NTNU
Norwegian University of
Science and Technology

Computational modelling of cardiac mechanics

Efficient simulation of a heartbeat

Elisabeth Topnes

Applied and Engineering Mathematics
Submission date: August 2016
Supervisor: Anton Evgrafov, MATH

Norwegian University of Science and Technology
Department of Mathematical Sciences

Abstract

In this thesis we will study the mathematical models for the mechanical behaviour of the heart. We will derive the non-linear partial differential equations involving the deformation of the cardiac tissue. Models for the different phases of the cardiac cycle will be studied and developed. The cardiac cycle will be simulated both for an incompressible and a compressible material model. A review of the underlying physiology and the theory of continuum mechanics is also conducted.

The aim of this thesis is to investigate how to speed up the computations for the complex non-linear mechanics of the heartbeat. This will be studied by testing different solvers and different parameters for the form compiler within the finite element framework FEniCS. The model will be run in parallel in order to see the benefit of running the code on multiple cores. The solver will also be verified by different verification methods.

Preface

This master thesis marks the ending of my enrolment in the Nordic Five Tech Master's programme in Applied and Engineering mathematics. The main supervisors of this thesis is Anton Evgrafov at the department of mathematical science at NTNU and dr. Joakim Sundnes at Simula research laboratory. The co-supervisor is Allan Peter Ensig-Karup at the department of applied mathematics and computer science at the technical university of Denmark.

I would first of all give an enormous thank to my husband, Henrik. This master thesis could not have been done without your continuous support on all levels. You are an excellent teacher and my best friend. I admire you greatly for your patience and guidance throughout this time.

A great thanks goes also to my supervisors Anton, Joakim and Allan for excellent supervision. I would also give a sincere thank to Line for proofreading my thesis.

Lastly, I would thank my dear daughter, Ingerid, for always reminding me what is truly important in my life.

Elisabeth Topnes Finsberg
Oslo, August 2016

Contents

Chapter 1

Introduction

Heart disease is the leading cause of death in almost all areas of the world. The ability to mathematically model the heart may lead to deeper insight and understanding of the diseases affecting it.

There are many mechanisms interacting on different levels in order to make the heart beat regularly. A multiscale mathematical model is therefore needed in order to describe these multiscale mechanisms. Multiscale models describe processes on very different time and length scales and these complete models are very computationally expensive. For clinical use the time efficiency of such solvers is crucial. We wish to investigate the possibility for creating a time efficient solver for the complex non-linear mechanics of the heartbeat.

In this master thesis, the main goal will be to implement a time efficient and realistic simulator of the mechanical events of the left ventricle during a heartbeat. The mechanical events can be summarized in a pressure and volume loop (PV-loop). The PV-loop is a two-dimensional plot describing the development of the pressure and volume throughout a heartbeat.

The first aim of this thesis is to describe the basic theory of cardiac mechanics. In Chapter 2 we therefore give an introduction to the anatomy and physiology of the mechanics of the heart. We focus on the microscopical mechanical contraction of the cells, and expand to the macroscopic level. On the macroscopic level we explain the different phases of a heartbeat that we will model. In Chapter 3 we provide a brief introduction to the field of continuum mechanics, which is needed to describe the cardiac mechanics. In Chapter 4, we derive the mathematical model for the mechanical behaviour of the heart. The implementation of our mechanical model for the contraction of the left ventricle will be described in Chapter 5. In Chapter 6 we present the results from our model. We will in

this chapter verify our model, as well as investigate the pressure and volume development throughout the heartbeat. Lastly, we will optimize the time efficiency of our model. This will be done both for an incompressible and compressible material model.

Chapter 2

Anatomy and physiology: From the contracting cells to the beating heart

In this chapter we give a brief overview of the basic anatomy and physiology of the heart, from the cell level to the macroscopic contraction of the heart. The main theory presented in Chapter 2 is taken from [1].

2.1 Heart cell physiology

The cell contraction - cross bridge cycles

The *cardiomyocytes* are the contractile muscle cells of the heart, and are often referred to as muscle fibers. A top-down view of the muscle structure is given in Figure 2.1. Within the cardiomyocytes, we find long protein chains called the *myofibrils*. The myofibrils consist of, among other proteins, the contractile proteins *myosin* and *actin* and the regulatory proteins *tropomysion* and *troponin*. The myofibrils are in turn built up by subunits called *sarcomeres*. The sarcomeres are divided into thick and thin *filaments* which slide along each other to create muscle contraction. The thick filaments are built up by myosin molecules and the thin filaments are built up by actin molecules. The myosin molecules are motor proteins and consist of protein chains that form into a tail and head. The thick filaments bind to the thin filaments through these myosin heads. These connections are called *crossbridges*, and are illustrated in Figure 2.2. The troponin protein controls the positioning of tropomyosin on the thin filament. When the muscle is at rest, the tropomyosin blocks the binding sites for the myosin heads. When troponin-c

binds to calcium, the tropomyosin is removed from the actin-myosin binding sites. This enables the myosin heads to make strong, high-force bindings to the thin filaments. When the muscle contracts, the myosin heads perform power strokes that make the thin and thick filaments slide along each other. After the power stroke, the myosin heads release the actin, bind to another actin molecule and perform a new power stroke. This process is referred to as the crossbridge cycle. The crossbridge cycle continues until the calcium concentration in the cell decreases. At this point the calcium unbinds from troponin, and tropomyosin again block the actin-myosin binding sites and the sarcomers fall back to their relaxed position.

The mechanical contraction of the cardiac muscle cells is initiated by the electrical impulses from neighbouring cells. The increase in the intracellular calcium concentration causes the contraction of the muscle cells. This increase is initiated by the action potentials in the neighbouring cells. This calcium dependent contraction will be described later in this master thesis by a very simplified phenomenological model.

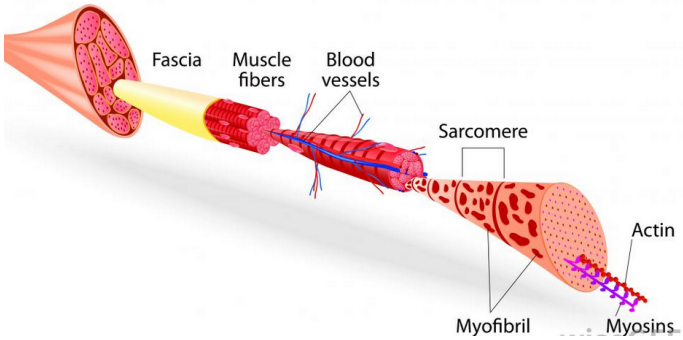


FIGURE 2.1: A top-down view of muscle http://www.wisegeekhealth.com/what-is-the-difference-between-cardiac-and-skeletal-muscles.htm.

2.2 Heart anatomy and physiology

Anatomy of the heart

The heart is a muscular organ and lies in the thoracic cavity between the lungs. It functions as the pump in the cardiovascular system, and pumps blood through the blood vessels into the organs and tissue of the body. The heart is divided into two distinct parts - a left and a right part - that are separated by the *interventricular septum*. The septum keeps the blood in the left and right part from mixing. The two sides of the heart consist in turn of two chambers, the *atrium* and *ventricle*, which again is separated from each other with the *atrioventricular valves*. These valves together with the *semilunar valves* between the ventricles and the arteries, keep the blood from flowing back into the

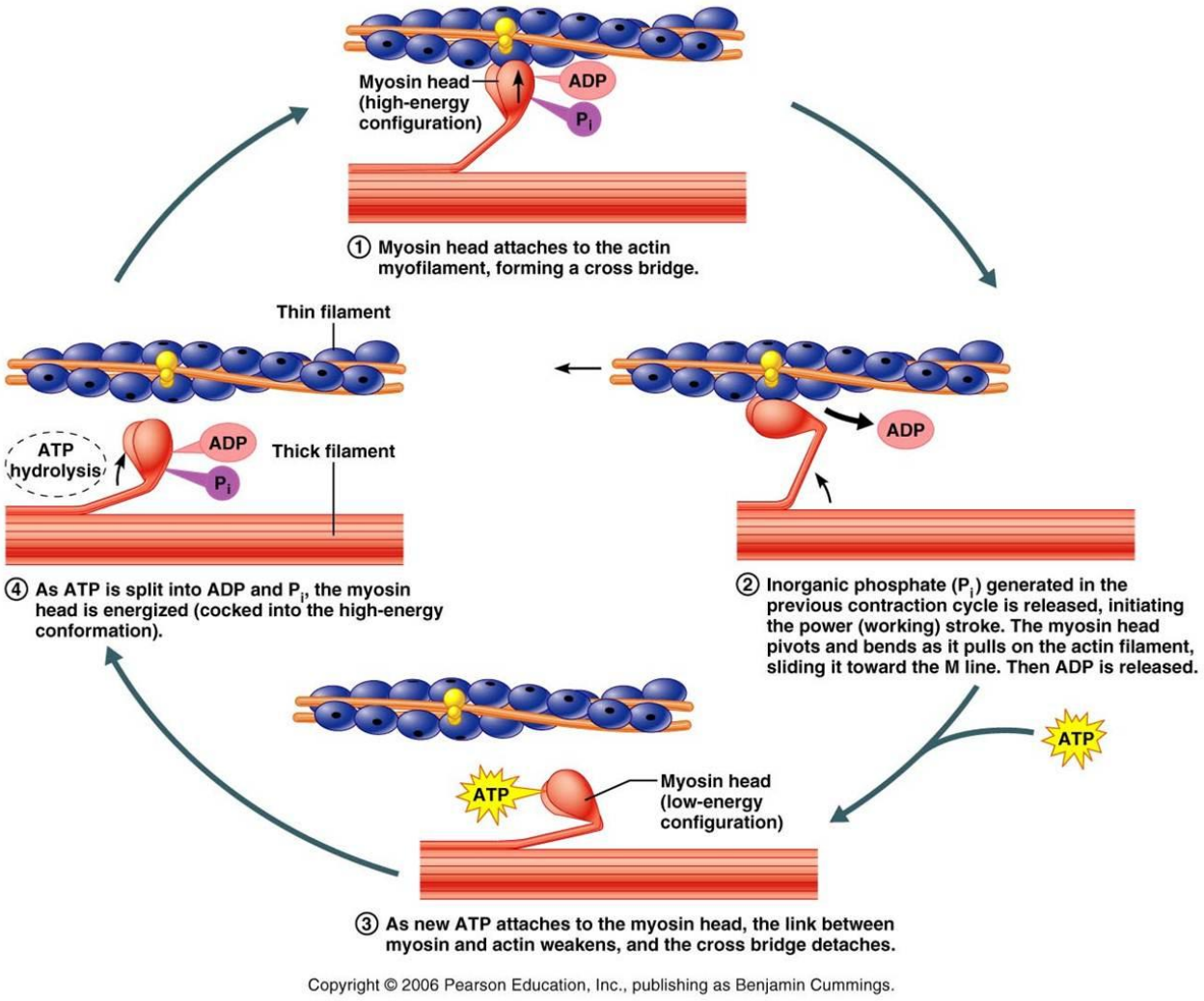


FIGURE 2.2: Illustration of the cross bridge cycle in the cardiac myocytes http://thegaitguys.tumblr.com/post/117514727354/more-on-weak-muscles-just-why-are-they-weak-know

heart and force the blood to flow in one direction. The two sides of the heart function as two separate pumps. The right side of the heart receives deoxygenated blood from the tissue into the right atrium. This blood then enters the right ventricle and from here it gets pumped through the pulmonary arteries into the lungs to get oxygen. This newly oxygenated blood flows back into the left atrium. From the left atrium the blood enters the left ventricle, and from here the blood gets pumped through the arteries back into the body.

The heart consists mostly of cardiac muscle called *myocardium*. The myocardium lies between a thin layer of epicardium on the outside and endocardium on the inside of the heart. The behaviour of the tissue of the heart is described by a material model which will be further explained in Chapter 3. The base is the area between the atria and

ventricles. The lower part is called the apex. An image of the heart is given in Figure 2.3.

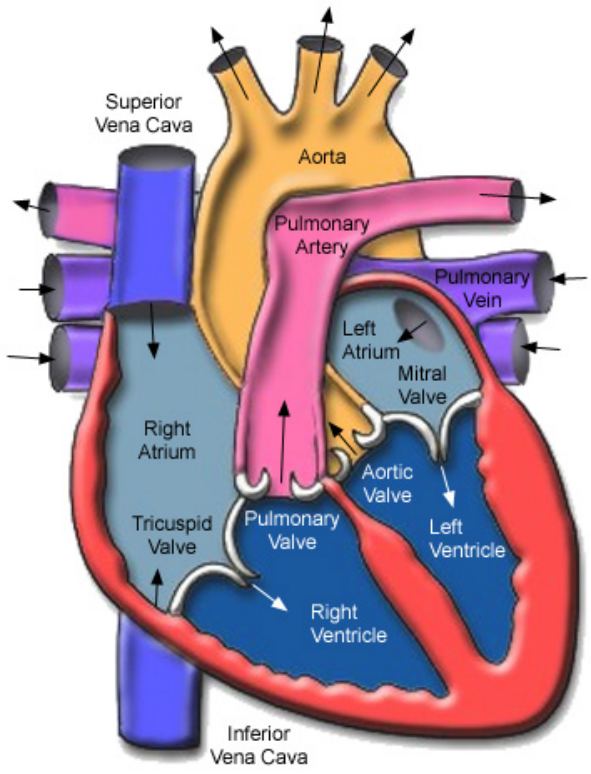


FIGURE 2.3: Anatomic figure with a longitudinal section of the heart https://lasersandtheheart.wikispaces.com/Mature+Heart+Anatomy.

The mechanical contraction of the heart - the PV-loop

The cycle of the left ventricle can be divided into four phases: the *filling phase*, the *isovolumic ventricular contraction*, the *ventricular ejection* and the *isovolumic ventricular relaxation*. To visualize the different phases of this mechanical cycle it is common to plot the pressure as a function of the volume. The resulting graph is referred to as a pressure-volume-loop (*PV-loop*) and is illustrated in Figure 2.4.

The first phase of the cardiac cycle is the passive filling. The atria and the ventricles are relaxing. The ventricles have just completed a contraction and the atria are getting filled with blood from the veins. When the pressure in the atria is higher than the pressure in the ventricles, the AV valves opens and the ventricles are passively filled with blood from the atria. The action potential in the atrial cells makes the chambers contract and squeeze blood into the ventricles. The passive filling in the late diastole stands for most of the filling of the ventricles. The atrial systole stands for about 20 % of the filling of the ventricles.

The isovolumic ventricular contraction is the phase when the ventricles contract and force the atrial valves to close. The vibration that occurs from this closure creates the first heart sound. The semilunar valves are also still closed. The semilunar valves prevent the blood from flowing from the arteries into the heart. The blood in the ventricles has nowhere to flow so the pressure in the ventricles rises. When the atria have finished the contraction the pressure in the atrias decreases and blood from the veins fill the atria with blood.

When the pressure in the ventricles is higher than the pressure in the arteries, the semilunar valves open. When these valves open, the blood in the ventricles is ejected into the arteries. This phase is called the ventricular ejection.

The last phase of the heart cycle is the isovolumic ventricular relaxation. As the ventricles relax after the contraction, the pressure in the ventricles fall. When the pressure is lower than in the arteries, the blood flows backwards and shuts the semilunar valves which creates the second heart sound. The phase is isovolumic as the pressure in the ventricles are still higher then in the atria such that the AV-valves are still closed.

Wiggers diagram is illustrated in Figure 2.5, and gives a more detailed picture of the electrical and mechanical events of the cardiac cycle.

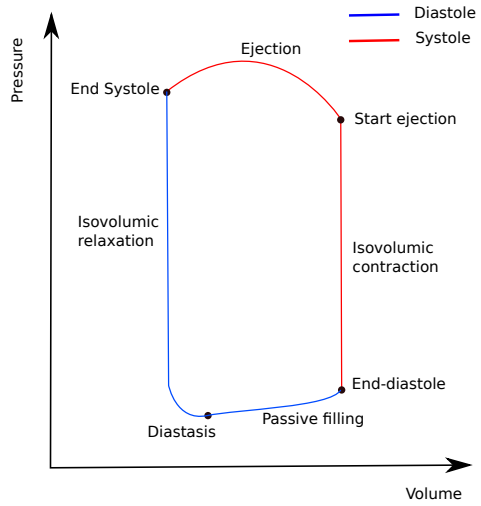


FIGURE 2.4: The pressure volume loop.

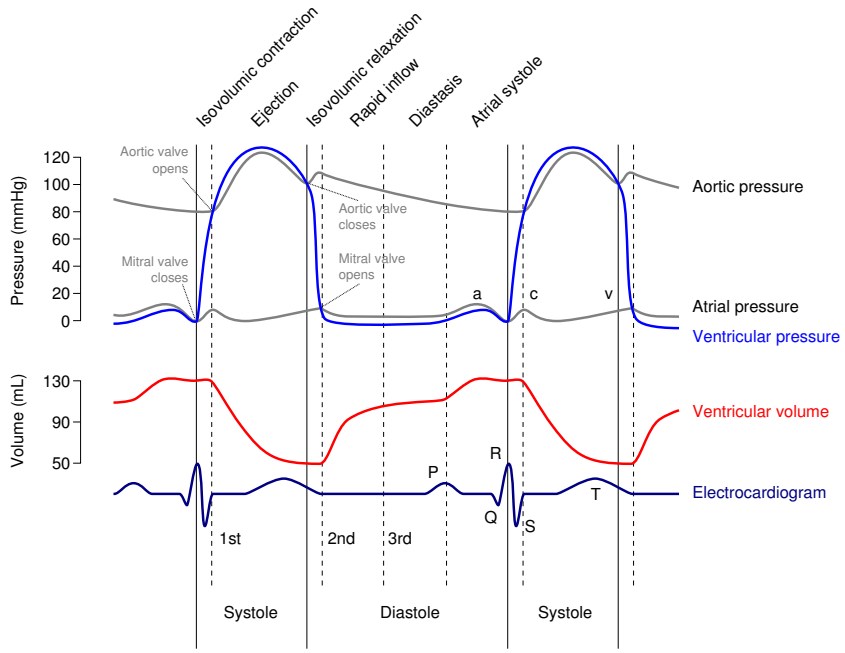


FIGURE 2.5: Wiggers diagram: pressure and volume plotted against time https://en.wikipedia.org/wiki/Wiggers_diagram.

In this chapter we have developed a basic understanding of the physiology of the heart. In the following chapter will describe the basic concepts of continuum mechanics, which we need to develop the mathematical model describing the mechanics of the left ventricle.

Chapter 3

Continuum mechanics

We will here give a brief introduction to the theory of continuum mechanics, which is necessary for understanding the mechanical models of the contracting heart. The main theory is taken from [2] and [3].

Continuum mechanics is used to describe different physical phenomena without taking internal micro structures into account. In continuum mechanics, the body $\mathcal{B}$ we are modelling is considered to be a continuous medium with a continuous distribution of particles. Both the mass and volume are considered to be continuous functions. A particle is a collection of many molecules, and must not be confused with the Newtonian point particle. In the following, we will define and describe important concepts that we will use to build the mechanical model of the left ventricle.

3.1 Kinematics and fundamental concepts

Let $\Omega_0 \subset \mathbb{R}^3$ and let $\vec{X} = \kappa_0(P,t)$ be a homeomorphic correspondence between a particle $P \in \mathcal{B}$ and a point $\vec{X} \in \Omega_0$ that $\mathcal{B}$ occupies at the initial time $t = 0$. A homeomorphic function is a bijective function in which the function itself and the inverse are continuous. Let now κ act on $\mathcal{B}$ to produce the region Ω at time t. The location that the particle occupies for a given time t is given by

$$\vec{x} = \kappa\left[\kappa_0^{-1}\left(\vec{X},t\right)\right] = \chi(\vec{X},t), \quad \forall \ \vec{X} \in \Omega_0 \ \ \forall \ t.$$

The motion of the body $\mathcal{B}$ is denoted by χ. The motion takes a point $\vec{X}$ in the reference configuration Ω_0 and maps it to the corresponding point $\vec{x} = (x_1, x_2, x_3)$ in the current configuration Ω. The motion is a diffeomorphism, i.e a bijective function where the

function itself and the inverse are smooth. The inverse of the motion maps a point in the current configuration back to the reference configuration. The inverse of the motion is defined as

$$\vec{X} = \chi^{-1}(\vec{x}, t).$$

An illustration of how the motion function operates is given in Figure 3.1.

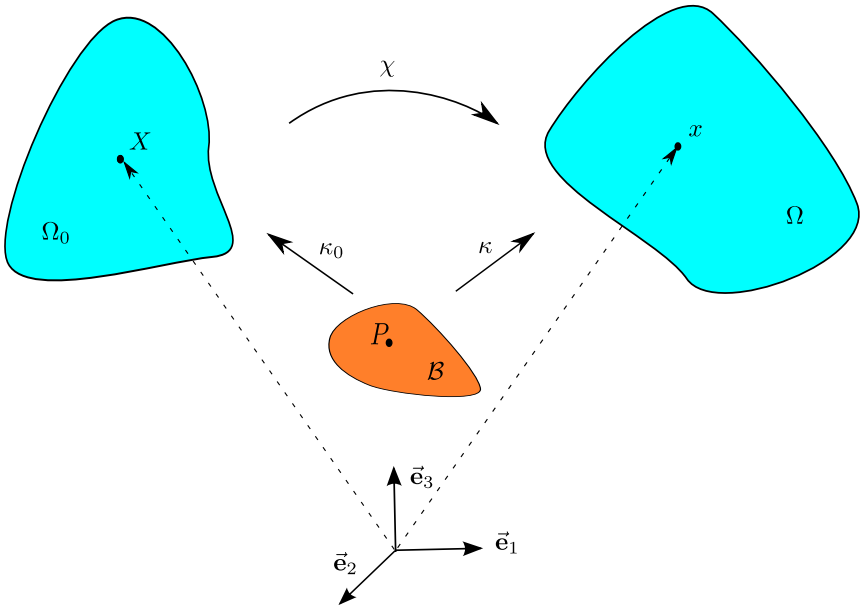


FIGURE 3.1: Illustration of the motion function.

There are different ways to describe the motion of a body. In the material description, also known as the Lagrangian description, we look at the different positions in space for the same particle as time goes by. The Lagrangian description is a common approach for solid mechanics, therefore this approach has been used in this master thesis.

Another alternative is the spatial description, also known as the Eulerian description, where the attention is on a particular point x in space. Here, we observe what happens at the same point in space as time goes by. The Eulerian description is typically used for fluid mechanics, and will not be used in this work.

We define the displacement field $\vec{u}$ is a vector field that relates the position vector in the reference configuration to the current configuration for a particle at a given time t

$$\vec{u}\left(\vec{X}, t\right) = \vec{x}(\vec{X}, t) - \vec{X}. \tag{3.1}$$

The derivatives of the deformed configuration, with respect to each component of the undeformed configuration, constitute a matrix that we call the deformation gradient. The deformation gradient transforms vectors in the undeformed geometry into the corresponding vectors in the deformed geometry. The deformation gradient $\mathbf{F}$ is a second order tensor, and is defined as

$$\mathbf{F} = \frac{d\chi\left(\vec{X}, t\right)}{d\vec{X}} = \frac{\partial \vec{x}}{\partial \vec{X}} = \begin{pmatrix} \frac{\partial x_1}{\partial X_1} & \frac{\partial x_1}{\partial X_2} & \frac{\partial x_1}{\partial X_3} \\ \frac{\partial x_2}{\partial X_1} & \frac{\partial x_2}{\partial X_2} & \frac{\partial x_2}{\partial X_3} \\ \frac{\partial x_3}{\partial X_1} & \frac{\partial x_3}{\partial X_2} & \frac{\partial x_3}{\partial X_3} \end{pmatrix}. \tag{3.2}$$

As the displacement can be written in terms of $\vec{x}$ we have that $\vec{x} = \vec{X} + \vec{u}$. The deformation gradient can then be written as

$$\mathbf{F} = \frac{\partial \vec{x}}{\partial \vec{X}} = \frac{\partial \vec{X}}{\partial \vec{X}} + \frac{\partial \vec{u}}{\partial \vec{X}} = \mathbf{I} + \nabla_0 \vec{u}. \tag{3.3}$$

Here, $\nabla_0 = \frac{\partial}{\partial \vec{X}}$ is the material displacement gradient. It is the differentiation operator with respect to the reference configuration, $\mathbf{I}$ is the identity matrix.

Note the following relationships between the material gradient and material divergence and spatial gradient and spatial divergence of the smooth scalar, vectors and tensor fields ϕ, u and $\mathbf{A}$ respectively:

$$\nabla \phi = \mathbf{F}^{-T} \nabla_0 \phi, \tag{3.4}$$

$$\nabla \vec{u} = \nabla_0 \vec{u} \mathbf{F}^{-1}, \tag{3.5}$$

$$\nabla \cdot \mathbf{A} = (\nabla_0 \cdot \mathbf{A})\, \mathbf{F}^{-T}. \tag{3.6}$$

The relations are derived by applying the chain rule.

3.2 The concept of strain

In order to get a precise measure of deformation of a continuum body, strain is introduced. There are many different definitions of strain, we will go through the most common in non-linear continuum mechanics.

Let $\vec{X}$ and $\vec{Y}$ be two neighbouring points in the reference geometry Ω_0.

Let us define $d\vec{X} = \vec{Y} - \vec{X}$, $d\epsilon = \|Y - \vec{X}\|$ and $\vec{a_0} = \frac{\vec{Y}-\vec{X}}{\|\vec{Y}-\vec{X}\|}$. Then $\vec{Y}$ can be written as

$$\vec{Y} = \vec{X} + \left(\vec{Y} - \vec{X}\right) = \vec{X} + \frac{\left(\vec{Y} - \vec{X}\right)\|\vec{Y} - \vec{X}\|}{\|\vec{Y} - \vec{X}\|\|} = \vec{X} + d\vec{X} = \vec{X} + \vec{a_0}d\epsilon.$$

Further, we have that $\vec{a_0}$ is the unit vector in the direction of the line element between the points $\vec{X}$ and $\vec{Y}$, and $d\epsilon$ is the distance between the two points.

Let $\chi\left(\vec{X}, t\right)$ be the motion function acting on the point $\vec{X}$ on the undeformed geometry. The motion function maps $\vec{X}$ on the undeformed geometry Ω_0 into the corresponding point $\vec{x}$ on the deformed geometry Ω, such that $\vec{x} = \chi\left(\vec{X}, t\right)$.

We want to investigate how the distance between the two neighbouring points $\vec{X}, \vec{Y}$ in Ω_0 is changed during the motion.

First, we Taylor expand the motion function $\chi(\vec{Y}, t)$:

$$\vec{y} = \chi\left(\vec{Y}, t\right) = \chi\left(\vec{X} + d\vec{X}, t\right) = \chi\left(\vec{X}, t\right) + d\vec{X}\frac{\partial\chi\left(\vec{X}, t\right)}{\partial\vec{X}} + \mathcal{O}\left(\|d\vec{X}\|^2\right).$$

The difference in the deformed geometry thus becomes

$$\vec{y} - \vec{x} = d\vec{X}\frac{\partial\chi\left(\vec{X}, t\right)}{\partial\vec{X}} + \mathcal{O}\left(\|d\vec{X}\|^2\right) = d\vec{X}\mathbf{F} + \mathcal{O}\left(\|d\vec{X}\|^2\right) = \vec{a_0}d\epsilon\mathbf{F} + \mathcal{O}\left(\|d\vec{X}\|^2\right).$$

This shows that for small distances $\|d\vec{X}\| = \|\vec{Y} - \vec{X}\|$, the term $\vec{a_0}d\epsilon\mathbf{F}$ is an approximation for the relative motion $\vec{y} - \vec{x}$.

Now we define the stretch vector $\vec{\lambda_{a_0}} = \vec{a_0}\mathbf{F}$. The length of the stretch vector

$$\lambda = \|\vec{\lambda_{a_0}}\| \tag{3.7}$$

is called the stretch. The length of the vector between $\vec{x}$ and $\vec{y}$ has the following 1st order approximation

$$\|\vec{y} - \vec{x}\| = \left[\left(\vec{y} - \vec{x}\right)^T\left(\vec{y} - \vec{x}\right)\right]^{\frac{1}{2}} = \left[\left(\vec{\lambda_{a_0}}d\epsilon\right)^T \cdot \vec{\lambda_{a_0}}d\epsilon\right]^{\frac{1}{2}} = \lambda d\epsilon.$$

We are ready to introduce the right Cauchy-Green deformation tensor $\mathbf{C} = \mathbf{F}^T\mathbf{F}$. Let us take the square of the stretch

$$\lambda^2 = (\vec{a_0}\mathbf{F})^T \cdot \vec{a_0}\mathbf{F} = \vec{a_0}^T\mathbf{F}^T\mathbf{F}\vec{a_0} = \vec{a_0}^T\mathbf{C}\vec{a_0}. \tag{3.8}$$

Using the right Cauchy-Green tensor, we only need the direction $\vec{a_0}$ of the fiber in order to compute the stretch of the fiber.

$\mathbf{C}$ is a symmetric matrix since

$$\mathbf{C} = \mathbf{F}^T\mathbf{F} = (\mathbf{F}^T\mathbf{F})^T = \mathbf{C}^T.$$

$\mathbf{C}$ is a positive definite matrix since $\forall\ \vec{V} \in \Omega_0 \setminus \{\vec{0}\}$, $\vec{V}^T\mathbf{C}\vec{V} = \vec{V}^T\mathbf{F}^T\mathbf{F}\vec{V} = \|\mathbf{F}\vec{V}\|_2 > 0$.

A different way to measure strain is to look at the change in the inner product product between two vectors on the reference configuration, and two vectors on the current configuration.

$$\begin{aligned}
&\frac{1}{2}\left(d\vec{x}^T \cdot d\vec{y} - d\vec{X}^T \cdot d\vec{Y}\right) \\
&= \frac{1}{2}\left(d\vec{X}^T\mathbf{F}^T \cdot \mathbf{F}d\vec{Y} - d\vec{X} \cdot d\vec{Y}\right) \\
&= \frac{1}{2}\left(d\vec{X}^T\mathbf{C}d\vec{Y} - d\vec{X} \cdot d\vec{Y}\right) \\
&= d\vec{X}^T\frac{1}{2}\left(\mathbf{C} - I\right)d\vec{Y}.
\end{aligned}$$

Here,

$$\mathbf{E} = \frac{1}{2}\left(\mathbf{C} - \mathbf{I}\right) \tag{3.9}$$

is the Green-Lagrange strain tensor. The tensors $\mathbf{C}$ will be used in the description of hyperelastic material, whilst the tensor $\mathbf{E}$ will be used in the testing of our model in Chapter 6.

We have now developed a way to measure the strain using a strain tensor operating on points in Ω_0.

The invariants of a tensor are quantities that are independent of the coordinate system. They are the coefficients of the characteristic polynomial of a matrix. It can be shown that the material model we will use for the myocardium depend on the invariants only. The invariants of $\mathbf{C}$ are defined as

$$I_1(\mathbf{C}) = \Sigma_i \mathbf{C}_{ii} = \operatorname{tr} \mathbf{C}, \tag{3.10}$$

$$I_2(\mathbf{C}) = \frac{1}{2}\left(I_1(\mathbf{C})^2 - \Sigma_{i,j} C_{ij} C_{ij}\right) \tag{3.11}$$

$$I_3(\mathbf{C}) = \det \mathbf{C} = \mathbf{J}^2, \tag{3.12}$$

where $\mathbf{J} = \det(\mathbf{F})$.

3.3 The concept of stress

There are different forces acting on a body. These forces can be divided into external and internal forces. External forces are force fields, like gravity and magnetism, acting on the whole of the body $\mathcal{B}$. External forces may also include forces acting on the external surface of the body. Internal forces are traction forces acting on "small" imaginary surfaces within the body. Stress is a measure of average force per unit area at a point x within, or on the boundary of the body $\mathcal{B}$.

Let us look at the body $\mathcal{B}$ occupying some region in space Ω at time t. If we cut the body in two halves and look at some infinitesimal imaginary surface ds on the lower half containing the point x, the resultant force df acting on ds is defined as

$$df = \vec{t} \cdot ds.$$

Here, $\vec{t}$ is the Cauchy traction vector describing the force per unit area in the deformed configuration

$$\vec{t} = \lim_{\Delta s \to 0} \frac{\Delta \vec{f}}{\Delta s} = \frac{\vec{df}}{ds}.$$

The corresponding traction vector on the undeformed geometry $\vec{T}$ on a small surface element on the undeformed geomtry dS is called the first Piola-Kirchhoff traction vector, and is related to the Cauchy traction vector through the relation

$$\vec{df} = \vec{t} \cdot ds = \vec{T} \cdot dS, \tag{3.13}$$

which is called Cauchy's postulate.

The first Piola-Kirchhoff traction vector points in the directon of $\vec{t}$, and relates forces on the deformed geometry to quantities on the undeformed geometry.

This leads us to

Cauchy's stress theorem. There exist unique second-order tensor fields σ and $\mathbf{P}$ such that

$$\vec{t}(\vec{x}, t, \vec{n}) = \sigma\vec{n} \tag{3.14}$$

$$\vec{T}(\vec{X}, t, \vec{N}) = \mathbf{P}\vec{N}. \tag{3.15}$$

Here, σ is the Cauchy stress tensor matrix and $\mathbf{P}$ is the first Piola-Kirchhoff stress tensor matrix. The vectors $\vec{N}$ and $\vec{n}$ are the unit normal vectors on the reference and current configuration, respectively. The Cauchy stress tensor matrix can be written as

$$\sigma = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{pmatrix}. \tag{3.16}$$

Here, σ_{11}, σ_{22} and σ_{33} are normal stresses, and $\sigma_{12}, \sigma_{13}, \sigma_{23}, \sigma_{31}, \sigma_{32}$ and σ_{21} are shear stresses. The components of the stress tensor are illustrated in Figure 3.2.

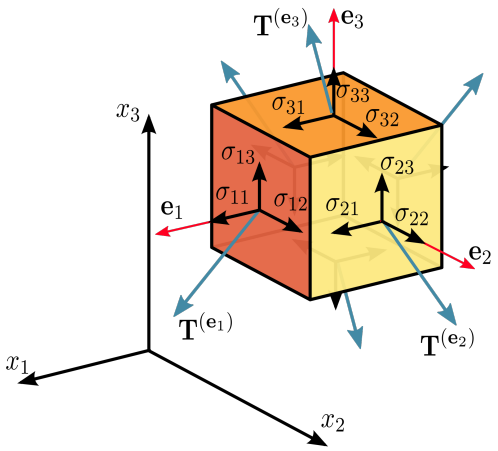


FIGURE 3.2: components of the Cauchy stress tensor acting on a cubic element https://en.wikipedia.org/wiki/Cauchy_stress_tensor.

We are now interested in a relation between the Cauchy stress tensor and the Piola-Kirchhoff stress tensor.

Theorem. [4] Let U be an open set in $\mathbb{R}^N$ and $\phi : U \to \mathbb{R}^N$ be an injective differentiable function with continuous partial derivatives, the Jacobian of which is nonzero $\forall$ x in U. Then for all real valued, compactly supported, continuous functions f, with support

contained in $\phi(U)$,

$$\int_{\phi(U)} f(v)dv = \int_U f(\phi(U))|\det \frac{\partial \phi}{\partial u}|du. \tag{3.17}$$

We consider only bounded sets Ω, $\Omega_0 \subseteq \mathbb{R}^3$ and define $\bar{\Omega} = \Omega + \partial\Omega$. We also assume that $\frac{\partial \chi}{\partial X} \neq 0$. This is a reasonable assumption for all practical purposes for this thesis.

We have that $\chi : \Omega_0 \to \Omega$ is injective and continuously differentiable with respect to both time and space [2].

Let us now define the function

$$f : \Omega \to \Omega_0$$
$$f = 1.$$

Then supp $f = \bar{\Omega}$ so that f has compact support. Furthermore, f is continuous on the open set Ω.

We can now write

$$\int_{\chi(V)} f(v)dv = \int_V f(\chi(V))|\det\left(\frac{\partial \chi}{\partial X}\right)|dV.$$

As we integrated over an arbitrary volume element, and as $f(x) = 1$, we have that

$$\int_{\chi(V)} dv = \int_V JdV,$$

as v is arbitarty

$$dv = JdV,$$
$$J = \det \mathbf{F} = \det\left(\frac{\partial \chi}{\partial X}\right).$$

In this chapter we have described the basics concepts of continuum mechanics needed to develop the mathematical model for cardiac mechanics. The derivation of the mathematical model for the cardiac mechanics will be given in the next chapter.

We now look at some infinitesimal volume element in the reference configuration, expressed by an area $\vec{dS} = dS\vec{N}$ and a vector $d\vec{Y}$

$$dV = d\vec{Y}^T d\vec{S}.$$

The corresponding volume in the current configuration is

$$dv = d\vec{y}^T d\vec{s} = JdV = Jd\vec{Y}^T dS.$$

By expressing dy in terms of the deformation gradient, we obtain

$$\left(\mathbf{F}d\vec{Y}\right)^T d\vec{s} = Jd\vec{Y}^T d\vec{S},$$

and further

$$d\vec{Y}^T \mathbf{F}^T d\vec{s} = d\vec{Y}^T Jd\vec{S}.$$

This shows us that infinitesimal areas in the current and reference configurations are related through

$$d\vec{s} = \mathbf{F}^{-T} Jd\vec{S}, \tag{3.18}$$

which is referred to as Nansons formula.

Now, we can easily find a relation between the Cauchy stress tensor σ and the first Piola-Kirchhoff stress tensor $\mathbf{P}$.

From 3.13 and 3.15 we have that

$$\sigma\vec{n}ds = \mathbf{P}\vec{N}dS.$$

By applying Nanson's formula we obtain

$$J\sigma\mathbf{F}^{-T} dS\vec{N} = \mathbf{P}dS\vec{N}.$$

This gives us the relation

$$\sigma = J^{-1}\mathbf{P}\mathbf{F}^T \tag{3.19}$$

which is referred to as the Piola transformation.

In this chapter we have provided an introduction to the basic concepts of continuum mechanics. Next, we will apply these concepts and describe the mathematical model for the cardiac mechanics.

Chapter 4

The mathematical model

In this chapter we will derive the partial differential equation for the mechanics of the heart. We will also formulate the boundary conditions for the left ventricle geometry. The incompressible and compressible Neo-Hookean model for the modelling of the passive behaviour of the tissue will be introduced. We will also explain the mathematics behind modelling the active contraction of the muscle fibers. Lastly, we will determine the variational form for the heart equations, and the equivalent minimization of potential energy formulation both for compressible and incompressible models.

4.1 Mechanical model for the heart

Newton's second law applied to a material region on the current configuration is [5]

$$\frac{d}{dt}\int_{\Omega}\rho\vec{v}dV = \Sigma\vec{F}. \tag{4.1}$$

Here, $\vec{v}$ is the velocity of a volume unit, ρ is the mass density, and $\Sigma\vec{F}$ is the resultant force acting on the material region.

It is common to divide the forces acting on the region Ω into body forces $\vec{F_B}$,

$$\vec{F_B} = \int_{\Omega}\vec{f_B}(\vec{x},t)dV, \tag{4.2}$$

and surface forces F_S,

$$\vec{F_S} = \int_{\partial\Omega}\vec{f_S}(dS,t)dV. \tag{4.3}$$

We know that for a small surface element dS, the force acting on dS is given by

$$\vec{f_S}(dS,t) = \sigma \vec{n} dS.$$

Hence, Newton's second law can be written as

$$\frac{d}{dt}\int_\Omega \rho \vec{v} dV = \int_\Omega \vec{f_B} dV + \int_{\partial\Omega} \sigma \cdot \vec{n} dS. \tag{4.4}$$

By applying the divergence theorem to the surface forces we get

$$\int_{\partial\Omega} \sigma \cdot \vec{n} dS = \int_\Omega \nabla \cdot \sigma dV, \tag{4.5}$$

where $\nabla = \frac{\partial}{\partial \vec{x}}$ is the spatial displacement gradient. This gives us the equation

$$\frac{d}{dt}\int_\Omega \rho \vec{v} dV = \int_\Omega \vec{f_B} dV + \int_\Omega \nabla \cdot \sigma dV. \tag{4.6}$$

By assuming that $\vec{v}$ is a continuously differentiable vector field we can interchange the order of integration and differentiation and write

$$\int_\Omega \left(\vec{f_B} + \nabla \cdot \sigma - \rho \frac{d\vec{v}}{dt} \right) dV = 0. \tag{4.7}$$

As we integrate over an arbitrary region and assume the integrand to be continuous everywhere, this means, the integrand itself must be zero. This gives us Cauchy's first equation of motion:

$$\vec{f_B} + \nabla \cdot \sigma - \rho \frac{d\vec{v}}{dt} = 0. \tag{4.8}$$

The term $\rho \frac{d\vec{v}}{dt}$ represents the inertial forces. If we assume that both the inertial forces and the body forces are neglectable compared to the surface forces representing the internal stresses, the equation becomes

$$\nabla \cdot \sigma = 0. \tag{4.9}$$

We are now interested in a relation between the reference configuration and the current configuration.

From (3.19) we have that $\mathbf{P} = J\sigma\mathbf{F}^{-T}$. Let us now look at the divergence of $\mathbf{P}$. By remembering that J is scalar and using the properties of the divergence (3.6), we can write

$$\begin{aligned}\nabla_0 \cdot \mathbf{P} = \nabla_0 \cdot \left(J\sigma\mathbf{F}^{-T}\right) = \nabla_0 \cdot \left(\sigma\left(J\mathbf{F}^{-T}\right)\right) \\ = \left(\nabla_0 \cdot \sigma\right) J\mathbf{F}^{-T} + \left(\nabla_0 \cdot \left(J\mathbf{F}^{-T}\right)\right)\sigma.\end{aligned}$$

By applying the divergence theorem, we have that

$$\int_{\Omega_0} \nabla_0 \cdot \left(J\mathbf{F}^{-T}\right) dV = \int_{\partial\Omega_0} J\mathbf{F}^{-T}\vec{N}dS.$$

If we now use Nansons formula and apply the divergence theorem again, we get

$$\int_{\partial\Omega_0} J\mathbf{F}^{-T}\vec{N}dS = \int_{\partial\Omega} \mathbf{I}\vec{n}ds = \int_{\Omega} \nabla \cdot \mathbf{I}dv = \vec{0}.$$

This is true as $\nabla \cdot \mathbf{I}$ is the zero vector $\vec{0}$. As we are integrating over an arbitrary volume, the integrand $\nabla_0 \cdot \left(J\mathbf{F}^{-T}\right)$ must be zero.

By applying (3.6) and using that $\nabla_0 \cdot \left(J\mathbf{F}^{-T}\right) = 0$, we get that

$$\nabla_0 \cdot \mathbf{P} = \left(\nabla_0 \cdot \sigma\right) J\mathbf{F}^{-T} = J\nabla\sigma. \tag{4.10}$$

This means that we have the two equivalent problems

$$\nabla \cdot \sigma = \nabla_0 \cdot \mathbf{P} = 0, \tag{4.11}$$

which are referred to as the *equilibrium equations of the heart*.

4.1.1 Boundary conditions

The boundary conditions are chosen to reflect the physical reality. In the $x-$direction on the base, we force the heart not to move. This is modelled by imposing a Dirichlet

condition. In the $y, z-$direction of the boundary on the base we want an elastic displacement condition, like a spring holding against the displacement. This is achieved by applying a Robin condition in the $y, z-$plane. For the epicardium, we set a Neumann boundary condition with a constant pressure σ_1. This pressure is due to the pressure from the blood vessels. For the endocardial boundary, we also impose a Neumann condition with a time varying pressure $\sigma_2(t)$ to simulate the time varying pressure throughout the cardiac cycle. A plot of the boundaries is given in Figure 4.1, where the endocardium is marked in grey, the epicardium is marked in brown and the base in red.

FIGURE 4.1: Illustration of the left ventricle with color marking on the endocardium (grey), epicardium (brown) and base (red).

4.1.2 Constitutive models for passive myocardium

In order to describe the real behaviour of the heart tissue, constitutive theories are needed. To assure realistic tissue behaviour, the heart is usually modelled as a homogeneous, incompressible, hyperelastic and orthotropic material [6].

Different models for the passive behaviour of the myocardium have been proposed [7, 8]. The simplest and earliest models considered the myocardium to be linear and isotropic, that is the material properties are equal along all basis vectors. There exist transversely isotropic models [7], and also ortotropic models [6] which are considered to capture the tissue behaviour realistic. In this master thesis, we will not consider material models of high complexity, as the time efficiency is the key feature of our solver. The material models considered are therefore non-linear and isotropic.

For hyperelastic materials, the energy is conserved, and the work done during the deformation is path independent. In other words the work done on a body is only dependent on the initial state in the reference configuration at time t_0 and the current state at some time t. For hyperelastic materials there exists a strain energy function $\Psi(\mathbf{F})$. This function describes the work done by the stresses from the reference configuration to the current configuraiton. The strain energy function is defined per unit reference volume.

In hyperelastic theory, the first Piola-Kirchhoff stress tensor, and the Cauchy stress tensor can be expressed in terms of the strain energy function as

$$\mathbf{P} = \frac{\partial \Psi}{\partial \mathbf{F}}, \quad \sigma = J^{-1}\mathbf{F}\frac{\partial \Psi}{\partial \mathbf{F}}. \tag{4.12}$$

The modified Saint Venant-Kirchhoff material model will be used for verification. It is one of the simplest hyperelastic material models and is given by

$$\Psi = \frac{\lambda}{2}\left(tr(E)\right)^2 + \mu\,\mathrm{tr}(E^2). \tag{4.13}$$

Here, $\mathbf{E}$ is the Green-Lagrange strain tensor, and μ and λ are material parameters.

The heart tissue can be assumed to be incompressible and hyperelastic. The Neo-Hookean material model is often used to model materials that are undergoing large deformations. It is a non-linear , hyperelastic and isotropic model. It exist in a compressible and an incompressible version given by

$$\Psi = C_1\left(I_1 - 3\right) - p\left(J - 1\right). \tag{4.14}$$

and

$$\Psi = C_1\left(\bar{I}_1 - 3\right) - D_1(J-1)^2, \tag{4.15}$$

$$\bar{I}_1 = J^{-\frac{2}{3}}\,\mathrm{tr}(\mathbf{C}), \tag{4.16}$$

respectively.

Here C_1 and D_1 are material constants describing the stiffness of the material, the units for these constants are Pascal.

4.1.3 Active force development

The heart cells contract due to electrical stimulation. The active stress can be described in various ways. We will in the following describe the active stress in two ways. The first

approach is to introduce an active part to the stress tensor. Such that the Cauchy stress tensor can be decomposed into a passive and an active part:

$$\sigma = \sigma_{passive} + \sigma_{active}.$$

The active force is developed along the fiber direction. Let $\vec{f}$ be a unit vector field oriented in the fiber direction. We can define the active force with respect to either the deformed or undeformed geometry. By defining the force with respect to the deformed geometry, we get the following expression for the total Cauchy stress:

$$\sigma = \frac{1}{J}\frac{\partial \Psi_{passive}}{\partial \mathbf{F}} + \frac{\sigma_a\left([Ca^{2+}]\right)}{J}\left(\mathbf{F}\vec{f}_0\right)\left(\mathbf{F}\vec{f}_0\right)^T. \tag{4.17}$$

Here, $\sigma_a\left([Ca^{2+}]\right)$ is some scalar function describing the calcium-dependent active force development. The direction of the active force generation is described by the tensor product $\vec{f}\vec{f}^T$. In this thesis we will asume that the fibers are oriented along the circumference of the undeformed geometry. The implementation of the circumferential fiber field is given in Chapter 5.

Using the Piola transfrom: $\mathbf{P} = J\sigma\mathbf{F}^{-T}$, the Piola-Krichhoff stress tensor, decomposed into an passive and active part, becomes:

$$\begin{aligned}\mathbf{P} &= \mathbf{P}_{passive} + \mathbf{P}_{active}\\ \mathbf{P} &= \frac{\partial \Psi_{passive}}{\partial \mathbf{F}} + \sigma_a\left([Ca^{2+}]\right)\mathbf{F}\vec{f}_0\vec{f}_0^{\,T},\end{aligned}$$

here, $\vec{f}_0$ denotes the fibers on the undeformed geometry.

For hyperelastic materials the expression for the Piola Kirchhoff stress tensor can be obtained by differentiating the strain energy function Ψ. The active contraction of the fibers can be described from the scalar function Ψ_{active}

$$\Psi_{active} = \sigma_a\left([Ca^{2+}]\right)\vec{f}_0^{\,T}\mathbf{C}\vec{f}_0 = \sigma_a\left([Ca^{2+}]\right)\lambda. \tag{4.18}$$

We decompose the strain energy function into an passive and active part:

$$\Psi = \Psi_{passive} + \Psi_{active}.$$

This is not a strain energy function as there is no conservation of energy, but this is a practical tool that simplifies the analysis of the problem.

4.2 Finite element formulation of the heart equations

We are now ready to formulate the finite element problem for the heart equations. We will formulate the weak formulation, as well as the equivalent formulation of the minimization of potential energy. This will be done both for compressible and incompressible material models.

4.2.1 Finite element formulation for compressible material models

Let $\partial H_1(0)$ the base, $\partial H_2(0)$ the endocardium and $\partial H_3(0)$ the epicardium. The union of the boundaries is written as $\partial H = \bigcup_{i=1}^{3} \partial H_i$.

The equilibrium equation for the heart on the undeformed geometry, together with the boundary conditions, is given by

$$\nabla_0 \cdot \mathbf{P} = 0 \ \text{ in } \ H(0) \tag{4.19}$$

$$\begin{aligned} &u_1 = 0 \ \text{ on } \ \partial H_1(0), \\ &\vec{T} = -k\vec{u} \ \text{ in } \ \partial H_1(0), \\ &\vec{T} = J\sigma_1 \mathbf{F}^{-T}\vec{N} \ \text{ on } \ \partial H_2(0), \\ &\vec{T} = J\sigma_2(t)\mathbf{F}^{-T}\vec{N} \ \text{ on } \ \partial H_3(0). \end{aligned} \tag{4.20}$$

Here, $\vec{u} = (u_1, u_2, u_3)$, and k is a scalar value representing the stiffness of the spring. The unknown is the displacement $\vec{u}$ and $\mathbf{P}$ typically depends on $\vec{u}$ in a non-linear way. Hence, the problem is non-linear. The solution method is a non-linear finite element formulation.

We search for a solution in the Sobolev space $H^1(H(0))$, which is defined as

$$H^1\left(H(0)\right) = \{\vec{v} \in L^2\left(H(0)\right) : \int_H \vec{v}^2 dx < \infty \text{ and } \int_H |\nabla \vec{v}|^2 dx < \infty\}.$$

We define $\hat{V}$ and V to be the test and trial space respectively as

$$\begin{aligned} \hat{V} &= \{\vec{v} \in H^1\left(H(0)\right) : \vec{v} = 0 \text{ on } \partial H\} \\ V &= \{\vec{v} \in H^1\left(H(0)\right) : \vec{v} \text{ satisfies (4.20) on } \partial H\}. \end{aligned}$$

To derive the non-linear variational form of the equilibrium equation of the heart, we multiply the differential equation with a test function $\vec{v} \in \hat{V}$. Thereafter we integrate over the domain and use Green's theorem to write the variational form:

$$\int_{H(0)} \nabla \cdot (\mathbf{P}) \cdot \vec{v} dV = \int_{\partial H(0)} \mathbf{P} \cdot \vec{N} \vec{v} dS - \int_{H(0)} \mathbf{P} \nabla \vec{v} dV.$$

The boundary is divided into three separate parts which gives us that

$$\int_{\partial H(0)} \mathbf{P} \cdot \vec{N} \vec{v} dS = \int_{\partial H_1} \mathbf{P} \cdot \vec{N} \vec{v} dS + \int_{\partial H_2} \mathbf{P} \cdot \vec{N} \vec{v} dS + \int_{\partial H_3} \mathbf{P} \cdot \vec{N} \vec{v} dS.$$

We now insert the boundary conditions from the boundary value problem and get the variational form $L(u; v) = 0$, where

$$L(\vec{u}; \vec{v}) = \int_{H(0)} \mathbf{P} \nabla \vec{v} dV + \int_{\partial H_1} k \vec{u} \vec{v} dS - \int_{\partial H_2} J \sigma_1 \mathbf{F}^{-T} \vec{N} \vec{v} dS - \int_{\partial H_3} J \sigma_2(t) \mathbf{F}^{-T} \vec{N} \vec{v} dS = 0. \tag{4.21}$$

Problem (4.19) can be formulated weakly as: find $\vec{u} \in V$ such that

$$L(\vec{u}; \vec{v}) = 0 \ \ \forall \vec{v} \in \hat{V}.$$

The function $L : V \times \hat{V} \to \mathbb{R}$ is a semi-linear form, nonlinear in the argument $\vec{u}$, and linear in $\vec{v}$.

In order solve this problem we construct a finite dimensional discretization of the test and the trial spaces, respectively $\hat{V}_h \subset \hat{V}$ and $V_h \subset V$.

The choices of discrete spaces depend on the selection of element type. A common element type is linear tetrahedral elements. That is, we partition the domain H into a finite number of tetrahedral elements K, where T_h denotes the set of all elements K.

Then,

$$\hat{V}_h = \{\vec{v} \in C\left(H(0)\right) : \vec{v} = 0 \text{ on } \ \partial H, \vec{v}|_K \text{ linear } \forall K \in T_h\}$$
$$V_h = \{\vec{v} \in C\left(H(0)\right) : \vec{v} \ \text{ satisfies } \ (4.20) \text{ on } \partial H, \vec{v}|_K \text{ linear } \forall K \in T_h\}.$$

Here, $C(H(0))$ is the space of continuous functions on $H(0)$, and $\vec{v}|_K$ denotes the linear basis functions on the elements. An illustration of these basis functions is given in Figure 4.2.

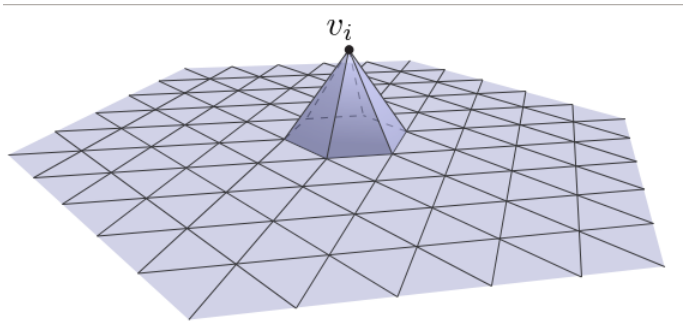


FIGURE 4.2: Illustration of the basis functions $\vec{v}|_K$ on the tetrahedral elements.

The discrete non-linear problem reads: find $\vec{u_h} \in V_h$ such that

$$L(\vec{u}_h; \vec{v}) = 0 \ \ \forall \vec{v} \in \hat{V}_h. \tag{4.22}$$

Let ϕ_j denote the piecewice linear Lagrangian basis functions for V_h. Then $\vec{u}_h = \sum_{j=1}^{N} U_j \phi_j$, and U_j are the coefficients.

We can write this as

$$L_i(\vec{u}_h, \hat{\phi}_i) = b(U_j) \text{ for } i, j = 1, ..., N.$$

We now have a non-linear system of equations with $v = \hat{\phi}_i$ with $b(U_j) = 0$ for$j = 1, ..N$, and $b : \mathbb{R}^N \to \mathbb{R}^N$.

4.2.2 Minimization of the potential energy

In this section we will derive the equivalent formulation of the weak form by applying the principle of stationary potential energy [2]. The minimization of potential energy formulation is easier to implement in FEniCS.

As we model the heart as a hyperelastic material, we assume the system that is modelled to be conservative. By assuming this, we require the existence of an energy functional Π, which describes the total potential energy of the system. The heart equations is derived from Cauchy's first equation of motion, and only the internal stresses are considered. Hence, the total potential energy is given by

$$\Pi(\vec{u}) = \int_{H(0)} \Psi(\mathbf{F}(\vec{u}))\, dV. \tag{4.23}$$

We are now interested in the equilibrium state of the system, that is stationary position of the potential energy. The equilibrium state is found by requiring the Gâteaux derivative of Π, with respect to the displacement $\vec{u}$ to be zero in every direction:

$$\lim_{\epsilon \to 0} \frac{\Pi\left(\vec{u} + \epsilon \vec{v}\right) - \Pi\left(\vec{u}\right)}{\epsilon} = 0. \tag{4.24}$$

In the following we will show that the minimal potential energy formulation is equivalent to the weak formulation.

By inserting (4.23) into (4.24) we get the expression.

$$\lim_{\epsilon \to 0} \frac{\int_{H(0)} \Psi\left(\mathbf{F}\left(\vec{u} + \epsilon \vec{v}\right)\right) dV - \int_{H(0)} \Psi\left(\mathbf{F}\left(\vec{u}\right)\right) dV}{\epsilon} = 0. \tag{4.25}$$

As the composition of Ψ and $\mathbf{F}$ is continuously differentiable we can pull the limit inside the integral. This leads us to the expression

$$\int_{H(0)} \left[\lim_{\epsilon \to 0} \frac{\Psi\left(\mathbf{F}\left(\vec{u} + \epsilon \vec{v}\right)\right) - \Psi\left(\mathbf{F}\left(\vec{u}\right)\right)}{\epsilon} \right] dV = 0. \tag{4.26}$$

The deformation gradient $\mathbf{F}(\vec{u} + \epsilon \vec{v})$ can be written in terms of its argument as

$$\mathbf{I} + \nabla_0 \vec{u} + \epsilon \nabla_0 \vec{v} = \mathbf{F}(\vec{u}) + \Delta \mathbf{F}.$$

Here, $\Delta \mathbf{F} = \epsilon \nabla_0 \vec{v}$.

$$\lim_{\Delta \mathbf{F} \to 0} \frac{\Psi(\mathbf{F} + \Delta \mathbf{F}) - \Psi(\mathbf{F})}{\Delta \mathbf{F}} = \frac{\partial \Psi}{\partial \mathbf{F}}$$

$$\lim_{\Delta \mathbf{F} \to 0} \frac{\Psi(\mathbf{F} + \Delta \mathbf{F}) - \Psi(\mathbf{F})}{\Delta \mathbf{F}} \frac{\Delta \mathbf{F}}{\epsilon} = \frac{\partial \Psi}{\partial \mathbf{F}} \frac{\Delta \mathbf{F}}{\epsilon}$$

We have that $\Delta \mathbf{F} \to 0$ as $\epsilon \to 0$.

This implies that

$$\lim_{\epsilon \to 0} \frac{\Psi(\mathbf{F} + \Delta\mathbf{F}) - \Psi(\mathbf{F})}{\Delta\mathbf{F}} = \lim_{\Delta\mathbf{F} \to 0} \frac{\Psi(\mathbf{F} + \Delta\mathbf{F}) - \Psi(\mathbf{F})}{\Delta\mathbf{F}} = \frac{\partial \Psi}{\partial \mathbf{F}} = \mathbf{P}.$$

We also have that

$$\frac{\Delta\mathbf{F}}{\epsilon} = \frac{\epsilon \nabla_0 \vec{v}}{\epsilon} = \nabla_0 \vec{v}.$$

This shows that

$$\lim_{\epsilon \to 0} \frac{\Pi(\vec{u} + \epsilon\vec{v}) - \Pi(\vec{u})}{\epsilon} = \int_{H(0)} \mathbf{P} \nabla_0 dV.$$

Which shows the equivalence of the minimization formulation and the weak form of the boundary value problem.

4.2.3 Mixed finite element formulation for incompressible material models

Many materials can sustain finite strains without noticeable volume changes. The only motions that are possible for these materials are isochoric, that is volume preserving motions. These materials are assumed to be incompressible. To model the incompressibility, we impose a constant volume constraint through the determinant of the deformation gradient J. In other words we solve $\nabla \mathbf{P}$ subject to $J = 1$.

A mixed formulation is used to solve these kind of problems. The mixed formulation is a two field variational principle. Besides the displacement field we introduce a pressure field $p \in L^2(H(0))$ which is treated as an independent variable. Here, p is the Lagrange multiplier of the optimization problem.

The general strain energy function for incompressible materials can be written as

$$\Psi = \Psi(\mathbf{F}) + p\,(J - 1)\,. \tag{4.27}$$

We will decompose Ψ into a passive part, involving the passive behaviour of the tissue, an active part, involving the active contraction of the cells, and an incompressible part. We define $\Psi_{tot} = \Psi_{passive} + \Psi_{active} + \Psi_{incompressible}$.

The Piola-Kirchhoff stress tensor is also decomposed into

$$\mathbf{P} = \mathbf{P}_{passive} + \mathbf{P}_{active} + \mathbf{P}_{incompressible}$$
$$\mathbf{P} = \frac{\partial \Psi_{passive}}{\partial \mathbf{F}} + \mathbf{P}_{active} + \frac{\partial \Psi_{incompressible}}{\partial \mathbf{F}}$$
$$\mathbf{P} = \frac{\partial \Psi_{passive}}{\partial \mathbf{F}} + \mathbf{P}_{active} + pJ\mathbf{F}^{-T}$$

To solve this problem we need to formulate the Galerkin method for mixed problems. In the compressible problem, we only solved for the displacement. In the incompressible case we have a two field equation system, and we wish to solve the following problem: find $\vec{u} \in V$ and $p \in L^2(H(0))$, such that

$$L(\vec{u}, \vec{v}) + b(\vec{v}, p) = 0 \; \forall \; \vec{v} \in \hat{V}$$
$$c(\vec{u}, w) = 0 \; \forall \; w \in L^2(H(0))$$

The functions $L : V \times \hat{V} \to \mathbb{R}$, $b : \hat{V} \times L^2(H(0)) \to \mathbb{R}$ and $c : V \times L^2(H(0)) \to \mathbb{R}$ are given by

$$L(\vec{u}, \vec{v}) = \int_{H(0)} \left(\frac{\partial \Psi_{passive}}{\partial \mathbf{F}} + \mathbf{P}_{active} \right) \nabla \vec{v} dV + \int_{\partial H_1} k\vec{u}\vec{v} dS$$
$$- \int_{\partial H_2} J\sigma_1 \mathbf{F}^{-T} \vec{N} \vec{v} dS - \int_{\partial H_3} J\sigma_2(t) \mathbf{F}^{-T} \vec{N} \vec{v} dS,$$
$$b(\vec{v}, p) = \int_{H(0)} pJ\mathbf{F}^{-T} \nabla \vec{v} dV,$$
$$c(\vec{u}, \vec{w}) = \int_{H(0)} p(J-1) dV,$$

where we have used that $\frac{\partial J}{\partial \mathbf{F}} = J\mathbf{F}^{-T}$.

The mixed finite element formulation can also be described through the equivalent minimization of potential energy with the mixed formulation. The potential energy is described through the functional

$$\Pi(\vec{u}, p) = \int_{H(0)} \Psi(\mathbf{F}(\vec{u})) + p(J-1) dV. \tag{4.28}$$

When using the equivalent minimization of the potential energy, we wish to solve the following problem:

$$\lim_{\epsilon \to 0} \frac{\Pi\left(\vec{u} + \epsilon \vec{v}\right) - \Pi\left(\vec{u}\right)}{\epsilon} = 0, \ \forall \ \vec{v} \ \in \ \hat{V} \quad \text{and} \quad \lim_{\epsilon \to 0} \frac{\Pi\left(\vec{p} + \epsilon \vec{w}\right) - \Pi\left(\vec{p}\right)}{\epsilon} = 0, \ \forall \ \vec{w} \ \in \ L^2(H(0)), \tag{4.29}$$

which is equivalent to solving

$$\lim_{\epsilon \to 0} \int_{H(0)} p \frac{J\left(\vec{u} + \epsilon \vec{v}\right) - J\left(\vec{u}\right)}{\epsilon} + \frac{\Psi\left(\mathbf{F}\left(\vec{u} + \epsilon \vec{v}\right)\right) - \Psi\left(\mathbf{F}\left(\vec{u}\right)\right)}{\epsilon} \, dV, \tag{4.30}$$

and

$$\int_{H(0)} p\left(J - 1\right) \vec{w} dV. \tag{4.31}$$

In summary the problem that we solve is

$$\lim_{\epsilon \to 0} \frac{\Pi\left(\vec{u} + \epsilon \vec{v}\right) - \Pi\left(\vec{u}\right)}{\epsilon} = 0, \tag{4.32}$$

where

$$\Pi\left(\vec{u}\right) = \int_{H(0)} \Psi_{tot}\left(\mathbf{F}\left(\vec{u}\right)\right) dV. \tag{4.33}$$

$$\Psi_{tot} = \Psi_{passive} + \Psi_{incompressible} + \Psi_{active} \tag{4.34}$$

The mixed finite element problem may lead to a phenomena known as locking, in which the material tend to be more stiff than it should. This is particularly a problem with lower order finite elements. It has been shown in [9] that using second order Lagrangian elements for the displacement $\vec{u}$ and first order Lagrangian elements for the hydrostatic pressure p, yields a stable numerical solution. This is also known as Taylor-Hood finite elements.

In this chapter we have presented the mathematical framework needed to create a model for the beating ventricle. The implementation of the mathematical model and the mechanical events throughout the PV-loop is given in the next chapter.

Chapter 5

Implementation

In this chapter we will give a brief introduction to the open-source program that have been used to solve the heart equations. We will also go through the implementation of our model for the heart beat. This will be done by presenting the implementation of the anatomy of the left ventricle, and also the implementation of each phase in the pressure volume loop.

5.1 FEniCS

To solve the partial differential equations for the mechanical behaviour of the heart, the open-source program FEniCS was used. The FEniCS project was initiated in 2003 and was originally a collaboration between the Universty of Chicago and Chalmers university of technology. It is a an open-source program for scientific computing, and has a special focus on solving differential equations by the finite element method. In this section, we will explain the structure and main main components of FEniCS.

The structure of FEniCS is given in Figure 5.1, where the dashed lines indicate data flow and solid lines indicate dependencies.

5.1.1 DOLFIN

DOLFIN is a C++ and Python library that serves as the main user interface of FEniCS. DOLFIN wraps the functionality of other FEniCS components and external software, and handles the communication between these components. In this master thesis, we use the Python interface. DOLFIN is organized as a collection of modules, with each

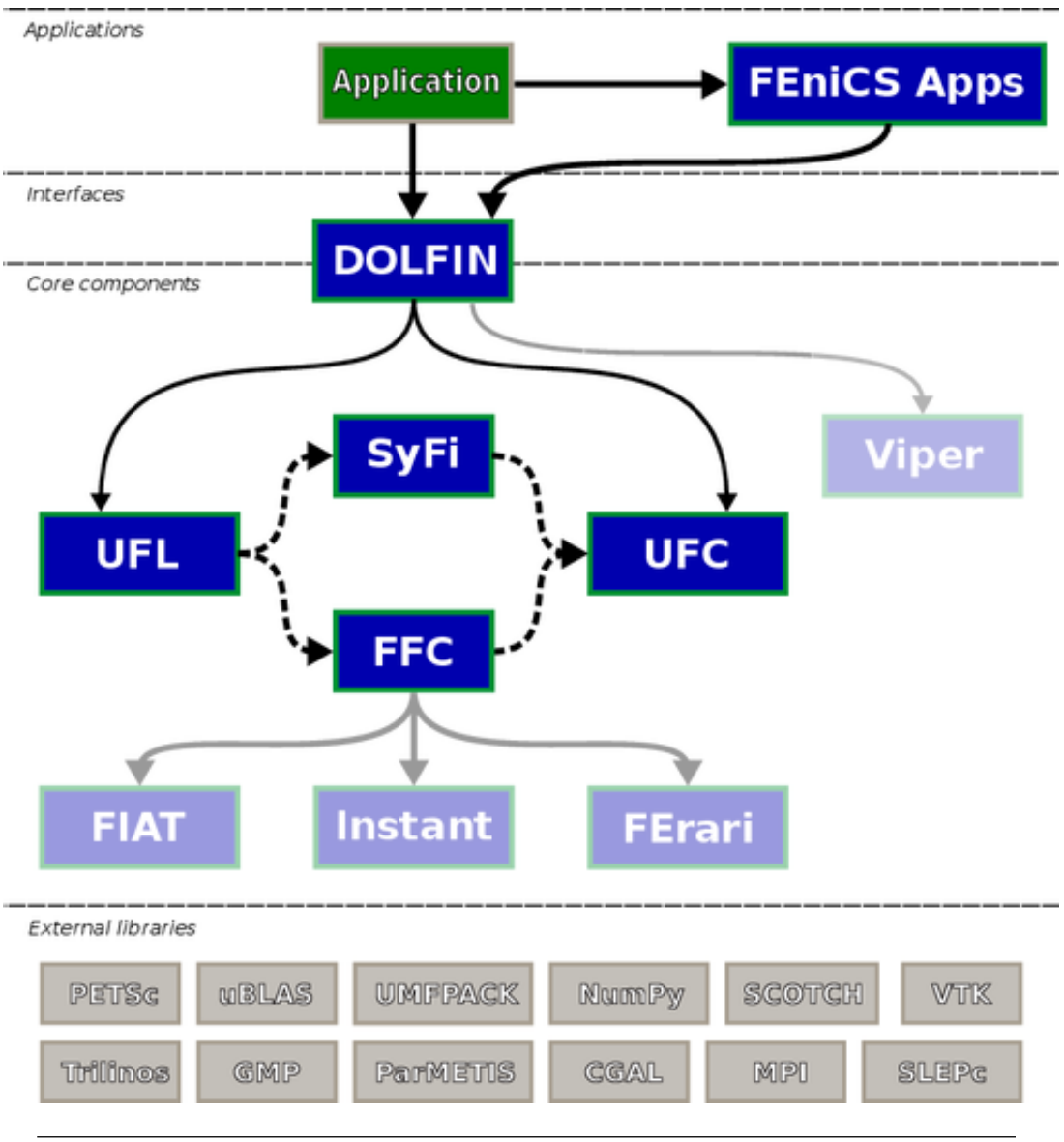


FIGURE 5.1: Structure of the FEniCS project https://fenicsproject.org/about/.

covering a certain area of functionality. Examples of these modules are functions and function spaces, finite elements, linear algebra, meshes and variational forms.

5.1.2 Form compilers translating UFL to UFC

The unified form language (UFL) is one of the core components for the solving of partial differential equations in FEniCS. It defines the language by which the PDE is expressed, and is the input language and front end of the FEniCS form compiler (FFC). SFC generates low level C++ code (UFC) from the high level mathematical description of a variational problem written in UFL. The process is illustrated in Figure 5.2.

FIGURE 5.2: The form compiler translates UFL code to UFC https://fenicsproject.org/about/components.html#about-components-ufc.

5.1.3 Solving the heart equations in DOLFIN

In the following we will present a simple example using DOLFIN to solve the heart equations on a unit cube slab of tissue. The material model we use is the compressible Neo-Hookean model:

$$\Psi = C_1(\bar{I}_1 - 3) + D_1(J-1)^2$$
$$\bar{I}_1 = J^{\frac{2}{3}} \operatorname{tr} \mathbf{C}.$$

A Dirichlet boundary condition is imposed on the boundary at $x = 0$, and a Neumann condition for the boundary at $x = 1$. The remaining boundaries have homogeneous Neumann conditions:

$$\begin{aligned} \nabla_0 \cdot \mathbf{P} = 0 \ \text{ in } \ [0,1] \times [0,1]\,, \\ \vec{u} = 0 \quad \text{on } \ \{0\} \times [0,1] \\ \vec{T} = J\sigma \mathbf{F}^{-T} \vec{N} \ \text{ on } \ \{1\} \times [0,1]\,. \end{aligned}$$

To import all DOLFIN functionalities, this line of code is written in Python.

Python code

```
from dolfin import *
```

Here, we use a build in mesh generation function in DOLFIN which, creates a unit cube mesh with 10×10×10 nodes.

Python code

```
mesh = UnitCubeMesh(10, 10, 10)
```

We now define the vector function space where we will search for our solution. The first argument of $VectorFunctionSpace()$ is $mesh$, which defines the domain of the function space. The second argument "$Lagrange$" defines the element types we are using, whilst

the third argument gives the degree of the elements. With degree 1, we have a node in every corner of the tetrahedral elements. By increasing this argument we increase the degree of the polynomial approximation, and also the computational expensiveness.

Python code

```
V = VectorFunctionSpace(mesh, "Lagrange", 1)
u  = Function(V, name = "Displacement")
du = TrialFunction(V)
v = TestFunction(V)
```

In this part of the code, we define two classes for identifying the Dirichlet and Neumann boundaries on the mesh. A mesh function $boundaries = MeshFunction()$ is defined in order to create a surface measure ds that can distinguish between the different boundaries. We mark the boundaries through the function .mark(boundaries,1) with the value 1 and 2 for the Dirichlet and Neumann part, respectively. This allows us to integrate over the Neumann boundary writing $ds(2)$. Next, the Dirichlet boundary condition is initiated by the functin DirichletBC() which takes three arguments. The V.sub(0) tells which part of the vector function space the boundary condition is valid for. By writing $V.sub(0)$, this access the $x-$component of the vector function. The second argument 0.0, tells us the value at the nodes, and the third argument 1 tells us that this holds for the nodes marked with the value 1.

Python code

```
# Create classes for defining parts of the boundaries
class DirichletBoundary(SubDomain):
    def inside(self, x, on_boundary):
        return near(x[0], 0.0)

class NeumannBoundary(SubDomain):
    def inside(self, x, on_boundary):
        return near(x[0], 1.0)

# Initialize sub-domain instances
DirichletBoundary = DirichletBoundary()
NeumannBoundary = NeumannBoundary()

# Initialize mesh function for boundary domains
boundaries = MeshFunction("size_t", mesh,2)
DirichletBoundary.mark(boundaries, 1)
NeumannBoundary.mark(boundaries, 2)

bcs = DirichletBC(V.sub(0), 0.0, boundaries, 1)
ds = Measure('ds', domain=mesh, subdomain_data=boundaries)
```

Here, we define the kinematics needed for the PDE. We see that the syntacs is close to the mathematical formulation.

Python code

```
# Kinematics
I = Identity(3)
F = variable(I + grad(u))
J = det(F)
C = F.T*F
E = 0.5*(C-I)
N = FacetNormal(mesh)
```

In the following we define the compressible Neo-Hookean material model.

Python code

```
# The compressible Neo-Hookean material model
I_1 = pow(J, -float(2)/3)*tr(C)
psi = 1*(I_1 - Constant(3.0)) + 0.5*(J-1)**2
P = diff(psi, F)
```

Here, we define the variational form and use the built in solve function from DOLFIN to solve non-linear variational form. An illustration of the displacement field on the cube is given in Figure 5.3.

Python code

```
sigma = 0.1
T = -dot(J*sigma*inv(F).T, N)
L = inner(P, grad(v))*dx + inner(T, v)*ds(2)

a = derivative(L, u, du)
solve(L == 0, u, bcs, J=a)
```

5.2 Modelling the anatomy of the left ventricle

The geometry used throughout these experiments is an ellipsoidal surface of revolution of different refinements. This geometry is a severe simplification compared to reality but early models of the heart used such geometries. An illustration of the linear tetrahedral mesh with 690 elements is given in Figure 5.4.

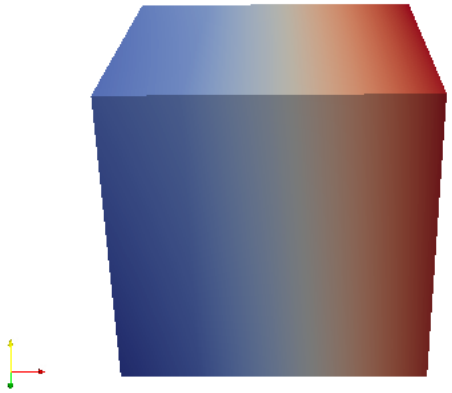


FIGURE 5.3: Plots of the displacement of the unit cube with Dirichlet boundary condition on the left boundary, and Neumann condition on the right boundary.

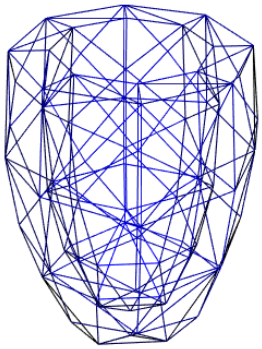


FIGURE 5.4: Linear tetrahedral mesh with 690 elements used to model the left ventricle

5.2.1 Implementing circumferential fiber field

The heart muscle cells are oriented in a complex but elegant manner. They spiral from the endocardium to the epicaridum in a nonlinear way. From the outer to the inner part of the heart wall, the orientation of the muscle fibres change, where the muscle fiber rotate from +50 to +70 degrees from the sub-epicaridal region to 0 degrees in the middle of epicardium to -50 to - 70 degrees with respect to the circumferential direction of the left ventricle. [6]. We implement a simplified fiber field, and orient the fibers along the circumference of the ellipsoid.

We get a Cartesian expression for the circumferential vector by forcing the inner product between the radial $\vec{r} = (x, y)$ vector and the circumferential vector $\vec{c} = (x_c, y_c)$ to be

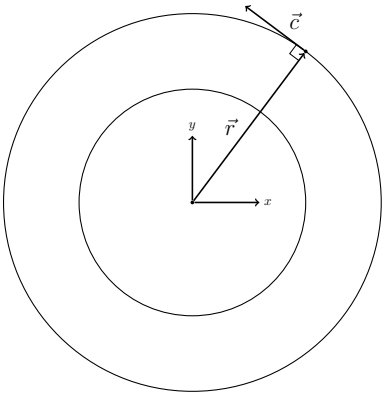


FIGURE 5.5: Illustration of the direction of the circumferential fibers

zero:

$$\vec{r} \cdot \vec{c} = (x, y) \cdot (x_c, y_c) = xx_c + yy_c = 0.$$

We easily see that the vector

$$\vec{c} = (x, -y)$$

satisfy this equation. The circumferential vector is defined in this way at every node of the mesh.

In the implemetation of the fiber field we first initiate the vector function space for the circumferential vectors for the fiber field. A function space for the components of the circumferential vectors is also initiated. Each vector component v_c_x, v_c_y and v_c_z have one degree of freedom.

Python code

```
def Create_circum_Fiber(mesh):
  # Function creates circumferential vector field
  V = VectorFunctionSpace(mesh, "CG", 1)
  F = FunctionSpace(mesh, "CG", 1)
  # Function for x,y,z-coordinates for circumferential vector
  v_c_x = Function(F)
  v_c_y = Function(F)
  v_c_z = Function(F)
  vc = Function(V)
```

Here, $dofs$ is a list that is reshaped such that it is containing the coordinates for the x, y and z components of each nodes in the mesh. In this way, $dofs$ is a correspondence between the node number i and the coordinates of the node.

Python code

```
dm = F.dofmap()
dofs = dm.tabulate_all_coordinates(mesh).reshape((-1,
  mesh.geometry().dim()))
```

The loop in the following code iterates over all nodes and creates the circumferential vector $\vec{c} = (x, -y)$ for each node. For the apex case, a simple vector of unit length is defined.

Python code

```
for i,coord in enumerate(dofs):
  norm = sqrt(coord[1]**2 + coord[2]**2)
  # assigns value for circumferential vector at every node in mesh
      if not near(norm, 0):
          v_c_y.vector()[i] = -coord[2]/norm
          v_c_z.vector()[i] = coord[1]/norm
      else:
        #Apex
          v_c_y.vector()[i] = 1
          v_c_z.vector()[i] = 0
```

Next, we assign the values for the components v_c_x, v_c_y and v_c_z to the vector function v_c. This is done by using the built in function $FunctionAssigner()$.

Python code

```

fax = FunctionAssigner(V.sub(0), F)
fax.assign(vc.split()[0], v_c_x)
fay = FunctionAssigner(V.sub(1), F)
fay.assign(vc.split()[1], v_c_y)
faz = FunctionAssigner(V.sub(2),F)
faz.assign(vc.split()[2], v_c_z)
return vc
```

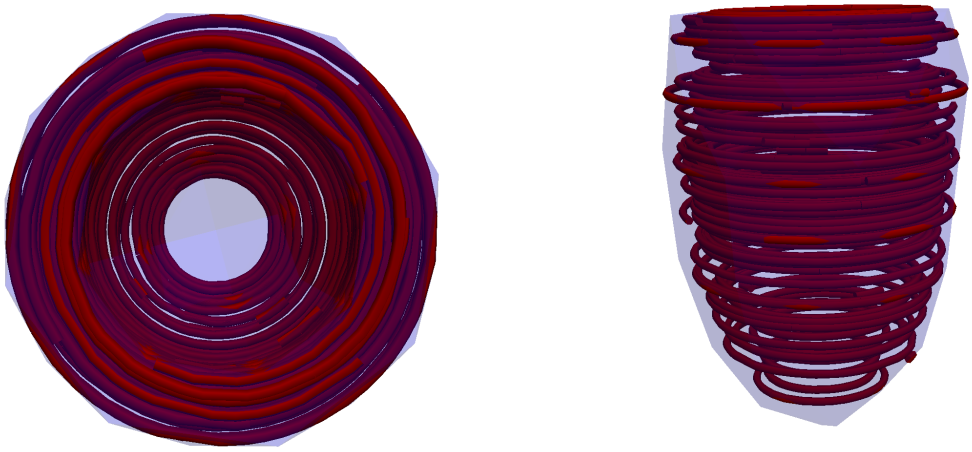
A streamline plot of the circumferential fiber field is given in Figure 5.6.

5.2.2 Implementation of the class heartmodel

In order to structure the code, a class called heartmodel is implemented. A skeleton that includes the headers of the most important functions in the class is given below.

Python code

```
class HeartModel(object):
  def __init__(self,mesh_name):
```



(A) View from above. (B) View from the side.

FIGURE 5.6: Streamline plots of the circumferential fiber field.

```
    self.iters = iters
    self.mesh_name = mesh_name
    self.mesh, self.fibers = self.get_mesh_and_fibers(mesh_name)
    self.solver_parameters = solver_parameters
    self.init_kinematics()
    self.set_boundaries()
    self.make_variational_form()
    self.solver_parameters = solver_parameters

  def init_kinematics(self):
    ...
  def get_mesh_and_fibers(self,mesh_name):
    ...
  def plot_mesh(self):
    ...
  def set_boundaries(self):
    ...
  def make_variational_form(self):
    ...
  def set_sigma(self,sigmaEndo):
    ...
  def solve(self,sigmaEndo):
    ...
  def get_volume(self):
    ...
```

5.2.3 Implementing the active contraction on the left ventricle

We simulate the active force development in the cells trough the intracellular calcium concentration. The function may be described as a function of time which is based on experimental [10]:

$$[Ca^{2+}(t)]_i = [Ca^{2+}]_{i_0} + \left([Ca^{2+}]_{i_{max}} - [Ca^{2+}]_{i_0}\right) \frac{t}{\tau_{C_a}} e^{1-\frac{t}{\tau_{C_a}}}. \tag{5.1}$$

Here, the resting level of the intracellular calcium concentration in the muscle cell is denoted $[Ca^{2+}]_{i_0}$, the maximum concentration is denoted by $[Ca^{2+}]_{i_{max}}$, and $t = \tau_{C_a}$ is the time when the maximum concentration is achieved.

This is a purely phenomenological model, and therefore a severe simplification of the actual events. However, it is reasonable to assume that the active force development depends on the intracellular calcium concentration, and in this case we assume a linear dependency. A plot of the function $[Ca^{2+}(t)]_i$ is given in Figure 5.7.

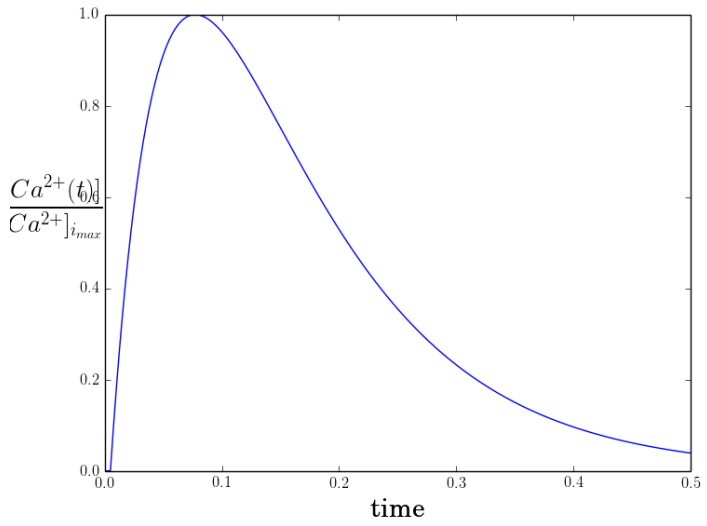


FIGURE 5.7: Scaled function for the intracellular calcium concentration as a function of time.

With the fibers oriented in the circumferential direction, the active stress in the fiber direction can be expressed through the tensor product

$$\mathbf{P}^a = [Ca^{2+}]_i(t) \left(\frac{\vec{c}}{\|\vec{c}\|} \otimes \frac{\vec{c}}{\|\vec{c}\|} \right) = \frac{[Ca^{2+}]_i(t)}{\sqrt{x^2+y^2}} \begin{pmatrix} x^2 & -xy & 0 \\ -xy & y^2 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{5.2}$$

The initializing of the active part of the Piola-Kirchhoff stress tensor is done in the function *init_kinematics*(). We first define the tensor function space for the stress. This initialization gives us the possibility to assign the value of the stress in an external file.

Python code

```
# from the function init_kinematics()
self.W = TensorFunctionSpace(self.mesh,'CG',1)
self.P_active = Function(self.W)
```

The Piola Kirchhoff stress tensor is the assigned the value of (5.2) at each time step t.

Python code

```
# from external file
tensorActive = project(outer(vc,vc),model.W)
model.P_active.assign(tensorActive*ca_transient(t)*0.00001)
```

When using the minimal potential energy formulation, the active contraction can also be implemented through the scalar function

$$\Psi_{active} = [Ca^{2+}]_i(t)]\vec{f_0}^T \mathbf{C}\vec{f_0}. \tag{5.3}$$

As we are dealing with a scalar function, the function space is chosen to be the real space $\mathbb{R}$.

Python code

```
# from the function init_kinematics()
self.sigmaActive = Function(FunctionSpace(self.mesh,"R",0))
self.psi_active = self.sigmaActive*inner(self.fibers, C*self.fibers)
```

In the same way as for the implementation of $\mathbf{P}_{ative}$, the initializing of Ψ_{active} inside the class gives us the opportunity to impose the active force for some time t in an external file.

Python code

```
#from external file
model.sigmaActive.assign(Constant(ca_transient(t)*1.5))
```

Computing the volume of the left ventricle

We calculate the volume of the left ventricular cavity in order to create the PV-loop. The volume of the ventricular cavity is a function of the pressure through the displacement $\vec{u}$.

The displacement depends on the endocardial $\sigma_2(t)$ and epicardial σ_1 pressure through the boundary conditions of the FEM formulation (4.21).

In order to compute the volume function, we use the fact that the surface area of the endocardial wall is known. We also know that the divergence of $\vec{x}$ with respect to the deformed geometry is 3. By noticing this, and applying the divergence theorem, we get

$$V = \int_\Omega \mathrm{d}v = \frac{1}{3}\int_\Omega \nabla \cdot \vec{x}\mathrm{d}v = \frac{1}{3}\int_{\partial\Omega} \vec{x}\vec{n}\mathrm{d}s.$$

We have that $\vec{x} = \vec{X} + \vec{u}$ (3.1), and convert the integral on the deformed geometry

$$\int_{\partial\Omega} \vec{x}\vec{n}\mathrm{d}s = \int_{\partial\Omega} \left(\vec{X} + \vec{u}\right)\vec{n}\mathrm{d}s,$$

to the undeformed geometry by applying Nansons formula (3.18) and obtain

$$V = \int_{\partial\Omega} \left(\vec{X} + \vec{u}\right)\vec{n}\mathrm{d}s = \int_{\partial\Omega_0} \left(\vec{X} + \vec{u}\right) J\mathbf{F}^{-T}\vec{N}dS.$$

The implementation is shown below.

Python code

```
def get_volume(self):
  I = Identity(3)
  F = variable(I + grad(self.u))
  J = det(F)

  N0 = -dot(J*inv(F.T),self.N)
  x = SpatialCoordinate(self.mesh)

  D = dot((x+self.u),N0)*self.ds(self.ENDO)
  vol = (1./3)*assemble(D)

  return vol
```

5.3 Implementation of the phases of the heartbeat - generating the PV-loop

We are now ready to implement the five phases of the heart beat. We start by simulating the passive filling phase of the heart. The two isovolumic phases is modelled equally by

a constant volume constraint using the Newton method. The ejection phase is modelled with a 2-module Windkessel model. The ending time for the phases is chosen such that we get the desired pressure volume development in each phase.

5.3.1 Implementation of the passive filling phase

During the passive filling, the volume and the pressure in the left ventricle increase as blood flows passively into the ventricle after the semilunar valves have closed. To model this, we simply increase the pressure linearly up to a chosen value for the end diastolic pressure.

Python code

```
def passive_filling(model):

  sigmaEndo = 0
  sigmaEpi = 0
  dsigmaEndo = 0.1
  while (sigmaEndo <= 0.1 ):

    v0 = model.solve(sigmaEndo,sigmaEpi)
    sigmaEndo =  sigmaEndo + dsigmaEndo

  sigmaEndo = sigmaEndo - dsigmaEndo
  u_temp = model.u
  vstart = v0

  return u_temp, sigmaEndo, vstart
```

5.3.2 Implementation of the isovolumic contraction phase

During the isovolumic contraction, the volume of the ventricular cavity remains constant while the pressure in the ventricle increases. To fulfill the isovolumic constraint and simulate this part of the heart cycle, Newton's method has been applied.

We define the function

$$f = V(p) - V_{EDV}, \tag{5.4}$$

$V(p)$ is the volume of the left ventricle which depend on the endocardial pressure p, and V_{EDV} is the chosen end diastolic volume. We want to solve the problem: find p such that $f(p) = 0$.

The Newton step becomes

$$p_n = p_{n-1} - \frac{f(p_{n-1})}{f\prime(p_{n-1})}. \tag{5.5}$$

We use a finite difference approximation for the derivative

$$f\prime(p) = \frac{df}{dV}\frac{dV}{dp} \approx \frac{V(p+\epsilon) - V(p)}{\epsilon}. \tag{5.6}$$

For each time step, we impose the time dependent active force under the constraint that the volume must be constant. Newton's method is then used to find the pressure that fulfills this constraint.

We first import the calcium function for the force development (5.1). The function f and the derivative f_{prime} that are needed for the newton step are implemented.

Python code

```
from Force import ca_transient

def f(p, vstart,model):
  volNew = model.solve(p)
  return  volNew - vstart

def f_prime(p,model):
  epsilon = 0.0001
  volEps = model.solve(p+epsilon)
  volNew = model.solve(p)
  return (volEps- volNew)/epsilon
```

The functions f and f_{prime} are used in the implementation of the Newton step:

Python code

```
def Newton_step(p0,model,vstart):
  j = 0
  tol = 0.001
  maxIterations = 4
  y = 1
  while (j < maxIterations) and (abs(y)> tol):
    y = f(p0,vstart,model)
    y_prime = f_prime(p0, model)
    p1 = p0 - y/y_prime
    j = j + 1
    p0 = p1
  return p1
```

The complete code for the isovolumic contraction is shown below.

Python code

```
def isovolumic(model,sigmaEndo,vstart):
  vc = model.fibers
  tensorActive = project(outer(vc,vc),model.W)

  epsilon = 0.0001
  t = 0.006
  dt = 0.01
  tol = 0.000000001

  maxIterations = 20
  sigmaEpi = 0

  while( (t <  0.035) ):
    try:
      model.sigmaActive.assign(Constant(ca_transient(t)*1.5))
      u_temp = model.u.copy(True)
      vAfterActiveStress = model.solve(sigmaEndo, sigmaEpi)
      u_temp = model.u.copy(True)
      volNew = vAfterActiveStress

      sigmaEndo = Newton_step(sigmaEndo,model,vstart)

      volNew = model.solve(sigmaEndo,sigmaEpi)

      t = t + dt

      if dt < 0.01:
        dt = dt*1.5

    except RuntimeError:
      dt = dt/2
      t = t - dt
      model.u.assign(u_temp)

      if dt < DOLFIN_EPS:
        raise

  return sigmaEndo, volNew, t
```

5.3.3 Implementation of the ejection phase - the Windkessel model

There exist Windkessel models of different complexity. In this master thesis the 2-module Windkessel model is applied to model the ejection phase of the cardiac cycle. The Windkessel model consists of ordinary differential equations which relates the dynamics

of aortic pressure and blood flow to various parameters such as arterial compliance, resistance to blood flow and the inertia of blood.

The cardiovascular system can be modelled analogous to an electrical circuit, such that electrical charge and currents represents blood volume and flow rates [11].

The arterial compliance comes from the blood vessels ability to accumulate and release blood due to elastic deformations, this is modelled analogously to capacitors in electrical circuits [12]. The resistance in the blood as it passes from the aorta into the narrower arterioles, referred to as peripheral resistance, is modelled analogously to a resistor. These resistors in the cardiovascular system is dependent on blood viscosity and the diameter of the vessels. Using this analogy, a particular vessel can be described by a combination of resistors and capacitors.

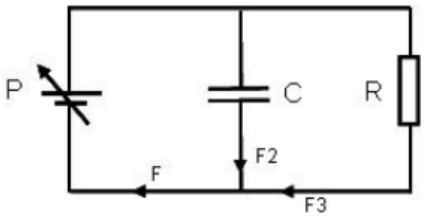


FIGURE 5.8: The 2-module windkessel model

Figure 5.8 shows the 2-module Windkessel model consisting of a capacitor C corresponding to the arterial compliance and a resistor R corresponding to the resistance in the blood. P represents the aortic pressure and F represents the blood flow rate in the aorta.

The pressure difference P is in general non-linear. As we are modelling a laminar flow the pressure difference can be assumed to be linear [13]. If we consider a cylindrical blood vessel and look at the pressure difference between the two ends of the cylinder, the resistance R, which is the proportionality constant between the pressure difference P and the flow F can be written as

$$R = \frac{P}{F}. \tag{5.7}$$

This is analogous to Ohms law for electrical circuits

$$R = \frac{V}{I}, \tag{5.8}$$

with, V representing the potential difference and I the current.

The compliance of the aortic vessel results in a net storage of blood which can be modelled analogous to the capacitor in an electrical circuit. The net flow through a blood vessel is denoted by $F = F_{in} - F_{out}$. We assume a linear relation between the rate of change of pressure in the vessel and the net flow

$$F = C_c \frac{dP}{dt}. \tag{5.9}$$

This is analogous to the relationship between the derivative of the potential with respect to time in electrical circuits

$$I = C_e \frac{dV}{dt}. \tag{5.10}$$

By combining the equations for the different components of the circuit

$$F = F_2 + F_3 \tag{5.11}$$

$$P = F_3 R \tag{5.12}$$

$$C \frac{\partial P}{\partial t} = F_2, \tag{5.13}$$

and applying Kirchhoff's law for currents we get the differential equation

$$F = \frac{P}{R} + C \frac{\partial P}{\partial t}. \tag{5.14}$$

Here F represents the blood flow out of the left ventricle, and hence represents a negative change of volume. Thus, we can write the Windkessel model as

$$\frac{-\partial V}{\partial t} = \frac{P}{R} + C \frac{\partial P}{\partial t}. \tag{5.15}$$

Hence, we can define the residual f as

$$f = \frac{\partial V}{\partial t} + \frac{P}{R} + C \frac{\partial P}{\partial t} = 0. \tag{5.16}$$

We use a first order finite difference approximation for the volume and pressure change:

$$f = \frac{V_n - V_{n-1}}{t_n - t_{n-1}} + \frac{p_n - p_{n-1}}{R} + C \frac{p_n - p_{n-1}}{t_n - t_{n-1}}, \tag{5.17}$$

and by factorizing with respect to the pressure we end up with the expression

$$f = V_n - V_{n-1} + \left(C + \frac{\Delta t}{R}\right) p_n - C p_{n-1}. \tag{5.18}$$

The derivative of the volume with respect to the pressure is approximated by the finite difference approximation

$$\frac{dV}{dp}\frac{dp}{dt} \approx \frac{V_n - V_{n-1}}{\epsilon},$$

hence

$$\frac{\partial f}{\partial p} = \frac{V_n - V_{n-1}}{\epsilon} + \left(C + \frac{\Delta t}{R}\right). \tag{5.19}$$

The volume and pressure of the next step is then found by the Newtons method, where the pressure and volume of the next step is given by

$$p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})},$$

with $V_n = V(p_n)$. The function f and f_prime are needed for the Newton step to be implemented.

Below we show the implementation of the Windkessel model and the ejection phase.

Python code

```
from dolfin import *
from Force import ca_transient
from circumFiber import *
C = 40
R =  0.003

def f(p, vstart,pstart,model,dt):
  volNew = model.solve(p)
  return  volNew + (C + dt/R)*p - vstart  - C*pstart

def f_prime(p,model,dt):
  epsilon = 0.0001
  volEps = model.solve(p+epsilon)
  volNew = model.solve(p)
  return (volEps - volNew)/epsilon + C + dt/R
```

The Newton step is implemented in the following.

Python code

```
def Newton_step(p0,model,vstart,dt):
  j = 0
  tol = 0.1
  maxIterations = 20
  y = 1
  pstart = p0
  while (j < maxIterations) and (abs(y)> tol):
    y = f(p0,vstart,pstart,model,dt)
    y_prime = f_prime(p0, model,dt)
    p1 = p0 - y/y_prime
    j = j + 1
    p0 = p1
  return p1
```

The ejection phase is implemented as a separate function in the same way as the isovolumic contraction phase.

Python code

```
def ejection(model,sigmaEndo,Vold,t):

  tol = 0.000000001
  dt = 0.1
  epsilon =0.0001
  maxIterations = 50
  sigmaEpi = 0
  j = 0
  vol_underformed = model.get_volume_undeformed()
  volNew = 1000
  u_temp = model.u.copy(True)

  while t < 0.155 and volNew > vol_underformed:
    try:

      t = t + dt
      model.sigmaActive.assign(Constant(ca_transient(t)*1.5)
      vAfterActiveStress = model.solve(sigmaEndo, sigmaEpi)
      u_temp = model.u.copy(True)
      volNew = vAfterActiveStress
      v_after_act = volNew
      sigmaEndoOld = sigmaEndo
      sigmaEndo = Newton_step(sigmaEndo,model,Vold,dt)
      volNew = model.solve(sigmaEndo,sigmaEpi)
      Vold = volNew
```

```
27
    except RuntimeError as ex:
29    print ex

31    dt = dt/2
      t = t - dt
33    model.u.assign(u_temp)

35  return sigmaEndo,volNew, t
```

5.3.4 Implementation of the isovolumic relaxation phase

The implementation of the isovolumic relaxation phase is equal to the isovolumic contraction phase. We solve for the function

$$f = V(p) - V_{ESV} = 0.$$

Here V_{ESV} represent the end systolic volume, which is the volume after end of ejection phase.

Python code

```
1 def
    isovolumic_relaxation(model,pListPlot,vListPlot,sigmaEndo,vstart,t):

3   epsilon = 0.0001
    dt = 0.008
5   tol = 0.000000001
    maxIterations = 20
7
    while( (t <  0.5) ):
9
      try:
11
        model.sigmaActive.assign(Constant(ca_transient(t)*1.5))
13      vAfterActiveStress = model.solve(sigmaEndo)


15
        volNew = vAfterActiveStress
17      sigmaEndo = Newton_step(sigmaEndo,model,vstart)
        volNew = model.solve(sigmaEndo)
19
        t = t + dt
21      if dt < 0.05:
          dt = dt*1.5
```

```
23
    except RuntimeError:
25    dt = dt/2
      t = t - dt
27
      if dt < DOLFIN_EPS:
29      raise

31  return sigmaEndo, volNew, t
```

In this chapter we have presented the open source program FEniCS. We have also described the implementation of our model for the left ventricle. In the next chapter we will present the results from the verification and time optimization of our model.

Chapter 6

Numerical results

In this chapter we present the results. The main output will be the CPU-time for the simulation of the pressure and volume loop. The PV-loop was modelled both for the compressible and incompressible Neo-Hookean material model. The hyperelasticity of the material models will be verified through measuring strain development in the tissue. In order to verify the model we will, for small deformations, compare it to analytical solutions from linear theory. The model will also be verified by the method of manufactured solutions [14], where our model is compared with an analytic solution. The model is then time optimized. This is done through optimizing the parameters for the form compilers. Next, the linear solvers in FEniCS are compared with respect to both CPU-time and number of Newton iterations for both the incompressible and compressible model. The material models are tested for two different mesh refinements. Lastly, the code is run in parallel on the two mesh refinements for the different linear solvers.

The computer used throughout these experiments is a MacBook Pro, with 2.2 GHz intel Core i7 processor and with 16 GB RAM. The visualization is done using the program ParaView [15], the vtk library [16] and matplotlib [17].

6.1 Verification of hyperelasticity

In the next experiment we impose the active contraction force in the fiber direction on a left ventricle geometry. The pressure on the endocardium and epicardium from the boundary conditions is set to zero. This is not physical but the aim of the experiment is to simulate the contraction and relaxation of the myocardium without taking the pressure development throughout the cardiac cycle into account. This experiment is done both for the incompressible and compressible Neo-Hookean model.

Figure 6.1 and Figure 6.3 shows how the ventricle contracts and relaxes in the direction of the circumferential fiber field for the incompressible and compressible Neo-Hookean model, respectively. A plot of the strain development in time and in terms of the active force development for the incompressible and compressible Neo-Hookean model is given in Figure 6.2 and Figure 6.4. The strain measure used is the fiber strain λ from (3.7). With the value of one, for the fiber strain, the fibers are at rest, with the value of 0.5, the fibers have contracted to 50% of its undeformed length. We see how the strain increases and decreases in response to the active force development. When the graphs are marked with red color these figures shows the value of the strain as the fibers are contracting, while the blue lines shows the strain value when the fibers are relaxing back to their original shape. The fact that the strain follows the same path, independent of the asymmetry of the active force function, illustrates the path independence of hyperelastic material.

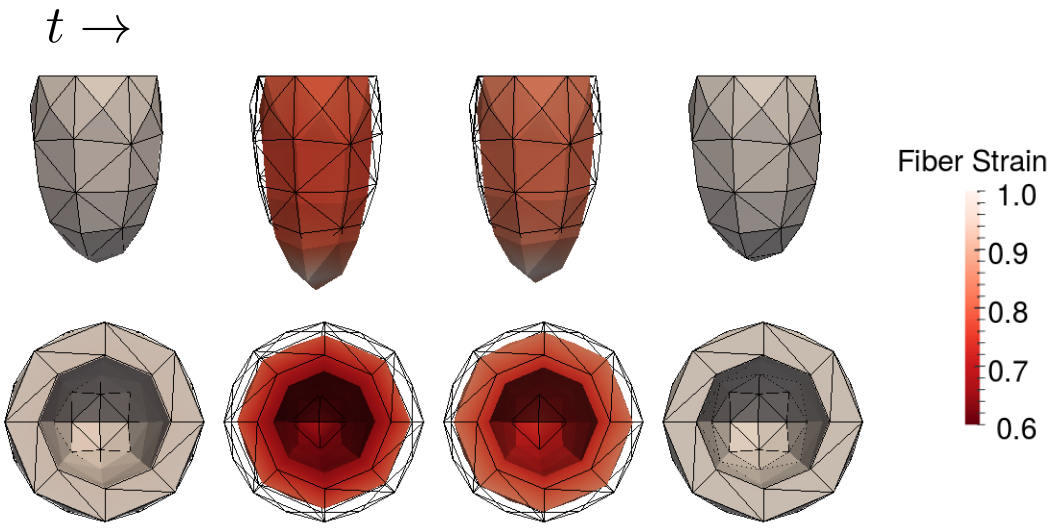
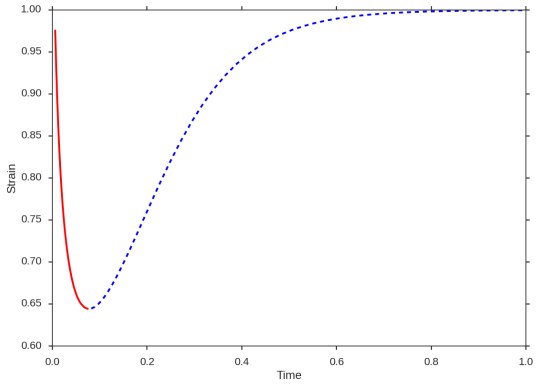


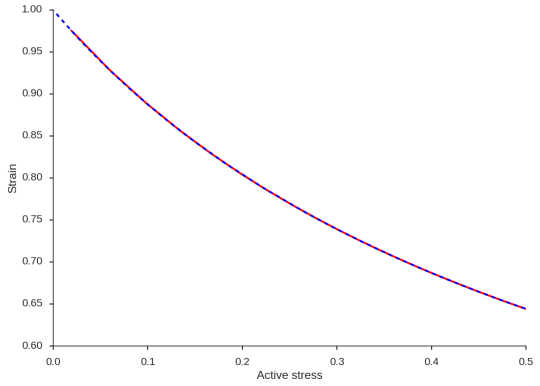
FIGURE 6.1: The displacement and fiber strain development of the left ventricle with circumferential fibers for different time steps during the active contraction induced by 5.1.

6.2 Verification of model

We will now verify our model through three different test cases. In the first test case we will compare our model to a very simple analytic test case. In the second test case we expect our model to behave like a linear elastic material for small deformations. In the last test case we test our solver against a non-linear analytic solution through the method of manufactured solutions.



(A) Strain/stretch λ (3.7) as function of time



(B) Strain/stretch λ (3.7) as function of stress σ_a

FIGURE 6.2: Plots of the strain development in the left ventricle during the active contraction and relaxation with incompressible Neo-Hookean Model.

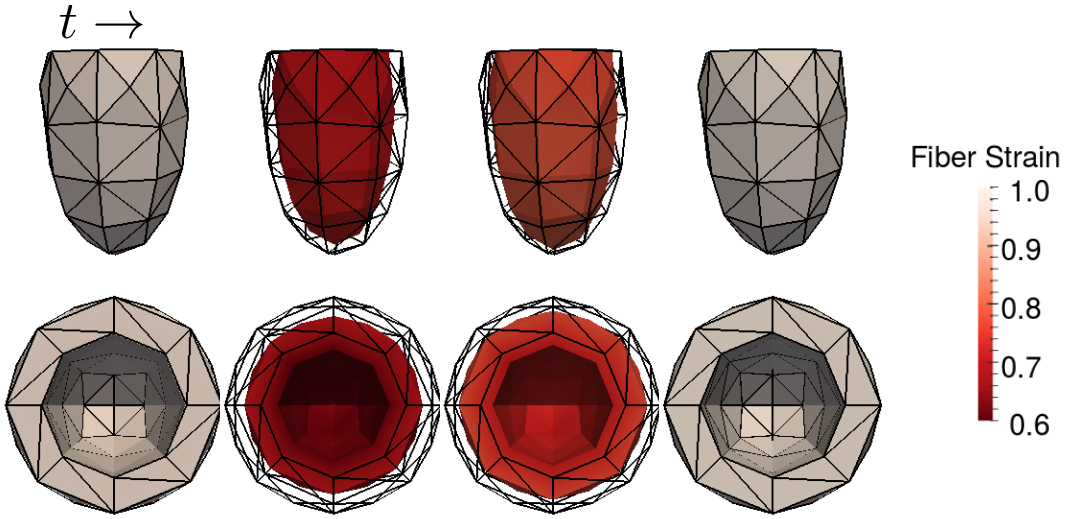
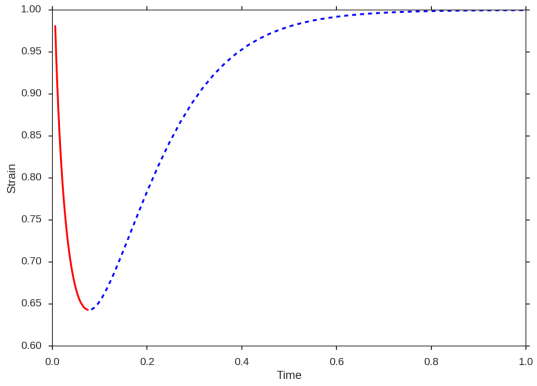


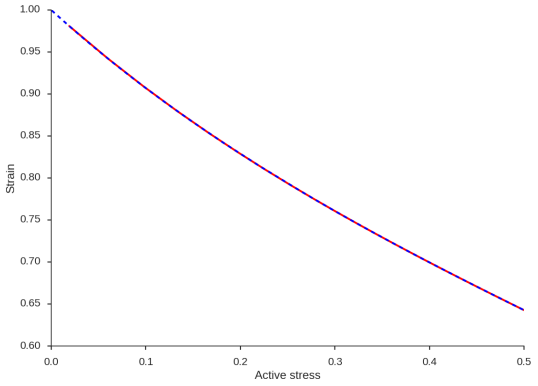
FIGURE 6.3: The displacement and fiber strain development of the left ventricle with circumferential fibers for different time steps during the active contraction induced by 5.1.

Test case 1

To verify the model, we solve a simple analytic test case where we impose Dirichlet conditions for $x = 0$ and $x = 1$ with $u = (0, 0, 0)$ and $u = (0.2, 0, 0)$, respectively. There is only elastic displacement due the Dirichlet condition on $x = 1$, and there is no active force contribution. For this case, we know that the $x-$component of the displacement field $\vec{u}$ will be a linear function in x. The plot of the x-component of the displacement field is given in Figure 6.5. The plot shows that we get the expected linear function. In this test case, both the strain and stress are constants on the domain, which is severe simplifications, such that this test is not sufficient for verification.



(A) Strain/stretch λ (3.7) as function of time



(B) Strain/stretch λ (3.7) as function of stress σ_a

FIGURE 6.4: Plots of the strain development in the left ventricle during the active contraction and relaxation with compressible Neo-Hookean Model.

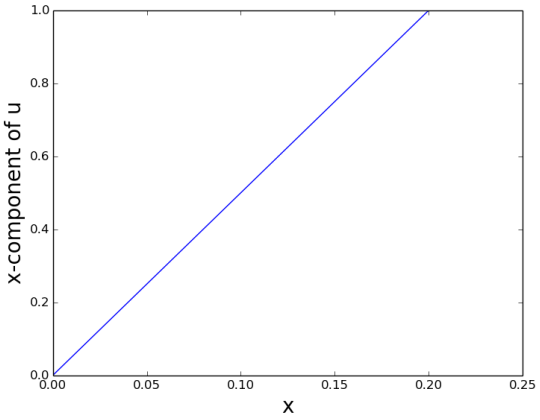


FIGURE 6.5: Plot of x-component of the displacement field against x.

Test case 2

In the second test case we impose a Dirichlet condition for the boundary $x = 0$. For the boundary on $x = 1$ we impose a Neumann condition with a pressure σ_{xx} in the $x-$direction. For the remaining boundary we impose homogeneous Neumann conditions. The material model used in this test case is

$$\Psi = \frac{\lambda}{2}\left(tr(E)\right)^2 + \mu\,\mathrm{tr}(E^2). \tag{6.1}$$

The reason for this choice of material model is that for small deformations, 6.1 behaves as a linear elastic material. We wish to compare the solution for our model with analytical expressions for the components of the small deformations strain tensor ϵ.

For linear elastic materials the analytic expressions for the components ϵ is

$$\epsilon_{xx} = \frac{1}{Y}\sigma_{xx} \tag{6.2}$$

$$\epsilon_{yy} = \epsilon_{zz} = \frac{-\nu}{Y}\sigma_{xx} \tag{6.3}$$

where

$$Y = \frac{\mu(3\lambda + 2\mu)}{(\lambda + \mu)}, \tag{6.4}$$

and

$$\nu = \frac{\lambda}{2(\lambda + \mu)}. \tag{6.5}$$

For small deformations, ϵ and the Green-Lagrange stain tensor $\mathbf{E}$ (3.9) are approximately equal. In Figure 6.6 and 6.7, a plot of the analytical expression for the components of the small deformations strain tensor, together with the numerical solutions of the components of the Green-Lagrange strain tensor from our model is given. We can see that the solution from our model coincides with the linear analytic solution for small values of the stress. As we increase the value of the stress, our model deviates from the linear model as expected.

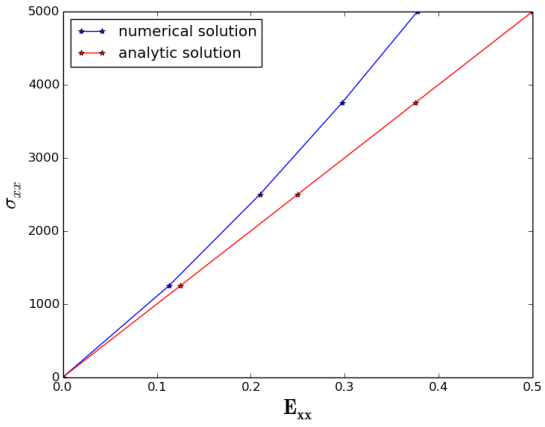
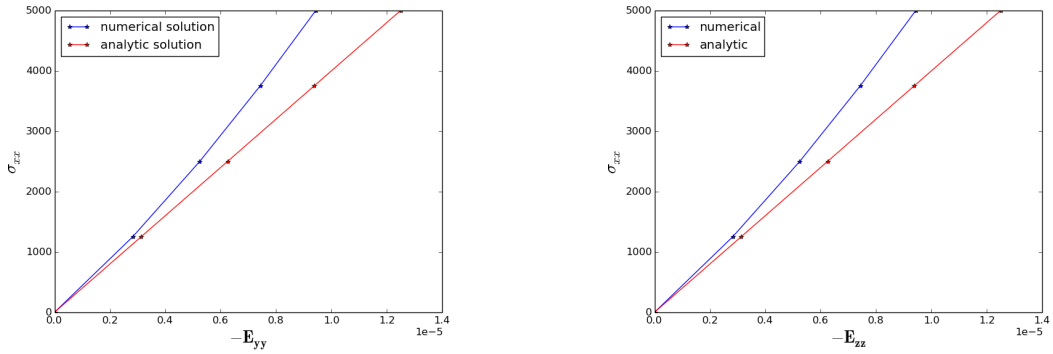


FIGURE 6.6: Plot of the x-component of the $\mathbf{E}$ against σ_{xx} together with analytic solution for 6.4.

Test case 3

In this section we test our model against a known manufactured analytic solution. This is called the method of manufactured solutions.

In this test case we impose an active force on a unit square geometry. The fiber field is set in the direction of the $y-$axis, such that $\vec{f_0} = (0, 1)^T$. We use the incompressible



(A) (B)

FIGURE 6.7: Plot of the $-\mathbf{E}_{\mathbf{yy}}$ and $-\mathbf{E}_{\mathbf{zz}}$against σ_{xx} together with analytic solution $-\nu$ (6.5)

Neo-Hookean model. Such that our strain energy function is

$$\Psi_{passive} = C_1 (I_1 - 3) - p (J - 1).$$

We have that the analytic expressions for the displacement and deformation gradient, which satisfies the incompressibility constraint, are given by:

$$\vec{u} = \begin{pmatrix} \frac{\alpha y^2}{2} \\ 0 \end{pmatrix}, \mathbf{F} = \begin{pmatrix} 1 & \alpha y \\ 0 & 1 \end{pmatrix}.$$

The Piola-Kirchhoff stress tensor with active contribution then becomes

$$\mathbf{P} = 2C_1\mathbf{F} + \sigma_a \left([Ca^{2+}]\right) \mathbf{F}\vec{f}_0\vec{f}_0^{\,T} - pJ\mathbf{F}^{-T}. \tag{6.6}$$

Here, we have used that for the incompressible Neo-Hookean model

$$\frac{\partial \Psi_{passive}}{\partial \mathbf{F}} = \frac{\partial \Psi_{passive}}{\partial I_1}\frac{\partial I_1}{\partial \mathbf{F}} = 2C_1\mathbf{F}.$$

We have also used without proof that

$$\frac{\partial \det(\mathbf{F})}{\partial \mathbf{F}} = \det(\mathbf{F})\,\mathbf{F}^{-T} = J\mathbf{F}^{-T},$$

this holds for all invertible second order tensors [2].

We have that the analytic expressions from the matrix calculations in the expression for (6.6) becomes

$$\mathbf{F}\vec{f}_0\vec{f}_0{}^T = \begin{pmatrix} 1 & \alpha y \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & \alpha y \\ 0 & 1 \end{pmatrix},$$

and

$$\mathbf{F}^{-T} = \begin{pmatrix} 1 & 0 \\ -\alpha y & 1 \end{pmatrix}.$$

The expression for the Piola-Kirchhoff stress tensor now becomes

$$\mathbf{P} = 2C_1\begin{pmatrix} 1 & \alpha y \\ 0 & 1 \end{pmatrix} + \sigma_a\left([Ca^{2+}]\right)\begin{pmatrix} 0 & \alpha y \\ 0 & 1 \end{pmatrix} - p\begin{pmatrix} 1 & 0 \\ -\alpha y & 1 \end{pmatrix}.$$

We are now ready to find the analytic solution to this problem

$$\nabla\cdot\mathbf{P} = .\begin{pmatrix} \frac{\partial P_{11}}{\partial x} + \frac{\partial P_{12}}{\partial y} \\ \frac{\partial P_{21}}{\partial x} + \frac{\partial P_{22}}{\partial y} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

The expression for the components of the stress tensor becomes

$$\begin{aligned} P_{11} = 2C_1 + pJ, P_{12} = 2C_1\alpha y + \sigma_a\left([Ca^{2+}]\right)\alpha y \\ P_{21} = -p\alpha y J, P_{22} = 2C_1 + \sigma_a\left([Ca^{2+}]\right) + pJ \end{aligned}$$

We now calculate the divergence of this tensor, and end up with two separable ordinary differential equations

$$\nabla\cdot\mathbf{P} = \begin{pmatrix} -J\frac{\partial p}{\partial x} + 2C_1\alpha + \sigma_a\left([Ca^{2+}]\right)\alpha \\ -\frac{\partial p}{\partial x}\alpha y J + J\frac{\partial p}{\partial y} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

The first differential equation gives us the function

$$p(x) = J^{-1}\alpha\left(2C_1 + \sigma_a\left([Ca^{2+}]\right)\right)x + K(y),$$

here, $K(y)$ is some function of y, this expression is found from the second differential equation. The total analytic expression for the Lagrange multiplier becomes:

$$p(x,y) = J^{-1}\alpha\left(2C_1 + \sigma_a\left([Ca^{2+}]\right)\right)x + \frac{\alpha^2}{2}\left(2C_1 + \sigma_a\left([Ca^{2+}]\right)\right)y^2 + K.$$

A plot of the analytic expressions for $\vec{u}$ and p together with a convergence plot through the relative error for $\|\vec{u} - \vec{u_h}\|_{H^1}$, the pressure field $\|p - p_h\|_{L^2}$ and the incompressibility constraint $\|J - 1\|_{L^2}$ is given in Figure 6.8 and Figure 6.9. We can see the order for the convergence is quadratic for the pressure field and with a exponent of 2.6 for the displacement field. The values of the parameters was set to: $\alpha = 0.6$, $C_1 = 0.1925$ and $\sigma_a\left([Ca^{2+}]\right) = 0.9$. The analytic solutions was first derived by Rossi in [18]. The implementation of this test case was based on a previous implementation [19].

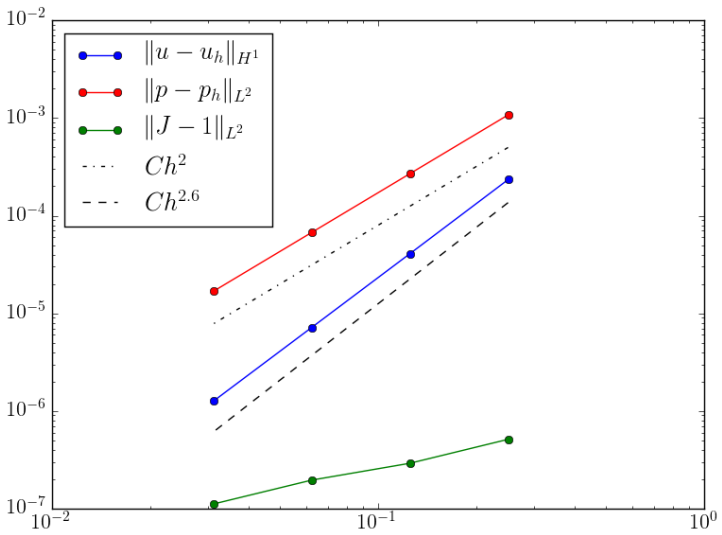
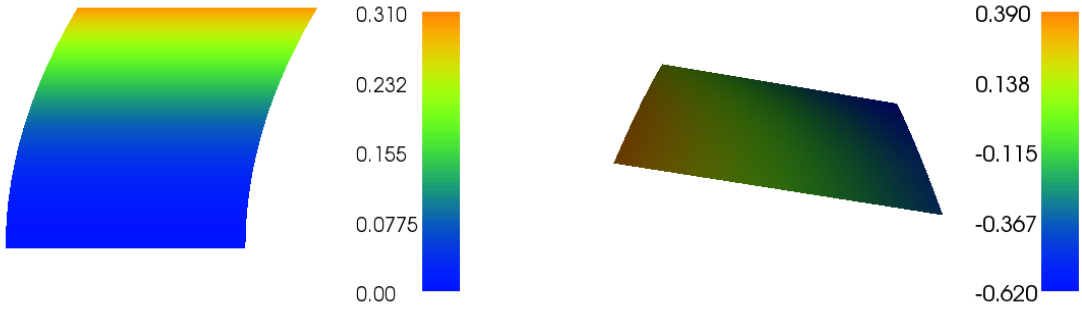


FIGURE 6.8: Plot of the convergence of the method through the relative error of the displacement field the pressure field and the compressibility condition.



(A) (B)

FIGURE 6.9: Plot of the analytic solutions for the displacement field $\vec{u}$ the pressure field p.

6.3 The PV-loop

We will now comment on the process of developing the PV-loop. We will go through each phase of the cycle of the left ventricle. The material parameter of the incompressible Neo-Hookean model

$$\Psi = C_1 (I_1 - 3) - p (J - 1).$$

was set to $C_1 = 0.45$ kPa. For the compressible model

$$\Psi = C_1 \left(\bar{I}_3 - 3\right) - D_1 (J-1)^2,$$
$$\bar{I}_3 = J^{-\frac{2}{3}} \operatorname{tr}(\mathbf{C}),$$

the parameters was set to $C_1 = 1.0$ kPa and $D_1 = 0.5$ kPa.

6.3.1 Passive filling

During the simulation of the passive filling phase we want the volume to double with respect to the undeformed geometry. The undeformed volume is of 134.5 mL. We also want the the pressure to be at a realistic end diastolic level of 1000 kPa. The stability of the rest of the loop is strongly dependent on the volume of the ventricle after the passive filling phase is completed, so that in order for the method to converge, the volume can not be arbitrary large after the passive filling. The incompressible model the loop is more stable when it comes to increasing the volume in the passive filling phase but the volume is lower than a realistic end diastolic volume. The volume reaches 161.5 mL at end diastolic volume, which is an 20 % increase of volume. For the compressible model has a

end diastolic volume of 155.04 mL, which is a 15 % increase of the undeformed volume. The material parameters are tuned manually to get a volume increase and a material stiffness that can go through the isovolumic contraction phase without collapsing.

6.3.2 Isovolumic contraction

After finishing the passive filling phase, the left ventricle goes into its next phase, the isovolumic contraction. The heart valves are closed, and the heart is contracting without volume change, such that the pressure increases rapidly. The goal of this phase is to get a pressure increase of minimum seven times the end diastolic pressure. The isovolumic contraction phase is the most unstable part of the PV-loop. The bigger the volume of the heart is after the passive filling, the more unstable the isovolumic contraction phase is. The modelling of this stage is also very unstable regarding variation of the material parameters and time step increase. If the tissue is too soft, the pressure will not increase as desired, and if the tissue is too hard the method will diverge. To achieve convergence of the method the material parameters and time steps are tuned manually such that we aim for a time efficient and realistic pressure and volume development in each phase. For the incompressible model the pressure is increased from 1000 kpa to 7450 kPa after the isovolumic contraction phase is over. The compressible model reaches the level of 8140 kPa after the isovolumic contraction phase is ended. The pressure rise after this phase is sufficient for both models.

6.3.3 Ejection

During the ejection phase of the left ventricle the blood is ejected from the left ventricle into the aorta. This process is modelled by a 2-module-Windkessel model. Which is a realistic differential equation for modelling the pressure and volume development when the blood is ejected from the left ventricle into the aorta. The Windkessel model has two parameters: R, which describes the blood vessels resistance depending on the diameter of the blood vessels and the viscosity of the blood, and the paramter C, describing the capacitance of the vessels, which is the ability to accumulate and release blood due to elastic displacement. These parameters are tuned in order to get both convergence and a realistic pressure volume development. For the incompressible Neo-Hookean model the parameters are set to $C = 40, R = 0.0015$, whilst for the compressible Neo-Hookean model the paramters was set to $C = 40, R = 0.002$. When the volume of the heart has reached its original undeformed size the ejection phase is finished.

6.3.4 Isovolumic relaxation

The last phase of the cycle is the isovolumic relaxation phase. This is implemented in the same way as the isovolumic contraction phase; by a constant volume constraint and using the Newtons method to find the pressure in the next time step. The stability of the isovolumic relaxation phase is much higher than the isovolumic contraction phase. The reason for this is that the steepness of the active force function is much lower than for the contraction phase for this time interval. This phase is ended when the pressure reaches its end systolic value.

6.3.5 Comparing results for incompressible and compressible material model

We created a model both for the incompressible and compressible Neo-Hookean model. The PV-loop for the incompressible Neo-Hookean model is given in Figure 6.10, and the loop for the compressible Neo-Hookean model is given in Figure 6.11.

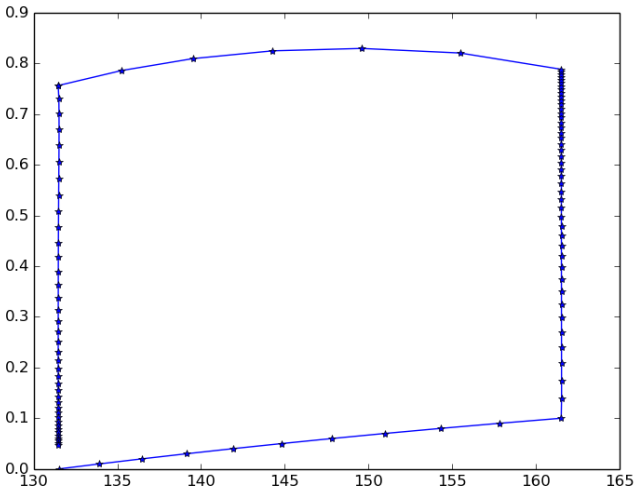


FIGURE 6.10: Pressure Volume loop for incompressible Neo-Hookean model

The displacement and the strain development throughout the PV-loop of the left ventricle for the incompressible and compressible Neo-Hookean model is given in Figure 6.12 and Figure 6.13, respectively. The figures contain pictures seen both from above and from the side of the ventricle, and include pictures for five different snapshots of the PV-loop. The first snapshot is when the mitral valve is opening (MVO) and blood starts flowing passively into the left ventricle, this is the start of the passive filling phase. The second snapshot is when the mitral valve is closing (MVC) and the ventricle

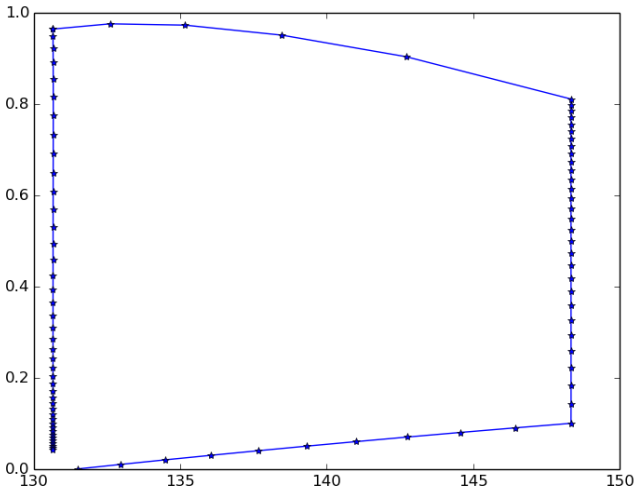


FIGURE 6.11: Pressure Volume loop for compressible Neo-Hookean model

starts to contract isovolumic. The third picture is when the aortic valve is opening (AVO) and the ejection of the blood into the aorta is starting, this is the end of the isovolumic contraction phase and the beginning of the ejection phase. The forth picture is when the aortic valve is closing (AVC), the ejection phase is ending and the isovolumic relaxation phase is starting. The last picture illustrates how the loop repeats itself after the isovolumic relaxation phase is over. The plot illustrates how the ventricle is contracting in the direction of the fiber field. We can see for both models, that the strain development is highest in the start and at the end of the ejection phase, when the left ventricle is contracting and ejecting blood into the aorta. For the incompressible model, we can see, in snapshot at AVO, that the myocardial wall has gotten thinner as the ventricle have contracted isovolumic. We can also observe that the ventricle is getting stretched more than for the compressible model. The strain development during the isovolumic contraction is more regularly distributed throughout the entire ventricle for the incompressible model. For the compressible model the strain is highest around apex and gradually fades upwards at AVO. At the end of the ejection phase both models show an evenly distributed strain throughout the ventricle.

The CPU-time and the number of newton iterations for the two material models are given in Table 6.1 and Table 6.2. We can see that the compressible Neo-Hookean model is over six times as fast as the incompressible model. This is as expected as the incompressible model solves a system both for the displacement $\vec{u}$ and for the pressure p. We evaluate how realistic the models are based on the pressure and volume development throughout the loop. Based on this criterion the two models are almost equally realistic. Although, the incompressible Neo-Hookean has a better volume increase during the passive filling

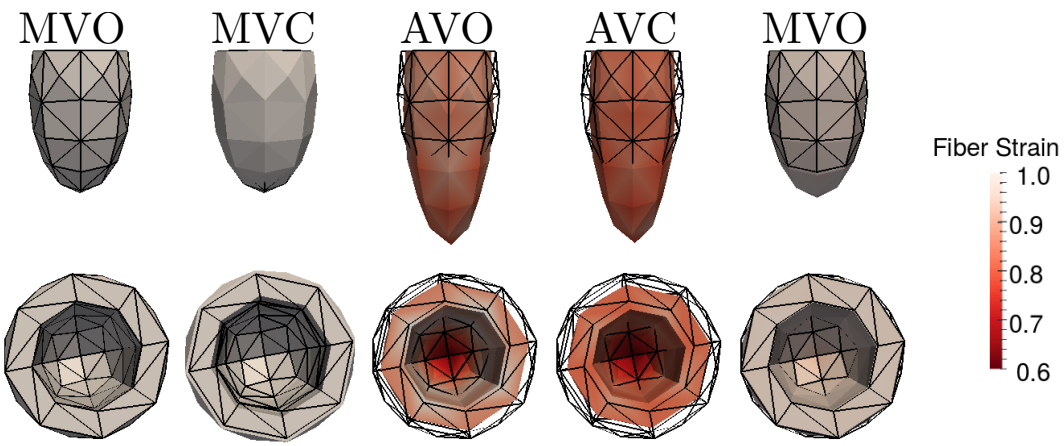


FIGURE 6.12: Displacement and fiber strain development for pressure volume loop with incompressible Neo-Hookean model

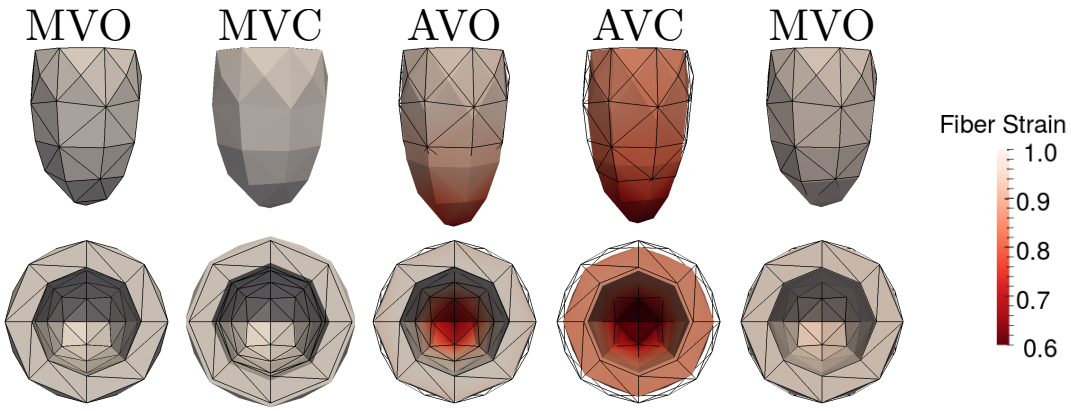


FIGURE 6.13: Displacement and fiber strain development for pressure volume loop with compressible Neo-Hookean model

phase with 20 % increase with respect to the undeformed volume, compared to 15% increase for the compressible model.

TABLE 6.1: Incompressible Neo-Hookean model

CPU time PV-loop	# Newton iterations per time step
76.68	495

TABLE 6.2: Compressible Neo-Hookean model

CPU time PV-loop	# Newton iterations per time step
12.25	555

TABLE 6.3: Time efficiency gain from FFC optimization for incompressible Neo-Hookean

Compiler properties	CPU time PV-loop
FFC optimized	76.68 s
FFC default	939.48 s

TABLE 6.4: Time efficiency gain from FFC optimization for compressible Neo-Hookean

Compiler properties	CPU time PV-loop
FFC optimized	12.25 s
FFC default	388.40 s

6.4 Optimization

6.4.1 Optimization of the form compiler

The performance of the default FFC paramters was very poor. The reason for this is that the default integration scheme of FFC compiler uses a high number of integration points per cell. By adjusting the form compiler parameters the efficiency improved very much. The FFC optimization parameters were switched on and specified as follows:

Python code

```
flags = ["-O3", "-ffast-math", "-march=native"]
parameters["form_compiler"]["quadrature_degree"] = 4
parameters["form_compiler"]["representation"] = "uflacs"
parameters["form_compiler"]["cpp_optimize"] = True
parameters["form_compiler"]["cpp_optimize_flags"] = " ".join(flags)
```

This has an extreme impact on the time efficiency of the solver. The time comparison is given in table 6.3 and 6.4. For the incompressible Neo-Hookean the time efficiency increases is by 12.25 times, from 939.48 seconds to 76.68 seconds. The compressible increases the time efficiency by 31.7 times, from 388.40 seconds to 12.25 seconds.

6.4.2 Optimizing by testing the linear solvers in FEniCS

The most time consuming part of solving the heart equations are the assembling of matrices and solving of linear systems. Hence, we wish to investigate the possibility to gain time efficiency by changing the linear solvers. In order to optimize this, we write

a program that iterates over all the linear solvers in FEniCS and compares the time for the entire PV-loop. This is done on two different mesh refinements, a coarse mesh with 690 elements, and a fine mesh with 5520 elements.

DOLFIN has several different linear solvers that can be specified in the solver parameters. There are both Krylov-method based solvers and LU-based solvers. The Krylov solvers did not converge for our method. The available LU based solvers that converged were the default lu-solver, the multifrontal massively parallel sparse direct solver (MUMPS), the built in LU solver from the PETSc library and the unsymmetric multifrontal sparse LU factorization method UMFPACK.

The PV-loop running in serial

We will first run the code in serial for both the incompressible and compressible Neo-Hookean model. This is done on the two different mesh refinements. The results for CPU-time and the number of Newton iterations for the incompressible model for the two refinements are given in Table 6.5, and Table 6.6. The LU-solver has the fastest solution time with 76.68 seconds for the complete PV-loop. The MUMPS and UMFPACK solvers follows closely with 78.77 seconds and 76.77 seconds, respectively. The PETSC solver is clearly the slowest one, and uses 395.80 seconds to complete the loop. The number of newton iterations is the same for all solvers. For the medium resolution mesh there is a much clearer difference between the different solvers. MUMPS is the fastets solver and uses 1113.63 seconds, while the slowest solver for this mesh is also the PETSC solver which uses 18091.90. The numbers of Newton iterations are 553 iterations for all solvers. The reason for this is probably that the small time steps causes the built in non-linear solver in FEniCS to converge for one Newton iteration for each time step for all solvers, such that the Newton steps are the same for all solvers.

The results for the compressible Neo-Hookean is given in Table 6.7 and 6.8. The compressible Neo-Hookean model is much faster than the incompressible version. This is as expected, as the compressible version only solves a system of equations for the displacement, whilst the incompressible version also has the pressure variable as unknown. For the low resolution mesh the fastest solver is the LU-solver from the PETSC library. The PV-loop is finished in 12.13 seconds using the PETSC solver as the specified linear solver. The other linear solvers are using almost identical amount of time. For the medium resolution mesh, the loop is finished in 61.70 seconds, where the LU-solver finished first. For this mesh refinement the PETSC solver is very slow, using twice the time as the other solvers. The number of Newton iterations are the same for all solvers on both mesh, which shows how the number of Newton iterations is dependent on the time step size.

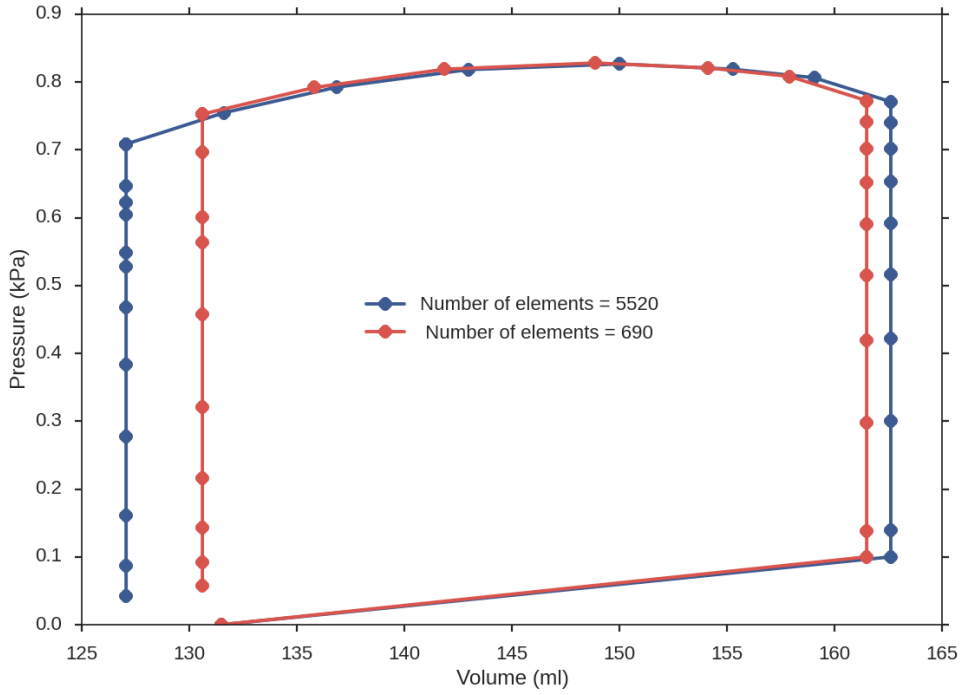


FIGURE 6.14: The PV-loop for the incompressible Neo-Hookean material model on different mesh refinements.

TABLE 6.5: Incompressible Neo-Hookean model,resolution mesh: low, number of cores = 1.

Linear solver	CPU time PV-loop	# Newton iterations PV-loop
LU	76.68	495
MUMPS	78.77	495
PETSC	395.80	495
UMFPACK	76.70	495

.

The PV-loops for both mesh refinements for the incompressible and compressible model is given in Figure 6.14 and Figure 6.15. For the incompressible Neo-Hookean the loops almomst coincides. The PV-loops are not identical but the volume and shape of the ventricle will vary slightly when the number of elements is increased. For the compressible model the loop for the coarse mesh is the most realistic, the loop for the finer refinement only has two points for the ejection phase, and hence lacks a realistic pressure volume development for this phase. The reason for this is most likely the values of the Windkessel parameters C and R. They were adjusted but the method is very sensitive regarding convergence such that this was the best loop that was achieved for this mesh resolution.

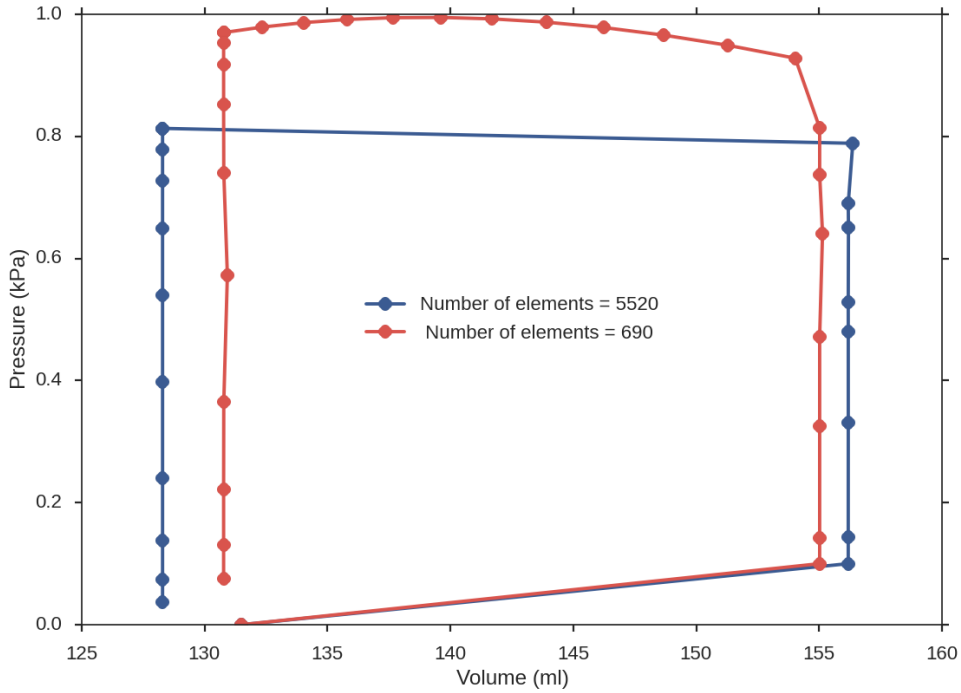


FIGURE 6.15: The PV-loop for the compressible Neo-Hookean material model on different mesh refinements.

TABLE 6.6: Incompressible Neo-Hookean model,resolution mesh: medium, number of cores = 1.

Linear solver	CPU time PV-loop	# Newton iterations PV-loop
LU	1811.91	553
MUMPS	1113.63	553
PETSC	18091.90	553
UMFPACK	1784.012	553

TABLE 6.7: Compressible Neo-Hookean model,resolution mesh: low, number of cores = 1.

Linear solver	CPU time PV-loop	# Newton iterations PV-loop
LU	12.25	555
MUMPS	12.40	555
PETSC	12.13	555
UMFPACK	12.15	555

TABLE 6.8: Compressible Neo-Hookean model,resolution mesh: medium, number of cores = 1.

Linear solver	CPU time PV-loop	# Newton iterations PV-loop
LU	61.70	375
MUMPS	61.98	375
PETSC	120	375
UMFPACK	62.25	375

6.4.3 Optimization by running the PV-loop run in parallel

We will now investigate the possibility for time optimization through running the PV-loop in parallel. During the parallelization the mesh is partitioned and distributed on multiple cores. The problem is then solved in parallel on the different cores. A picture of the partition of the fine mesh is given in Figure 6.16.

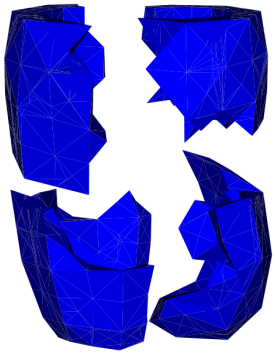


FIGURE 6.16: The division of the mesh with the code run in parallel

We test the code by running it in parallel on two and four cores for both incompressible and compressible material model. It is only the LU-solver and the MUMPS solver that are working in parallel for both models.

We will first discuss the results for the incompressible material model. We see from Table 6.9, that for the low resolution mesh, the PV-loop has the lowest run time in parallel on four cores, the LU-solver uses 58.21 seconds and 495 Newton iterations. From Table 6.11 we see that the same solver is fastest for the medium resolution mesh, using 763.28 seconds and 555 Newton iterations to complete the PV-loop. For the medium resolution mesh The MUMPS solver do not increase its time efficiency from two to four cores. The reason for this is that division of the mesh is probably more time demanding than the solving efficiency that is gained from running the model on four cores. Table 6.13 shows how the execution time for the different phases of the PV-loop is distributed, the solver

TABLE 6.9: Incompressible Neo-Hookean model, resolution mesh: low, number of cores = 2.

Linear solver	CPU time PV-loop	# Newton iterations PV-loop
LU	59.83	495
MUMPS	61.75	495
PETSC	-	-
UMFPACK	-	-

TABLE 6.10: Incompressible Neo-Hookean model, resolution mesh: low, number of cores = 4.

Linear solver	CPU time PV-loop	# Newton iterations PV-loop
LU	58.21	495
MUMPS	61.13	495
PETSC	-	-
UMFPACK	-	-

TABLE 6.11: Incompressible Neo-Hookean model, resolution mesh: medium, number of cores = 2.

Linear solver	CPU time PV-loop	# Newton iterations PV-loop
LU	789.49	553
MUMPS	798.23	553
PETSC	-	-
UMFPACK	-	-

that is used is the MUMPS solver, and the code is run in parallel on four cores on the low resolution mesh. We can see that the isovolumic contraction and isovolumic relaxation is the most time demanding phases for the incompressible Neo-Hookean model.

For the compressible model, and the low resolution mesh the best time is 10.59 seconds. This CPU-time was achieved while using the MUMPS solver on two cores. When running the loop on four cores it takes more time to divide the mesh, and communicate between the different cores, than it is to gain from solving it faster. For the finer refinement mesh the best time reached using the LU-solver was 42.46 seconds. The results for the code run in parallel for the compressible Neo-Hookean model are given in the Table 6.14, Table 6.15, Table 6.16 and Table 6.17.

The time efficiency for the different linear solvers run in parallel and series for the two mesh refinements with the incompressible and compressible Neo-Hookean are summarised

Table 6.12: Incompressible Neo-Hookean model, resolution mesh: medium, number of cores = 4.

Linear solver	CPU time PV-loop	# Newton iterations PV-loop
LU	763.28	555
MUMPS	808.97	555
PETSC	-	-
UMFPACK	-	-

Table 6.13: Newton iterations in each phase, incompressible Neo-Hookean, resolution = low, LU,# cores = 4

Phase	CPU time	# Newton iterations
Passive filling	0.73	4
Isovolumic Contraction	24.16	211
Ejection Phase	11.77	101
Isovolumic relaxation	21.56	179
Total	58.21	495

Table 6.14: Compressible Neo-Hookean model, resolution mesh: low, number of cores = 2.

Linear solver	CPU time PV-loop	# Newton iterations PV-loop
LU	10.8	555
MUMPS	10.59	555
PETSC	-	-
UMFPACK	-	-

Table 6.15: Compressible Neo-Hookean model, resolution mesh: low, number of cores = 4.

Linear solver	CPU time PV-loop	# Newton iterations PV-loop
LU	11.8	555
MUMPS	11.89	555
PETSC	-	-
UMFPACK	-	-

TABLE 6.16: Compressible Neo-Hookean model, resolution mesh: medium, number of cores = 2.

Linear solver	CPU time PV-loop	# Newton iterations PV-loop
LU	42.46	375
MUMPS	43.09	375
PETSC	-	-
UMFPACK	-	-

TABLE 6.17: Compressible Neo-Hookean model, resolution mesh: medium, number of cores = 4.

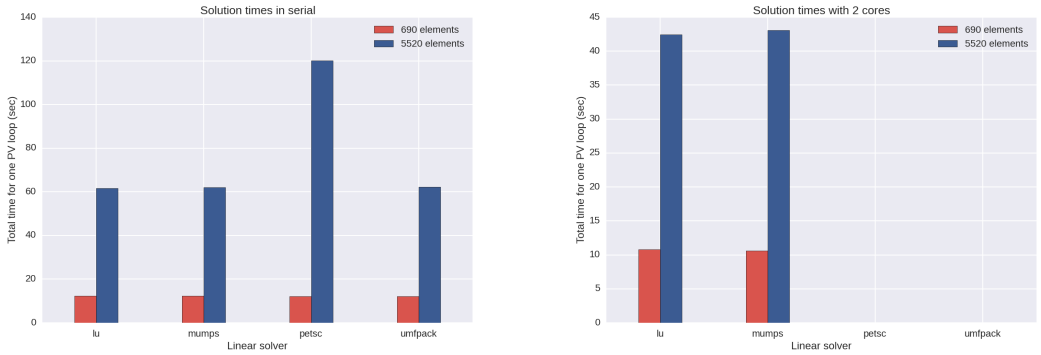
Linear solver	CPU time PV-loop	# Newton iterations PV-loop
LU	45.26	375
MUMPS	44.61	375
PETSC	-	-
UMFPACK	-	-

TABLE 6.18: Newton iterations in each phase, compressible Neo-Hookean, MUMPS,# cores 2

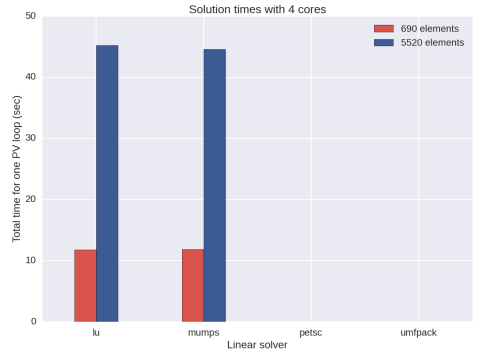
Phase	CPU time	# Newton iterations
Passive filling	0.21	4
Isovolumic Contraction	3.30	160
Ejection Phase	3.478	175
Isovolumic relaxation	3.81	216
Total	10.59	555

in the histograms in Figure 6.17 and 6.18.

The optimal version on our test cases is the compressible Neo-Hookean model run in parallel on two cores using the MUMPS as specified linear solver. The CPU-time for each phase together with the number of Newton iterations are given in Table 6.18. We can see that the three last phases of the heartbeat is the most time demanding. The passive filling phase uses only four Newton iterations and 0.21 seconds. Whilst the isovolumic relaxation uses 216 newton iterations and is the most time demanding. During the passive filling phase there is no active contraction while in the other phases there is a Newton step solving a constraint problem for each time step. This makes the last three phases more time demanding.

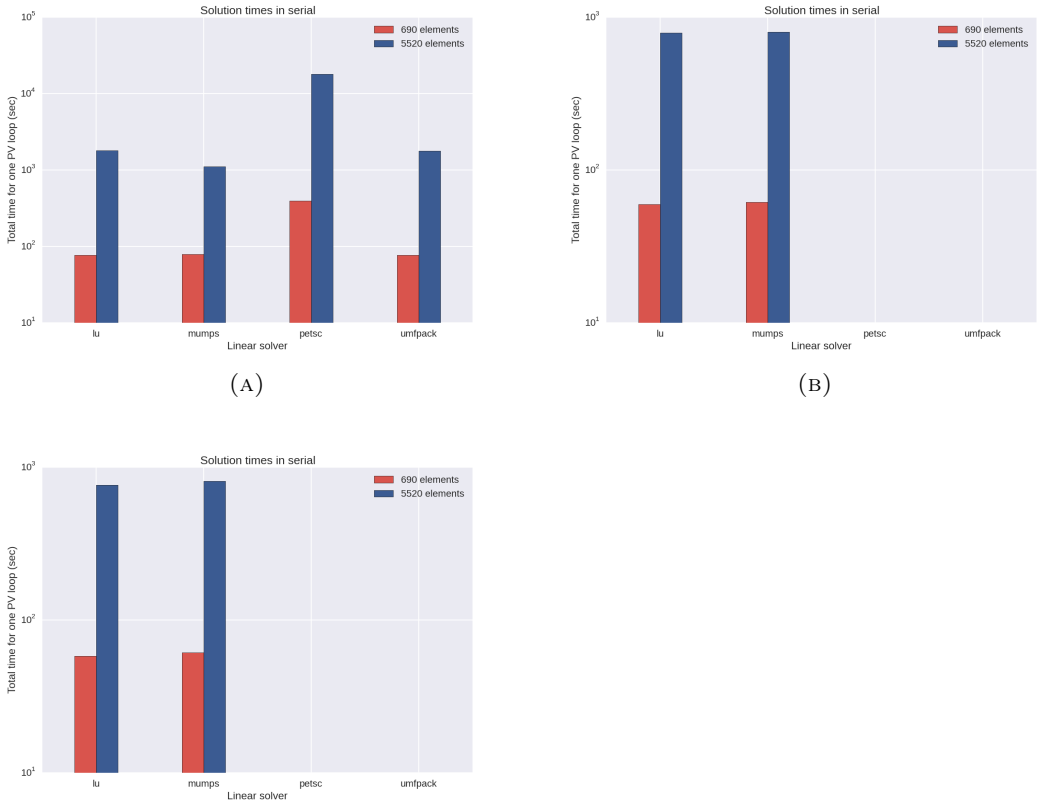


(A) (B)



(C)

FIGURE 6.17: The solution time for the different linear solvers in FEniCs run in series an in parallel on two and four cores.



(A)

(B)

(C)

FIGURE 6.18: The solution time for the different linear solvers in FEniCs with series and parallellization on two and four cores.

Chapter 7

Conclusion and further work

In the numerical experiments presented in this master thesis, we have modelled the myocardium as a hyperelastic isotropic material. The active contraction and relaxation of the myocardium have been simulated on a left ventricle geometry for both incompressible and compressible Neo-Hookean material model. The geometry used was an ellipsoidal surface of revolution. The fiber field for the cardiac muscle cells have been implemented along the circumference of the ellipsoid. In these numerical experiments the mechanical contraction of the heart is activated by a purely phenomenological model for the intracellular calcium concentration. We have compared our model to an analytic linear model and the results coincides, as expected, for small deformations. The method have also been verified trough the method of manufactured solutions, where the results also coincides.

The implementation of the PV-loop was done on two different mesh refinements. These models where compared with respect to time efficiency and how realistic the PV-loop was.

In the simulation of the passive filling phase the convergence problems of the model prevents the volume from increasing as much as desired. For the isovolumic contraction the pressure development is sufficient. The ejection phase shows a realistic pressure volume development for all cases except for the medium resolution mesh for the compressible Neo-Hookean model. For this case it was only two points for the entire phase. This is not realistic, as the model is very sensitive to adjustments of the parameters, we did not manage to get better results for this case. The isovolumic relaxation phase did not have convergence problems and managed to reduce the pressure to the start value for all cases. The reason for the higher stability of this phase is probably the lower steepness of the active force curve during the time interval for the relaxation phase.

The time efficiency of the code was optimized by manually increasing the time steps such that convergence and time efficiency was achieved. The most time demanding part of the code was the assembling of the matrices and solving the linear systems of the problem. Hence, all linear solvers of FEniCS was tested to find the most time efficient solver for this problem. The compiler parameters were also changed. This resulted in a great gain in time efficiency. This is by far the adjustment that had most impact on the time efficiency. For the incompressible model the time CPU-time improved with over 12 times, and for the compressible model the CPU-time increased by over 31 times. The fastest generation of the PV-loop was done in ca 10 seconds for the low resolution mesh, runnning in parallel, with the compressible Neo-Hookean material model, using the MUMPS solver.

We are interested in output that is close to reality. The PV-loop was the main output that was analysed for this model. As both the incompressible and compressible model had very similar pressure volume developments, the compressible version is clearly the preferred model to use in order to provide time efficiency. In order to further analyse the model it should be compared with known physical quantities for the heart.

The material model used throughout these numerical experiments is the Neo-Hookean model. This an isotropic material model. The choice of the material parameters was based on the convergence of the method. The myocardium is known to be orthotropic, therefore a more realistic material model and fiber field could be implemented in order to achieve physical output. The geometry used throughout this master thesis is an ellipsoidal surface of revolution. In order to have a more realistic model patient specific geometries could be used.

The model could also be coupled to the Eikonal Diffusion model [20] for electro physiology to activate the contraction of the cardiac cells.

Bibliography

[1] Dee Unglaub Silverthorn, William C Ober, Claire W Garrison, Andrew C Silverthorn, and Bruce R Johnson. *Human physiology: an integrated approach*. Pearson/Benjamin Cummings, 2009.

[2] Gerhard A Holzapfel. *Nonlinear solid mechanics*, volume 24. Wiley Chichester, 2000.

[3] Javier Bonet and Richard D Wood. *Nonlinear continuum mechanics for finite element analysis*. Cambridge university press, 1997.

[4] Joel Hass, Maurice D Weir, George Brinton Thomas, and George Brinton. *University calculus*. Pearson Addison-Wesley Boston, 2007.

[5] BR Munson, DF Young, and TH Okiishi. *Fundamentals of Fluid Mechancis*. 2002.

[6] Gerhard A Holzapfel and Ray W Ogden. 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, 2009.

[7] Julius M Guccione, Andrew D McCulloch, and LK Waldman. Passive material properties of intact ventricular myocardium determined from a cylindrical model. *Journal of biomechanical engineering*, 113(1):42–55, 1991.

[8] Martyn P Nash and Peter J Hunter. Computational mechanics of the heart. *Journal of elasticity and the physical science of solids*, 61(1-3):113–141, 2000.

[9] Jeffrey A Weiss, Bradley N Maker, and Sanjay Govindjee. Finite element implementation of incompressible, transversely isotropic hyperelasticity. *Computer methods in applied mechanics and engineering*, 135(1):107–128, 1996.

[10] PJ Hunter, AD McCulloch, and HEDJ Ter Keurs. Modelling the mechanical properties of cardiac muscle. *Progress in biophysics and molecular biology*, 69(2):289–331, 1998.

[11] L De Pater and JW Van den Berg. An electrical analogue of the entire human circulatory system. *Medical electronics and biological engineering*, 2(2):161–166, 1964.

[12] James P Keener and James Sneyd. *Mathematical physiology*, volume 1. Springer, 1998.

[13] Vincent Creigen, Luca Ferracina, Andriy Hlod, Simon van Mourik, Krischan Sjauw, Vivi Rottschäfer, Michel Vellekoop, and Paul Zegeling. Modeling a heart pump. *European Study Group Mathematics with Industry*, page 7, 2007.

[14] Patrick J Roache. Code verification by the method of manufactured solutions. *Journal of Fluids Engineering*, 124(1):4–10, 2002.

[15] Amy Henderson, Jim Ahrens, Charles Law, et al. *The ParaView Guide*. Kitware Clifton Park, NY, 2004.

[16] Will J Schroeder, Bill Lorensen, and Ken Martin. *The visualization toolkit*. Kitware, 2004.

[17] John D Hunter et al. Matplotlib: A 2d graphics environment. *Computing in science and engineering*, 9(3):90–95, 2007.

[18] Simone Rossi, Ricardo Ruiz-Baier, Luca F Pavarino, and Alfio Quarteroni. Orthotropic active strain models for the numerical simulation of cardiac biomechanics. *International journal for numerical methods in biomedical engineering*, 28(6-7):761–788, 2012.

[19] Henrik N. T. Finsberg. *https://bitbucket.org/finsberg/pulse_adjoint*. 2016. Online accessed August-2016.

[20] Ender Konukoglu, Jatin Relan, Ulas Cilingir, Bjoern H Menze, Phani Chinchapatnam, Amir Jadidi, Hubert Cochet, Meleze Hocini, Hervé Delingette, Pierre Jaïs, et al. Efficient probabilistic model personalization integrating uncertainty on data and parameters: Application to eikonal-diffusion models in cardiac electrophysiology. *Progress in biophysics and molecular biology*, 107(1):134–146, 2011.