2,3)))
    + dble
#(4 * PSI44f * devff(3,3) * devff(2,3)) + dble(4 * PSI44s *
    devss(3
#,3) * devss(2,3)) + dble(4 * PSI88 * devfs(3,3) * devfs(2,3)
    ) - 0.
#2D1 / 0.3D1 * SIGBAR(2,3)
```

```
      DDSDDE(4,5)=DDSDDE(4,5)+
     #4 * PSI11* devB(1,2) * devB(1,3) + 4 * PSI14f * (devB(1,2)
     #* devff(1,3) + devff(1,2) * devB(1,3)) + 4 * PSI14s * (devB
        (1,2) *
     # devss(1,3) + devss(1,2) * devB(1,3)) + 4 * PSI18 * (devB
        (1,2) * d
     #evfs(1,3) + devfs(1,2) * devB(1,3)) + 4 * PSI44f * devff
        (1,2) * de
     #vff(1,3) + 4 * PSI44s * devss(1,2) * devss(1,3) + 4 * PSI88
        * devf
     #s(1,2) * devfs(1,3)


      DDSDDE(4,6)=DDSDDE(4,6)+
     #4 * PSI11* devB(1,2) * devB(2,3) + 4 * PSI14f * (devB(1,2)
     #* devff(2,3) + devff(1,2) * devB(2,3)) + 4 * PSI14s * (devB
        (1,2) *
     # devss(2,3) + devss(1,2) * devB(2,3)) + 4 * PSI18 * (devB
        (1,2) * d
     #evfs(2,3) + devfs(1,2) * devB(2,3)) + 4 * PSI44f * devff
        (1,2) * de
     #vff(2,3) + 4 * PSI44s * devss(1,2) * devss(2,3) + 4 * PSI88
        * devf
     #s(1,2) * devfs(2,3)


      DDSDDE(5,6)=DDSDDE(5,6)+
     #4 * PSI11* devB(1,3) * devB(2,3) + 4 * PSI14f * (devB(1,3)
     #* devff(2,3) + devff(1,3) * devB(2,3)) + 4 * PSI14s * (devB
        (1,3) *
     # devss(2,3) + devss(1,3) * devB(2,3)) + 4 * PSI18 * (devB
        (1,3) * d
     #evfs(2,3) + devfs(1,3) * devB(2,3)) + 4 * PSI44f * devff
        (1,3) * de
     #vff(2,3) + 4 * PSI44s * devss(1,3) * devss(2,3) + 4 * PSI88
        * devf
     #s(1,3) * devfs(2,3)



CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        DO K1=1, NTENS
        DO K2=1, K1-1
          DDSDDE(K1, K2)=DDSDDE(K2, K1)
        END DO
      END DO
```

```
      DDSDDE(1,1) = DDSDDE(1,1) + 2 * SIGISO(1,1)
      DDSDDE(1,2) = DDSDDE(1,2)
      DDSDDE(1,3) = DDSDDE(1,3)
      DDSDDE(1,4) = DDSDDE(1,4) + SIGISO(1,2)
      DDSDDE(1,5) = DDSDDE(1,5) + SIGISO(1,3)
      DDSDDE(1,6) = DDSDDE(1,6)
      DDSDDE(2,1) = DDSDDE(2,1)
      DDSDDE(2,2) = DDSDDE(2,2) + 2 * SIGISO(2,2)
      DDSDDE(2,3) = DDSDDE(2,3)
      DDSDDE(2,4) = DDSDDE(2,4) + SIGISO(1,2)
      DDSDDE(2,5) = DDSDDE(2,5)
      DDSDDE(2,6) = DDSDDE(2,6) + SIGISO(2,3)
      DDSDDE(3,1) = DDSDDE(3,1)
      DDSDDE(3,2) = DDSDDE(3,2)
      DDSDDE(3,3) = DDSDDE(3,3) + 2*SIGISO(3,3)
      DDSDDE(3,4) = DDSDDE(3,4)
      DDSDDE(3,5) = DDSDDE(3,5) + SIGISO(1,3)
      DDSDDE(3,6) = DDSDDE(3,6) + SIGISO(2,3)
      DDSDDE(4,1) = DDSDDE(4,1) + SIGISO(1,2)
      DDSDDE(4,2) = DDSDDE(4,2) + SIGISO(1,2)
      DDSDDE(4,3) = DDSDDE(4,3)
      DDSDDE(4,4) = dble(DDSDDE(4,4)) + dble(SIGISO(1,1)) / 0.2D1
     #+ dble(SIGISO(2,2)) / 0.2D1
      DDSDDE(4,5) = dble(DDSDDE(4,5)) + dble(SIGISO(2,3)) / 0.2D1
      DDSDDE(4,6) = dble(DDSDDE(4,6)) + dble(SIGISO(1,3)) / 0.2D1
      DDSDDE(5,1) = DDSDDE(5,1) + SIGISO(1,3)
      DDSDDE(5,2) = DDSDDE(5,2)
      DDSDDE(5,3) = DDSDDE(5,3) + SIGISO(1,3)
      DDSDDE(5,4) = dble(DDSDDE(5,4)) + dble(SIGISO(2,3)) / 0.2D1
      DDSDDE(5,5) = dble(DDSDDE(5,5)) + dble(SIGISO(1,1)) / 0.2D1
     #+ dble(SIGISO(3,3)) / 0.2D1
      DDSDDE(5,6) = dble(DDSDDE(5,6)) + dble(SIGISO(1,2)) / 0.2D1
      DDSDDE(6,1) = DDSDDE(6,1)
      DDSDDE(6,2) = DDSDDE(6,2) + SIGISO(2,3)
      DDSDDE(6,3) = DDSDDE(6,3) + SIGISO(2,3)
      DDSDDE(6,4) = dble(DDSDDE(6,4)) + dble(SIGISO(1,3)) / 0.2D1
      DDSDDE(6,5) = dble(DDSDDE(6,5)) + dble(SIGISO(1,2)) / 0.2D1
      DDSDDE(6,6) = dble(DDSDDE(6,6)) + dble(SIGISO(2,2)) / 0.2D1
     #+ dble(SIGISO(3,3)) / 0.2D1

C       END IF
      RETURN
      END
```

D Calculate Ejection Fraction

D.1 Python Script

```
#This Python script is used as a part of calculating the end-
    diastole volume, end-systole volume and ejection fraction.
#The script extracts coordinates and displacement at the end of
    the analysis for every node and writes them to a txt-file.
#The txt-file is then processed further using a MATLAB script.

from odbAccess import *
from abaqusConstants import *

# create odb object from odb file
outputDatabase = openOdb(path='C:\Users\Gaute\Documents\
    MASTERTHESIS\Abaqus\MainModel\TEM_8layers_fnsn150_f45_14.odb')

# get access to the nodal displacement data
frame = outputDatabase.steps[ 'static' ].frames[-1]

dispField = frame.fieldOutputs['U']

# get access to the part instance
my_part_instance = outputDatabase.rootAssembly.instances['TEM-8
    LAYERS-1']


# Write deformed shape to txt file

outFile = open("deformed_shape_TEM_8layers_fnsn150_f45.txt" , 'w'
    )

# write points

numNodesTotal = len( my_part_instance.nodes )

for i in range( numNodesTotal ):

    curNode =  my_part_instance.nodes[i]

    defNodePos = curNode.coordinates + dispField.values[i].data

    outFile.write( '\n' )

    for j in range( 3 ):
                if j<2:
                        outFile.write( str( defNodePos[j] ) + ' '
                            )
                else:
```

```
                        outFile.write( str( defNodePos[j] ) )

outputDatabase.close()
```

D.2 MATLAB Script

```
%This MATLAB-script is used to calculate the end-diastole volume,
%end-systole volume and ejection fraction for the left ventricle
    model.
%The scrpit uses an Abaqus inputfile which must contain a node set
     with all
%nodes on endocaridal surface.
%Further at txt file (created with the python script) containing
    the node
%coordinates at the end-systole state.

clear all
close all
clc

%Extract relevant line numbers from input file. Change keywords to
     coincide
%with the currect file.
inputFile = fopen('TEM_8layers_fnsn150_f45.inp','rt');
counter = 0;
while 1
    tline = fgetl(inputFile);
    counter = counter + 1;
    if ischar(tline)
        U = strfind(tline, '*Node');
        if isfinite(U) == 1;
            nodeLine = counter;
        end
        U = strfind(tline, '*Element');
        if isfinite(U) == 1;
            elsLine = counter;
        end

        U = strfind(tline, 'nset=endoNodes');
        if isfinite(U) == 1;
            endoNodeline = counter;
        end
        U = strfind(tline, 'nset=basalNodes');
        if isfinite(U) == 1;
            endoNodestop = counter;
            break
        end
    end
end
```

```
fclose(inputFile);
%Number of nodes
nnodes=elsLine-nodeLine-1;
%Extract node coordinates at end-diastole state.
orgNode = csvread('TEM_8layers_fnsn150_f45.inp',nodeLine, 1, [
    nodeLine,1,nodeLine+nnodes-1,3]);

%Rotate nodes 90 degrees about x-axis as done with the instance in
     the
%Abaqus model
tempY = orgNode(:,2);
tempZ = orgNode(:,3);

orgNode(:,2) = cos(pi()/2)*tempY(:)-sin(pi()/2)*tempZ(:);
orgNode(:,3) = sin(pi()/2)*tempY(:)+cos(pi()/2)*tempZ(:);

%Extract node numbers from the endoNode set
endoNodeSet = csvread('TEM_8layers_fnsn150_f45.inp',endoNodeline,
    0, [endoNodeline,0,endoNodestop-2,15]);
endoNodeSet(endoNodeSet==0) = [];
endoNodeSet = sort(endoNodeSet(:));

%Extract node coordinates at end-systle state.
deformedNode =textread('deformed_shape_TEM_8layers_fnsn150_f45.txt
    ','','headerlines', 3);

%Creating variables containing node coordinates only from the
    nodes on the
%endocardial surface
orgEndoCoor = zeros(numel(endoNodeSet),3);
deformedEndoCoor = zeros(numel(endoNodeSet),3);

for i=1:numel(endoNodeSet)
    orgEndoCoor(i,:) = orgNode(endoNodeSet(i),:);
    deformedEndoCoor(i,:) = deformedNode(endoNodeSet(i),:);
end

%Caclutate the end-diastole and end-systole volume using the
    convhull
%function
[TriIdx, V] = convhull(orgEndoCoor(:,1),orgEndoCoor(:,2),
    orgEndoCoor(:,3));

[TriIdx2, V2] = convhull(deformedEndoCoor(:,1),deformedEndoCoor
    (:,2),deformedEndoCoor(:,3));

%Write out the volumes (in ml) and ejection fraction
V*0.001
V2*0.001
ejectionFraction=(V-V2)/V
```

```
%The remainder of the script is plots used for visual confirmation
    of the
%validity of the result

figure()
hold on
scatter3(orgEndoCoor(:,1), orgEndoCoor(:,2), orgEndoCoor(:,3), '.'
    )
scatter3(deformedEndoCoor(:,1), deformedEndoCoor(:,2),
    deformedEndoCoor(:,3), 'r.')
xlabel('X')
ylabel('Y')
zlabel('Z')
hold off

figure()
hold on
scatter3(deformedEndoCoor(:,1), deformedEndoCoor(:,2),
    deformedEndoCoor(:,3), '.')
xlabel('X')
ylabel('Y')
zlabel('Z')
hold off

figure()
hold on
trisurf(TriIdx, orgEndoCoor(:,1),orgEndoCoor(:,2),orgEndoCoor(:,3)
    )
xlabel('X')
ylabel('Y')
zlabel('Z')
hold off

figure()
trisurf(TriIdx2, deformedEndoCoor(:,1),deformedEndoCoor(:,2),
    deformedEndoCoor(:,3))
xlabel('X')
ylabel('Y')
zlabel('Z')
```

E Calculate Wall Thickness, Longitudinal and Radial Shortening

```
#This python script extracts nodal displacement and calculates the
     longitudinal shortening, radial shortening and wall
    thickening
#Variables which needs to be changed are: odb file-path, instance
    name and node numbers.

from odbAccess import *
from abaqusConstants import *

# create odb object from odb file
outputDatabase = openOdb(path='C:\Users\Gaute\Documents\
    MASTERTHESIS\Abaqus\MainModel\TEM_4layers_f100_f60_14.odb')

# get access to the nodal displacement data
frame = outputDatabase.steps[ 'static' ].frames[-1]

dispField = frame.fieldOutputs['U']

# get access to the part instance
my_part_instance = outputDatabase.rootAssembly.instances['TEM-4
    LAYERS3-1']

#Variables with node numbers on the form [endocardium,epicardium]
equNodes = [36,27]
ApexNodes = [11, 21]

#Extracting the coordinates before and after deformation
#Note that the note label is not the same as the node number.

numNodesTotal = len( my_part_instance.nodes )
for i in range( numNodesTotal ):
        nodeLabel = my_part_instance.nodes[i].label
        if nodeLabel==equNodes[0]:
                node1int = i
                node1Pos = my_part_instance.nodes[i].coordinates
                defNode1Pos = node1Pos + dispField.values[i].data
        if nodeLabel==equNodes[1]:
                node2int = i
                node2Pos = my_part_instance.nodes[i].coordinates
                defNode2Pos = node2Pos + dispField.values[i].data
        if nodeLabel==ApexNodes[0]:
                apexint = i
                apexEndoPos = my_part_instance.nodes[i].
                    coordinates
                defEndoApexPos = apexEndoPos + dispField.values[i
                    ].data
```

```
        if nodeLabel==ApexNodes[1]:
                apexEpiint = i
                apexEpiPos = my_part_instance.nodes[i].coordinates
                defEpiApexPos = apexEpiPos + dispField.values[i].
                    data

#Calculate the wanted parameters
initalWallTh = sqrt((node2Pos[0]-node1Pos[0])**2 + (node2Pos[1]-
    node1Pos[1])**2 + (node2Pos[2]-node1Pos[2])**2)
defWallTh = sqrt((defNode2Pos[0]-defNode1Pos[0])**2 + (defNode2Pos
    [1]-defNode1Pos[1])**2)
thFrac = ( defWallTh - initalWallTh) / initalWallTh

initalApexWallTh = sqrt((apexEndoPos[0]-apexEpiPos[0])**2 + (
    apexEndoPos[1]-apexEpiPos[1])**2 + (apexEndoPos[2]-apexEpiPos
    [2])**2)
defApexWallTh = sqrt((defEndoApexPos[0]-defEpiApexPos[0])**2 + (
    defEndoApexPos[1]-defEpiApexPos[1])**2 + (defEndoApexPos[2]-
    defEpiApexPos[2])**2)
thApexFrac = ( defApexWallTh - initalApexWallTh) /
    initalApexWallTh

initalLong = 20 - apexEndoPos[2]
defLong = 20 - defEndoApexPos[2]
relLongShort = (initalLong - defLong) / initalLong

initalRad = node1Pos[0]
defRad = defNode1Pos[0]
radShort = (initalRad-defRad)/initalRad

#Printing the parameters to the command window
print "equ wallth:"
print initalWallTh
print defWallTh
print thFrac
print "apex wallth:"
print initalApexWallTh
print defApexWallTh
print thApexFrac
print "rel Long short:"
print initalLong
print defLong
print relLongShort
print "Radial shortening"
print initalRad
print defRad
print radShort
```

F Calculate Torsion

F.1 Python Script

```
#This Python script extracts displacement history of a set of
    chosen nodes as part of calculating the rotation of the left
    ventricle model.
#The script writes the node coordinates for each frame of the
    analysis to a file. This must be done for the 4 nodes on the
    endocardium and epicardium of both the base and apex.
#The four files created using this script is processed further
    using a MATLAB script.

from odbAccess import *
from abaqusConstants import *
from math import *

# create odb object from odb file
outputDatabase = openOdb(path='C:\Users\Gaute\Documents\
    MASTERTHESIS\Abaqus\MainModel\TEM_4layers_fnsn150_f45_14.odb')

# get access to the nodal displacement data
allFrames = outputDatabase.steps[ 'static' ].frames
lastFrame = outputDatabase.steps[ 'static' ].frames[-1]

numFrames = len (allFrames)

dispField = lastFrame.fieldOutputs['U']

# get access to the part instance
my_part_instance = outputDatabase.rootAssembly.instances['TEM-4
    LAYER3-1']

#Declare variables with the node numbers from the different
    regions on the form [center, left, up, right down ] when
    looking from the apex to the base.
#The center basal node number is set to 0, and given the global
    coordinate (0,0,20) which is the center of the base in the
    model.

apexEpiNodes = [21, 419, 376, 496, 615  ]
apexEndoNodes = [11, 553, 144, 187, 345  ]
basalEndoNodes = [0, 40, 51, 50, 41  ]
basalEpiNodes =  [0, 49, 47, 44, 48 ]

#Choose node data to write to file
activeNodes = apexEpiNodes8layers

#Write data to files (change file name to coincide with nodes)
```

```
outFile = open("rotation_TEM_8layers_fnsn150_f45_apexEpi.txt" , 'w
    ' )

# write points
numNodesTotal = len( my_part_instance.nodes )
for i in range( numNodesTotal ):
        nodeLabel = my_part_instance.nodes[i].label
        if nodeLabel==activeNodes[0]:
                nodeCenterint = i
                nodeCenterPos = my_part_instance.nodes[i].
                    coordinates
        if nodeLabel==activeNodes[1]:
                nodeLeftint = i
                nodeLeftPos = my_part_instance.nodes[i].
                    coordinates
        if nodeLabel==activeNodes[2]:
                nodeUpint = i
                nodeUpPos = my_part_instance.nodes[i].coordinates
        if nodeLabel==activeNodes[3]:
                nodeRightint = i
                nodeRightPos = my_part_instance.nodes[i].
                    coordinates
        if nodeLabel==activeNodes[4]:
                nodeDownint = i
                nodeDownPos = my_part_instance.nodes[i].
                    coordinates

for k in range (numFrames):
        if activeNodes[0]==0:
                defNodeCenterPos = [0,0,20]
        else:
                nodeCenterDisp = outputDatabase.steps[ 'static' ].
                    frames[k].fieldOutputs['U'].values[
                    nodeCenterint].data
                defNodeCenterPos = nodeCenterPos + nodeCenterDisp

        nodeLeftDisp = outputDatabase.steps[ 'static' ].frames[k].
            fieldOutputs['U'].values[nodeLeftint].data
        defNodeLeftPos = nodeLeftPos + nodeLeftDisp

        nodeUpDisp = outputDatabase.steps[ 'static' ].frames[k].
            fieldOutputs['U'].values[nodeUpint].data
        defNodeUpPos = nodeUpPos + nodeUpDisp

        nodeRightDisp = outputDatabase.steps[ 'static' ].frames[k
            ].fieldOutputs['U'].values[nodeRightint].data
        defNodeRightPos = nodeRightPos + nodeRightDisp

        nodeDownDisp = outputDatabase.steps[ 'static' ].frames[k].
            fieldOutputs['U'].values[nodeDownint].data
```

```
        defNodeDownPos = nodeDownPos + nodeDownDisp


        for j in range( 3 ):
                outFile.write( str( defNodeCenterPos[j] ) + ' ' )
        for j in range( 3 ):
                outFile.write( str( defNodeLeftPos[j] ) + ' ' )
        for j in range( 3 ):
                outFile.write( str( defNodeUpPos[j] ) + ' ' )
        for j in range( 3 ):
                outFile.write( str( defNodeRightPos[j] ) + ' ' )
        for j in range( 3 ):
                if j<2:
                        outFile.write( str( defNodeDownPos[j] ) +
                            ' ' )
                else:
                        outFile.write( str( defNodeDownPos[j] ) )

        outFile.write("\n")

outputDatabase.close()
```

F.2 MATLAB Script

```
%This MATLAB script calculates the rotation of endocardium and
    epicardium
%for both the base and apex for all frames through the Abaqus
    analysis. It
%takes as input 4 txt files with nodal coordinate data, created
    with the
%Python script.

%NOTE: the function atan2 change sign at 180 degrees which is
    relevant for
%some of the nodes. For simplicity, when this is relevant, the
    node in
%question is not included in the calculation of the average
    rotation. As
%the rotation of the different nodes are almost equal, this has
    little or
%no effect on the results.

clear all
close all
clc

%Reading node data from files

nodeCoorApexEpi =textread('
    rotation_TEM_4layers_fnsn150_f45_apexEpi.txt','','headerlines'
```

```
    , 0);
nodeCoorApexEndo =textread('
    rotation_TEM_4layers_fnsn150_f45_apexEndo.txt','','headerlines
    ', 0);
nodeCoorBasalEpi =textread('
    rotation_TEM_4layers_fnsn150_f45_basalEpi.txt','','headerlines
    ', 0);
nodeCoorBasalEndo =textread('
    rotation_TEM_4layers_fnsn150_f45_basalEndo.txt','','
    headerlines', 0);


%Initalize variable

angleV=zeros(4); %Angles of the vectors the 4 undeformed nodes
    creates in the global csys
defv=zeros(4,2); %X and Y components of the deformed vectors
angleDefV=zeros(4,length(nodeCoorApexEpi)); %Angles of the vectors
     the 4 deformed nodes creates in the global csys, for every
    frame
rot=zeros(4,length(nodeCoorApexEpi)); %Rotation of the 4 nodes for
     every frame

%Caclulate rotation of apex at epicardium

for i=1:4
    angleV(i) = atan2(nodeCoorApexEpi(1,2+3*i),nodeCoorApexEpi
        (1,1+3*i))*180/pi();
    if angleV(i)<-170
        angleV(i) = 180;
    end

    for j=1:length(nodeCoorApexEpi)
    defv(i,1) = nodeCoorApexEpi(1,1)-nodeCoorApexEpi(j,1+3*i);
    defv(i,2) = nodeCoorApexEpi(1,2)- nodeCoorApexEpi(j,2+3*i);
    angleDefV(i,j) = atan2(nodeCoorApexEpi(j,2+3*i),
        nodeCoorApexEpi(j,1+3*i))*180/pi();
    if (angleDefV(i,j)) < -170
        angleDefV = angleDefV - 360*sign(angleDefV);
    end
    rot(i,j) = (angleDefV(i,j)-angleV(i));
    end
end

%Average rotation of apex at epicaridum

avgRotApexEpi = -(rot(1,:)+rot(2,:)+rot(3,:)+rot(4,:))/4;


%Caclulate rotation of apex at endocardium
```

```
for i=1:4
    angleV(i) = atan2(nodeCoorApexEndo(1,2+3*i),nodeCoorApexEndo
        (1,1+3*i))*180/pi();
    if angleV(i)<-170
        angleV(i) = 180;
    end

    for j=1:length(nodeCoorApexEndo)
    defv(i,1) = nodeCoorApexEndo(1,1)-nodeCoorApexEndo(j,1+3*i);
    defv(i,2) = nodeCoorApexEndo(1,2)- nodeCoorApexEndo(j,2+3*i);
    angleDefV(i,j) = atan2(nodeCoorApexEndo(j,2+3*i),
        nodeCoorApexEndo(j,1+3*i))*180/pi();
        if (angleDefV(i,j)) < -170
        angleDefV = angleDefV - 360*sign(angleDefV);
        end
    rot(i,j) = (angleDefV(i,j)-angleV(i));
    end
end

%Average rotation of apex at endocaridum

avgRotApexEndo = -(rot(1,:)+rot(2,:)+rot(4,:))/4;


%Caclulate rotation of base at epicardium

for i=1:4
    angleV(i) = atan2(nodeCoorBasalEpi(1,2+3*i),nodeCoorBasalEpi
        (1,1+3*i))*180/pi();
    if angleV(i)<-170
        angleV(i) = 180;
    end

    for j=1:length(nodeCoorBasalEpi)
    defv(i,1) = nodeCoorBasalEpi(1,1)-nodeCoorBasalEpi(j,1+3*i);
    defv(i,2) = nodeCoorBasalEpi(1,2)- nodeCoorBasalEpi(j,2+3*i);
    angleDefV(i,j) = atan2(nodeCoorBasalEpi(j,2+3*i),
        nodeCoorBasalEpi(j,1+3*i))*180/pi();
    if (angleDefV(i,j)) < -170
        angleDefV = angleDefV - 360*sign(angleDefV);
    end
    rot(i,j) = (angleDefV(i,j)-angleV(i));
    end
end

%Average rotation of base at epicaridum

avgRotBasalEpi = -(rot(1,:)+rot(2,:)+rot(3,:)+rot(4,:))/4;
```

```
%Caclulate rotation of base at endocardium

for i=1:4
    angleV(i) = atan2(nodeCoorBasalEndo(1,2+3*i),nodeCoorBasalEndo
        (1,1+3*i))*180/pi();
    if angleV(i)<-170
        angleV(i) = 180;
    end

    for j=1:length(nodeCoorBasalEndo)
    defv(i,1) = nodeCoorBasalEndo(1,1)-nodeCoorBasalEndo(j,1+3*i);
    defv(i,2) = nodeCoorBasalEndo(1,2)- nodeCoorBasalEndo(j,2+3*i)
        ;
    angleDefV(i,j) = atan2(nodeCoorBasalEndo(j,2+3*i),
        nodeCoorBasalEndo(j,1+3*i))*180/pi();
    if (angleDefV(i,j)) < -170
        angleDefV = angleDefV - 360*sign(angleDefV);
    end
    rot(i,j) = (angleDefV(i,j)-angleV(i));
    end
end

%Average rotation of base at endocaridum

avgRotBasalEndo = -(rot(1,:)+rot(2,:)+rot(4,:))/3;

%Write out rotation at end-systole

avgRotApexEpi(length(nodeCoorApexEpi))
avgRotBasalEpi(length(nodeCoorBasalEpi))
avgRotApexEndo(length(nodeCoorApexEpi))
avgRotBasalEndo(length(nodeCoorBasalEpi))


%Plot rotatation of all frames for visual confirmation of
    calculation

t=linspace(0,0.2,length(nodeCoorApexEpi));

figure()
plot(t, avgRotApexEpi(:), 'b')
hold on
plot(t, avgRotApexEndo(:), 'r')
plot(t, avgRotBasalEpi(:), 'm')
plot(t, avgRotBasalEndo(:), 'g')

xlabel('Analysis time [s]')
ylabel('Rotation [^{\circ}]')
legend('Apex Epi', 'Apex Endo', 'Basal Epi', 'Basal Endo ')
hold off
```