Development of a pipeline for patient specific finite element modelling of the left ventricle of the human heart

M B A H C H R A M E H F R U

Master of Science Thesis
Stockholm, Sweden 2014

Development of a pipeline for patient specific finite element modelling of the left ventricle of the human heart

M B A H C H R A M E H F R U

Master's Thesis in Scientific Computing (30 ECTS credits)
Master Programme in Scientific Computing (120 credits)
Royal Institute of Technology year 2014
Supervisor at KTH was Johan Jansson
Examiner was Michael Hanke

TRITA-MAT-E 2014:67
ISRN-KTH/MAT/E--14/67--SE

Royal Institute of Technology
School of Engineering Sciences

KTH SCI
SE-100 44 Stockholm, Sweden

URL: www.kth.se/sci

Abstract

In order to perform simulations of the human heart using a heart finite element solver developed at the Computational Technology Laboratory at KTH with a data set provided by Philips consisting of surface meshes of a whole heart, the surface mesh has to be converted to a volume mesh. The conversion is manual and time consuming. Therefore the purpose of this thesis is to develop algorithms and software tools for automatic generation of a finite element model in the form of a volume mesh of the left ventricle of a human heart based on the available Philips data set. The developed model can be used for the simulation of blood flow by solving the Navier–Stokes equations. The method used for generating the model is based on deformation of an *a priori* finite element volume mesh to fit the extracted inner wall surface mesh of the left ventricle, from the aforementioned data set. The deformation is done by solving a nonlinear partial differential equation (PDE) using the finite element method.

The method starts with the characterization of an external field that describes the distance from the target surface mesh, and then uses this external field as a component of the total force responsible for deforming the object. The method is validated in three space dimension by deforming a sphere into an ellipsoid. For this test case, two implementations of the PDE were tested and evaluated.
The method was then applied to the above mentioned Philips data set. The report summarizes the findings and proposes improvements for the future work.

Utveckling av arbetsflöde för patientspecifik finit element modellering av vänster kammare på mänskligt hjärta

Sammanfattning

För att utföra simuleringar av ett mänskligt hjärta med hjälp av en finita elements hjärtlösare på Computational Technology Laboratory på KTH, med data från Philips innehållande ytnät från ett helt hjärta, krävs att ytnätet omvandlas till ett volymnät. Omvandlingen görs för hand och tar mycket tid. Syftet med den här uppsatsen är därför att utveckla algoritmer och mjukvaruverktyg för automatisk generering av finita elementmodellen i form av ett volymnät av ett mänskligt hjärtas vänsterkammare baserat på tillgänglig data från Philips. Den utvecklade metoden kan användas för blodflödessimulering genom att lösa Navier-Stokesekvationerna. Metoden som har använts för att generera modellen baseras på deformering av ett förut bestämd finit elementnät för att passa ytnätet på vänsterkammarens uttagna innervägg från Philips data. Deformeringen görs genom att lösa en ickelinjär partialdifferentialekvation med hjälp av finita element metoden.

Metoden börjar med karaktäriseringen av ett yttre fält som beskriver avståndet från det önskvärda ytnätet och sedan använder detta yttre fält som en del i den totala kraften som ansvarar för objektets deformering. Metoden har verifierats i tre rymddimensioner genom att en sfär deformeras till en ellips. I detta fall testades och utvärderades två implementeringar av partialdifferentialekvation. Sedan applicerades metoden på den tidigare nämnda datan från Philips. Rapporten sammanfattar upptäckterna och föreslår framtida förbättringar.

Contents

Chapter 1

Introduction

The human heart is one of the most important organs which is responsible for the continuous blood supply to all parts of the human body [29, 30]. It is a three dimensional muscular structure consisting of four chambers namely, the right and left atrium, and the right and left ventricle. The left ventricle receives oxygenated blood from the lungs and pumps it to the body while the right ventricle receives oxygen deficient blood from the body and pumps it to the lungs. The flow of blood into the ventricles is controlled by valves which allow the blood to flow only in one direction at a given time [14]. Problems pertaining to the heart can sometimes be fatal, in fact, it has been shown that heart disease is one of the major causes of death in the Western world [27] and [12]. For the cases where heart disease does not lead to death, the geometry and function of the heart might change as a result of the disease [18]. These changes over time differ from person to person. Accordingly, an impaired functionality of the left ventricle might lead to an abnormal cardiac cycle [18]. Therefore understanding the dynamics of the heart (which includes structural changes over a cardiac cycle, systole and diastole, and blood flow) is crucial. The aforementioned understanding can be achieved by studying non-invasive methods.

Concerning non–invasive imaging modalities such as phase contrast MRI (magnetic resonance imaging), Doppler ultrasound imaging and computed tomography (CT), at the moment, none of them can be used for obtaining detailed quantitative information about the dynamics of the heart [18] for the following reasons:

With respect to Doppler imaging, it is a non-invasive procedure that is used to measure blood flow through the blood vessels by sending high frequency sound waves to the region of interest, where the waves are reflected back upon hitting red blood cells. This type of imaging differs from regular ultrasound imaging in that in regular ultrasound imaging, only images of the tissue are produced and no information about blood flow is obtained. It has been reported by [18] that for Doppler ultrasound imaging, the angle between the ultrasound beam and blood flow (see [26]) becomes an impediment when measuring the received signal.

For the case of phase contrast MRI, it relies on the fact that a uniform motion of tissue (for example the heart) in a magnetic field gradient causes a change in phase of the magnetic resonance signal [22]. This phase change is proportional to the velocity of the tissue. Conventional MRI produces three dimensional images for diagnostic purposes but is void of information pertaining to blood flow and tissue movement. As

mentioned by [18] the disadvantage of phase contrast MRI is due to the fact that it takes time to get the signal and its spatial resolution is small compared to CT.

Computed tomography (CT) is an imaging method which uses X-rays and computer processing to generate an image of tissue density in cross-sections through the patient's body. It is a very fast imaging method and has a high spatial resolution [4]. However, the radiation produced by CT scans is high and not suitable for the purpose of getting an image from a healthy person since there is an increased risk of developing cancer in future.

Despite these limitations, the inclusion of a mathematical model geared towards Computational Fluid Dynamics (CFD) to compliment the non-invasive methods has been in continuous development by many authors. As explained in [21], mathematical models have the advantage of gaining profound insights of physiological mechanisms which may not be measured easily. Also, it may help to clarify certain concepts where ambiguity arises due to subjective definitions and interpretations [21]. CFD deals with the prediction of the behaviour of fluids and the effects of fluid motion past objects by solving numerically a set of partial differential equations that describe the flow [11]. Usually, these equations with appropriate boundary conditions are set up on a mesh, which is a representation of a domain of interest that is partitioned into finite elements [6]. The obtained solutions consist of velocity and pressure at each mesh point at a given time.
Since human anatomy varies from person to person, mesh generation has to be patient specific and is typically done in a computational multi-stage approach consisting of: segmentation of medical image, surface mesh generation, volume mesh generation and mesh optimization [25]. This approach has to be carried out such that the generated mesh should be valid, geometrically accurate, smooth and have an appropriate mesh quality [25].

Remarkable results have been obtained using CFD to simulate blood flow in the left ventricle of the human heart [18, 28]. In this framework, CFD approaches can be grouped into two categories [8, 28], namely: a prescribed method and fluid-structure interaction (FSI). The former makes use of an equation to specify the boundary of the patient specific data acquired from one of the medical imaging methods mentioned earlier while the latter takes into account the interaction of heart muscles (structure) with the internal (or in some other cases, external) blood flow (fluid) and depicts more realism than the former [28]. In the context of FSI, it can either be done in a monolithic fashion or by solving the equations for fluid flow and structure deformations separately and couple them together by updating the boundary conditions at the interface between structure and fluid, iteratively. In the monolithic approach, the equations for fluid flow and structure deformation are solved together without decoupling.

Combining the ideas in the preceding section to develop a patient specific model for the left ventricle, which can be helpful in diagnosis, has been a challenge. This is due to the many time consuming manual steps involved from acquiring raw data (images) to creating a model [20]. Some authors have used different approaches to automate this process by using a generic model. In fact, Philips [31] has developed a patient specific model of the whole heart composed of surface meshes that can be generated from any medical imaging method. They started with a generic model of the heart and adaptively matched it to the three dimensional images of the patient [31], while in [23], they had an *a priori* model of an object. This object is in the form of an image to be segmented. This *a priori* model is defined such that it has the same topology, geometry and elastic characteristics as the object. The model is then put into the image data where a fictitious force field deforms the

model by pulling its boundaries towards the image edges until it reaches equilibrium. The fictitious force is known as gradient vector flow (GVF) [32]. It is derived by minimizing a certain energy functional of the image data. After achieving the preceding phases with the aforementioned model, it can now be used for tracking the heart motion in the cardiac cycle and extracting useful parameters such as volumes, local stresses and strain. However this method is tailored to MRI data. An approach which has not yet been tested is to use the Philips heart model and an a priori model for the automatic generation of a patient specific heart model. The novelty of this project lies in the fact that the GVF will be used on the Philips heart model to construct an external force field. Although the Philips heart model is not an image, it has internal surfaces which makes it suitable to use the GVF.

The advantages of the gradient vector flow approach are: It does not depend on initialization and it is able to capture boundaries [23, 32]. This approach is particularly interesting with respect to the existing heart solver code under development at the Computational Technology Laboratory at KTH. To use the solver with an available data set from Philips, consisting of surface meshes of time snapshots of a whole human heart, the model has to be converted to a volume mesh before it can be used. The process of conversion is manual and laborious. Thus, this calls for an automated process. The automation is done by deforming the *a priori* model in the form of a volume mesh by solving a non linear partial differential equation using a finite element method.

Therefore, the purpose of this work is to develop algorithms and software tools such that patient specific data based on the Philips heart model, can be automatically deformed into a finite element model wherein the Navier–Stokes equations can be solved to simulate blood flow in the left ventricle of the heart, using an existing heart solver code. The tools and algorithms should work in a parallel computing environment because they will be developed using FEniCS-hpc software. FEniCS-hpc takes a weak form of a partial differential equation as input and generates a solver which is automatically parallellized. One of the components of FEniCS-hpc which aids the parallelization is Dolfin-hpc. In Dolfin-hpc a mesh is automatically distributed and Dolfin-hpc contains parallel libraries for linear algebra. This means that each processing element (core) will have a copy of only the portion of the mesh for which it can carry out computations. In the chapters that follow, Chapter 2 lays the ground for the theoretical aspects of the model deformation, Chapter 3 gives a brief introduction to the software and Chapter 4 presents results of the test case. Chapter 5 presents applications and results with some discussions, and Chapter 6 highlights what has been achieved in this work and what lies ahead.

Chapter 2

Methods

2.1 Introduction

In order to create a patient specific left ventricle model from a generalized model, based on the Philips heart model, the generic model has to be adjusted or deformed automatically to fit the data. This deformation is achieved by an external force field constructed from the data. This chapter introduces how an elastic object in an external force field is deformed. The characteristics of the external force field is described, and thereafter how the external force field causes the object to deform. The resulting equations are then solved using a finite element method.

2.2 Force field

The external force that will be used in deforming an elastic object is based on the gradient vector flow (GVF) force field [23, 32]. Although the gradient vector flow field is derived from images by a minimization of a certain energy functional [32], in this project there are no images involved. Instead, the GVF is derived from a general domain exhibiting regions of interest like boundaries or internal surfaces. Let Ω be an open bounded domain of $\mathbb{R}^3$ and $\partial\Omega$ its boundary. The GVF in Ω is defined as a vector field (external force field) $\mathbf{U}$ obeying

$$\begin{aligned} \alpha\nabla^2\mathbf{U} - |\nabla f|^2(\mathbf{U} - \nabla f) = 0 \ \text{ in } \ \Omega, \\ \mathbf{U} = 0 \ \text{ on } \ \partial\Omega. \end{aligned} \tag{2.1}$$

With

$$f : \Omega \longrightarrow \mathbb{R},$$

a real valued function and α a parameter which accounts for the balance between the first term and the second term of equation (2.1): if there are irregularities in the domain, increasing α smooths out these irregularities. In [32], f is referred to as the edge map function. The edge map function is in principle an edge detection method which identifies significant discontinuities of gray level of an image [7]. In this project, f is chosen to have similar properties as the edge map function, with the goal to identify the boundary of the heart model. These properties include:

- ∇f has vectors pointing towards the boundary and are normal to the boundary at the boundaries. This will cause the model's boundary undergoing deformation to converge to the target boundary.

- ∇f has a large magnitude only in the immediate vicinity of the boundary.
- In a homogeneous region (for example, the inner part of the left ventricle), ∇f is almost zero.

Therefore f is chosen as

$$f(x,y,z) = \frac{1}{\bar{\sigma}\sqrt{2\pi}} \exp\left(\frac{-g(x,y,z)^2}{2\bar{\sigma}^2}\right), \tag{2.2}$$

where $g(x,y,z)$ measures the distance from the boundary to the center of the domain, that is $g(x,y,z) = 0$ on the boundary, and $\bar{\sigma}$ (full width at half maximum) is a parameter which determines how wide or thin the boundary is.
In general, when an analytical expression for distance is not possible to derive from complex geometries (like the left ventricle), the solution to the Eikonal equation (equation (2.3)) can be used to get the distance, function $g(x,y,z)$:

$$\begin{aligned} |\nabla g(x,y,z)| &= 1 \quad (x,\ y,\ z) \quad \text{in} \quad \Omega \\ g(x,y,x) &= 0 \quad (x,\ y,\ z) \quad \text{on} \quad \partial\Omega \end{aligned} \tag{2.3}$$

Some observations can be made from equation (2.1) such as: if $|\nabla f|$ is very small, the force field is dominated by $\nabla^2\mathbf{U}$ term, corresponding to a Poisson problem, giving an almost constant field $\mathbf{U}$. If $|\nabla f|$ is very large the second term becomes pronounced, and equation (2.1) is solved by setting $\mathbf{U} = \nabla f$. Thus equation (2.1) enforces the condition that $\mathbf{U}$ should be equal to the gradient of f at the immediate vicinities of the boundary, and it also ensures that the field $\mathbf{U}$ vary slowly far into the domain (homogeneous) regions. To solve equation (2.1), we seek the steady-state solution of the diffusive process [32]:

$$\begin{aligned} \frac{\partial \mathbf{U}}{\partial t} = \alpha\nabla^2\mathbf{U} - (\mathbf{U} - \nabla f)|\nabla f|^2 \ \text{ in } \ \Omega, \\ \mathbf{U} = 0 \ \text{ on } \ \partial\Omega. \end{aligned} \tag{2.4}$$

2.3 Model Deformation

In this section, deforming the object (which is in the form of a volume mesh) is described in terms of a time dependent non-linear partial differential equation (PDE). First, a general description of the PDE will be given followed by connecting the mesh quality to the PDE. Consider that the mesh is modelled as an isotropic elastic object, having an external force field (described in the previous section) acting on it and Ω, $\partial\Omega$ represents the mesh and its boundary respectively. Also let $\mathbf{T} = [0,\ T] \subset \mathbb{R}$ be the time domain. There exists two types of forces acting on the object, namely the force field which acts on the whole domain and a force acting on the boundary of the object due to stress [15],

$$\sigma\colon \Omega \times \mathbf{T} \longrightarrow \mathbb{R}^3 \times \mathbb{R}^3,$$

which is of the form $\sigma \cdot \mathbf{n}$, where n is normal to the surface. The stress, is given by

$$\sigma = 2\mu\mathbf{e}, \tag{2.5}$$

where $\mathbf{e}\colon \Omega \times \mathbf{T} \longrightarrow \mathbb{R}^3 \times \mathbb{R}^3$ is the strain given by $\mathbf{e} = \mathbf{I} - \mathbf{B}$, $\mathbf{B}\colon \Omega \times \mathbf{T} \longrightarrow \mathbb{R}^3 \times \mathbb{R}^3$ with $\mathbf{B} = \mathbf{F}\mathbf{F}^T$, $\mathbf{F}\colon \Omega \times \mathbf{T} \longrightarrow \mathbb{R}^3 \times \mathbb{R}^3$ represents the deformation gradient given by $\mathbf{F} = \mathbf{I} + \nabla\mathbf{w}$, $\mathbf{I}$ is the identity matrix, $\mathbf{w} : \Omega \times \mathbf{T} \longrightarrow \mathbb{R}^3$ is the displacement vector, and

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

is the elastic modulus. E and ν are Young's elastic modulus and Poisson's ratio respectively. Young's modulus tells us how stiff the object is and the Poisson's ratio expresses the propensity of the object to shrink when stretched. The deformation gradient can be interpreted as a mapping, from cells in a reference mesh to a current mesh configuration after the mesh has undergone a deformation. The deformation gradient evolves during the deformation process and so does the stress. According to [10], this evolution is given by

$$\frac{\partial \mathbf{F}}{\partial t} = \nabla \mathbf{u}\mathbf{F}, \tag{2.6}$$

with initial condition $\mathbf{F}_0 = \hat{\mathbf{F}}$, $\hat{\mathbf{F}}$ is the deformation gradient with respect to a scaled equilateral reference cell having the maximum mesh quality Q, given by $Q = 1$ and $\mathbf{u} : \Omega \times \mathbf{T} \longrightarrow \mathbb{R}^3$ the displacement velocity vector. The mesh quality is defined to be

$$Q = \frac{\mathrm{d} \left\|\mathbf{F}\right\|_F^2}{\det(\mathbf{F})^2} \tag{2.7}$$

where d is the dimension of the mesh and $\left\|\mathbf{F}\right\|_F$ is the Frobenius norm defined as

$$\left\|\mathbf{F}\right\|_F \quad = \quad \sqrt{\mathrm{trace}(\mathbf{F}\mathbf{F}^T)}. \tag{2.8}$$

$\mathbf{F}$ is computed at a current state and the stress (equation (2.5)) is updated accordingly since $\mathbf{B}$ also evolves in time. The evolution of $\mathbf{B}$ (see [10]) is via

$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \mathbf{u}\mathbf{B} + \mathbf{B}\nabla \mathbf{u}^T. \tag{2.9}$$

The total force on the object is obtained by summing up external forces and contact forces

$$\tilde{F} = \int_{\partial\Omega} \sigma \cdot \mathbf{n} d\Gamma + \int_{\Omega} \mathbf{U} d\Omega, \tag{2.10}$$

where $\mathbf{U}$ is the external force obtained by solving equation (2.4) and $\Gamma = \partial\Omega$. From Newton's second law (change in momentum per unit time), $\tilde{F}$ can be expressed as $\int_\Omega \rho \ddot{\mathbf{w}} d\Omega$, where ρ is the density, $\ddot{\mathbf{w}}$ is the rate of change of velocity and the integral indicates the sum of all particles in the body occupying a volume $d\Omega$ and mass $\rho d\Omega$.
Thus equation (2.10) becomes

$$\int_{\Omega} \rho \ddot{\mathbf{w}} d\Omega = \int_{\partial\Omega} \sigma \cdot \mathbf{n} d\Gamma + \int_{\Omega} \mathbf{U} d\Omega \tag{2.11}$$

Applying the divergence theorem on the first term on the right hand side, yields

$$\int_{\Omega} \rho \ddot{\mathbf{w}} d\Omega = \int_{\Omega} (\nabla \cdot \sigma + \mathbf{U}) d\Omega \tag{2.12}$$

At equilibrium, the forces balance out, giving

$$\int_{\Omega} (\rho \ddot{\mathbf{w}} - \nabla \cdot \sigma - \mathbf{U}) d\Omega \quad = \quad 0,$$

from which follows that

$$\rho \ddot{\mathbf{w}} \quad = \quad \nabla \cdot \sigma + \mathbf{U} \tag{2.13}$$

Therefore with appropriate boundary conditions, the Elasticity PDE reads (dropping the density since it acts like a scaling factor)

$$\begin{aligned} \ddot{\mathbf{w}} &= \nabla \cdot \sigma + \mathbf{U}, && \text{in} \quad \Omega \times \mathbf{T} \\ \sigma &= 2\mu(\mathbf{I} - \mathbf{B}) && \text{in}, \quad \Omega \times \mathbf{T} \\ \sigma \cdot \mathbf{n} &= \mathbf{g_1} && \text{on} \quad \partial\Omega \times \mathbf{T} \\ \mathbf{w} &= \mathbf{g_2} && \text{on} \quad \partial\Omega \times \mathbf{T} \\ \mathbf{w} &= \mathbf{g_4} && \text{in} \quad \Omega \quad \text{for} \quad t = 0 \\ \dot{\mathbf{w}} &= \mathbf{g_3} && \text{in} \quad \Omega \quad \text{for} \quad t = 0 \end{aligned} \tag{2.14}$$

Where $\mathbf{g_1} : \Omega \times \mathbf{T} \longrightarrow \mathbb{R}^3$ and $\mathbf{g_2} : \Omega \times \mathbf{T} \longrightarrow \mathbb{R}^3$ correspond to the Neumann and Dirichlet boundary condition respectively, $\mathbf{g_4} : \Omega \longrightarrow \mathbb{R}^3$ is the initial condition on the displacement which corresponds to the initial mesh configuration and $\mathbf{g_3} : \Omega \longrightarrow \mathbb{R}^3$ is the initial condition on the rate of change of the displacement (velocity). If $\mathbf{g_2} = 0$, then the boundary of the object is not allowed to move. Equation (2.14) is solved with a two step procedure. First calculate $\mathbf{U}$ by evolving equation (2.4) and then inserting it in equation (2.14). In Chapter 3, the implementation of this solution algorithm will be discussed.
For a discrete setting, we discretise the domain Ω by the mesh which is partitioned into finite set of cells, K, with non overlapping interiors such that $\cup \mathrm{K} = \Omega$. The cells are usually of polygonal shapes such as tetrahedral. The $\mathbf{L}_2$ norm quality of each cell in the mesh is given by

$$\|Q\|^2_{\mathbf{L_2}} = \int_{\mathrm{K}} Q^2 dx, \tag{2.15}$$

with dx being cell dimension. To see how the mesh quality is influenced by updating the mesh with $\mathbf{w}$ from equation (2.14), we study the energy of the body since the stationary solution of equation (2.14) corresponds to the minimum energy (potential).
The energy functional is given by

$$E(\mathbf{w}) = E_{\text{elastic}} + E_{\text{external}} \tag{2.16}$$

$$\begin{aligned} E_{\text{elastic}} &= \int_\Omega \frac{1}{2}\sigma_{ij} e_{ij} d\Omega \\ \text{but } \sigma_{ij} &= \mu\, e_{ij} \\ E_{\text{elastic}} &= \int_\Omega \frac{1}{2}\mu e_{ij}\, e_{ij} d\Omega \\ E_{\text{elastic}} &= \int_\Omega \frac{1}{2}\mu\, \text{trace}(\mathbf{e}^T\mathbf{e}) d\Omega \\ \text{but } \mathbf{e} &= \mathbf{I} - \mathbf{B} \\ E_{\text{elastic}} &= \int_\Omega \frac{1}{2}\mu \text{trace}((\mathbf{I} - \mathbf{B})^T(\mathbf{I} - \mathbf{B})) d\Omega \end{aligned} \tag{2.17}$$

For each cell in the mesh

$$\begin{aligned} E_{\text{elastic}} &= \sum_{\mathrm{K}\in\Omega} \int_{\mathrm{K}} \frac{1}{2}\mu\, \text{trace}((\mathbf{I} - \mathbf{B})^2) d\Omega \\ &= \sum_{\mathrm{K}\in\Omega} \int_{\mathrm{K}} \frac{1}{2}\mu\, \text{trace}(\mathbf{I} - 2\mathbf{B} + \mathbf{B}^T\mathbf{B}) d\Omega \\ &= \sum_{\mathrm{K}\in\Omega} \int_{\mathrm{K}} \frac{1}{2}\mu \left[\text{trace}(\mathbf{I}) - 2\,\text{trace}(\mathbf{B}) + \text{trace}(\mathbf{B}^T\mathbf{B})\right] d\Omega \end{aligned} \tag{2.18}$$

From the definition of Frobenius norm

$$\|\mathbf{F}\|_F = \sqrt{\text{trace}(\mathbf{F}\mathbf{F}^T)} = \sqrt{\text{trace}(\mathbf{B})}$$

From equation (2.7), neglecting the normalisation and scale factors, one obtains

$$\begin{aligned} Q &= \|\mathbf{F}\|_F^2 \\ &= \text{trace}(\mathbf{B}) \end{aligned} \tag{2.19}$$

Note that

$$\begin{aligned} \mathbf{B}^T\mathbf{B} &= (\mathbf{F}\mathbf{F}^T)^T\mathbf{F}\mathbf{F}^T \\ &= \mathbf{F}\mathbf{F}^T\mathbf{F}\mathbf{F}^T \\ &= \mathbf{B}^2 \end{aligned} \tag{2.20}$$

trace($\mathbf{B}$) is computed from equation (2.18) and using equation (2.15), the following is obtained

$$\begin{aligned} \|Q\|_{L_2} &= \int_{\text{K}} Q^2 dx \\ &= \int_{\text{K}} \text{trace}(\mathbf{B})^2 dx \end{aligned} \tag{2.21}$$

Computing trace($\mathbf{B}$) by minimizing equation (2.18), the cell quality is evaluated as illustrated by equation (2.21). Thus, when at equilibrium equation (2.18) acts as a smoother by giving high stiffness to cells with low Q value.

The creation of the mesh goes through the following steps.

while $t < T$ **do**
  Compute the solution to the Eikonal equation, $g(x, y, z)$
  Compute: $f(x, y, z) = \frac{1}{\bar{\sigma}\sqrt{2\pi}} \exp\left(\frac{-g(x,y,z)^2}{2\sigma^2}\right)$
  Solve: $\frac{\partial \mathbf{U}}{\partial t} = \alpha\nabla^2\mathbf{U} - |\nabla f|^2(\mathbf{U} - \nabla f)$ with BC condition
  Solve: $\ddot{\mathbf{w}} = \nabla \cdot \sigma + \mathbf{U}$
  Check for convergence by examining the change in volume of the mesh
  if no change in volume **then**
    break
  end if
  $t+ =$ time step
end while

2.4 Numerical implementation

Equations (2.3), (2.4) and (2.14) are solved numerically by the finite element method. As mentioned in [8], finite element methods possess a strong mathematical base which employs measuring an *a posterior* estimate of the error, which is a corner stone for adaptive methods.

Consider an arbitrary domain, $\Omega \subset \mathbb{R}^d \quad d = 2 \text{ or } 3$, of a given problem. This domain is assumed to be discretized by the mesh mentioned above. A local function space, V, is defined on each cell with a set of rules assigned to the functions in V. These local function

spaces are then used in constructing a global function space, V_h, known as the finite element space. Typically, the finite element discretization steps are:

1. Derive a weak formulation. This entails multiplying the equation to be solved by an appropriate test function and integrating to obtain a weak formulation. The space of the test functions is taken to be the Solobev space given by

$$\mathrm{H}_0^1(\Omega) := \{v \in \mathbf{L}^2(\Omega) \,|\, \nabla v \ \in \mathbf{L}^2(\Omega), \ \ v\,|_{\partial\Omega} = 0\}$$

   where

$$\mathbf{L}^2(\Omega) := \{v \,|\, \left(\int_\Omega |v|^2 d\Omega\right)^{\frac{1}{2}} < \infty\}$$

   ∇v is the gradient of v. The finite element solution is then sought from a finite element space, $V_h \subset \mathrm{H}_0^1(\Omega)$, build from a local function space on each cell of the mesh.

2. Obtain an algebraic system. The solution to be computed known as the trial function, is expressed as a linear combination of basis functions on each element in the domain. Each basis function has a compact support either across neighbouring element (for continuous finite element) or within an element (discontinuous finite element). The linear combination is then substituted in the weak formulation with the test functions chosen as the basis functions yielding a discretized weak formulation which could be a linear or a non linear algebraic system.

3. Next step involves evaluation of the integrals representing the weak formulation using a reference coordinate system.

4. Solving the algebraic system.

2.4.1 Finite Element method for the Eikonal equation

To solve equation (2.3) numerically, we use the continuous Galerkin's method of degree one ($cG(1)$) (using Lagrange elements), with Newton's method since it is a non linear partial differential equation. The $cG(1)$ method is defined by using continuous trial and test functions of degree one. As noted in [5], the addition of artificial viscosity smooths out the discontinuity in regions where the gradient of the solution is less than one. Adding artificial viscosity is at the expense of accuracy but a gain in stability. This idea is used in this numerical scheme as follows;

Squaring both sides of equation (2.3) and adding an artificial viscosity term, $\beta\Delta g$, where β is a small constant proportional to the mesh size h, and g is the finite element solution, the following is obtained

$$\beta\Delta g + |\nabla g|^2 = 1 \tag{2.22}$$

Multiplying equation (2.22) with a test function, $v \in \mathrm{H}_0^1(\Omega)$, and integrate

$$(\beta\Delta g,\, v) + (|\nabla g|^2,\, v) = (1,\, v), \tag{2.23}$$

where

$$(\gamma,\, \theta) = \int_\Omega \gamma\theta d\Omega$$

Using integration by parts on the first term of equation (2.23) and applying boundary conditions on the test function gives

$$(\beta\nabla g,\, \nabla v) + (|\nabla g|^2,\, v) = (1,\, v) \tag{2.24}$$

To derive Newton's method, let

$$g = g_o + \partial g \tag{2.25}$$

where g_o is a know approximation of g and ∂g is a correction term. Substituting into equation (2.24)

$$\begin{aligned} (\beta\nabla(g_o + \partial g),\, \nabla v) + (|\nabla(g_o + \partial g)|^2,\, v) &= (1,\, v) \\ (\beta\nabla g_o + \beta\nabla\partial g,\, \nabla v) + (|(\nabla g_o)^2 + (\nabla\partial g)^2 + 2\nabla g_o\,\nabla\partial g|,\, v) &= (1,\, v) \end{aligned}$$

The absolute value sign can be dropped since it has been ensured that it is always positive by squaring it

$$(\beta\nabla g_o,\, \nabla v) + (\beta\nabla\partial g,\, \nabla v) + ((\nabla g_o)^2 + (\nabla\partial g)^2 + 2\nabla g_o\nabla\partial g),\, v) = (1,\, v)$$

$$(\beta\nabla g_o,\, \nabla v) + (\beta\nabla\partial g,\, \nabla v) + ((\nabla g_o)^2,\, v) + ((\nabla\partial g)^2,\, v) + (2\nabla g_o\nabla\partial g,\, v) = (1,\, v) \tag{2.26}$$

Dropping higher order terms in ∂g in equation (2.26), that is the term $((\nabla\partial g)^2,\, v)$, to get

$$(\beta\nabla g_o,\, \nabla v) + (\beta\nabla\partial g,\, \nabla v) + ((\nabla g_o)^2,\, v) + (2\nabla g_o\nabla\partial g,\, v) = (1,\, v),\ \ \forall\, v \in V_h \tag{2.27}$$

Equation (2.27) is solved for ∂g and the Newton's method is used to update g according to equation (2.25).

Let $V_h \subset \mathrm{H}_0^1(\Omega)$ be a space of continuous piecewise linear functions on Ω. Replacing g_o with U_h^o, $(U_h^o \in V_h)$, the finite element approximation for equation (2.27) reads, find $\partial U \in V_h$ such that

$$(\beta\nabla U_h^o,\, \nabla v) + (\beta\nabla\partial U_h,\, \nabla v) + ((\nabla U_h^o)^2,\, v) + (2\nabla U_h^o\nabla\partial U_h,\, v) = (1,\, v),\ \ \forall\, v \in V_h \tag{2.28}$$

Taking the basis of the space to be $\{\phi_i\}_i^{n_i}$ with n_i interior nodes, then setting $v = \phi_i$ and

$$\partial U_h = \sum_{j=1}^{n_i} \alpha_j \phi_j$$

where α_j are the unknowns to be computed, thus equation (2.28) becomes

$$(\beta\nabla U_h^o,\, \nabla\phi_i) + \sum_{j=1}^{n_i} \alpha_j(\beta\nabla\phi_j,\, \nabla\phi_i) + ((\nabla U_h^o)^2,\, \phi_i) + \sum_{j=1}^{n_i} \alpha_j(2\nabla U_h^o\nabla\phi_j,\, \phi_i) = (1,\, \phi_i) \tag{2.29}$$

Equation (2.29) is an algebraic equation, which can be written as

$$\sum_{j=1}^{n_i} \alpha_j(\beta\nabla\phi_j,\, \nabla\phi_i) + \sum_{j=1}^{n_i} \alpha_j(2\nabla U_h^o\nabla\phi_j,\, \phi_i) = (1,\, \phi_i) - (\beta\nabla U_h^o,\, \nabla\phi_i) - ((\nabla U_h^o)^2,\, \phi_i) \tag{2.30}$$

Let

$$a(\phi_j, \phi_i) = \sum_{j=1}^{n_i} \alpha_j(\beta\nabla\phi_j,\, \nabla\phi_i) + \sum_{j=1}^{n_i} \alpha_j(2\nabla U_h^o\nabla\phi_j,\, \phi_i)$$

be the bilinear form and

$$L(\phi_i) = (1,\, \phi_i) - (\beta \nabla U_h^o,\, \nabla \phi_i) - ((\nabla U_h^o)^2,\, \phi_i)$$

be the linear form. The following Algorithm 1 which is adapted from [15] will be used to implement equation (2.30) in the software FEniCS-HPC. For more about FEniCS-HPC, see chapter 3.

Algorithm 1 Newton's Iteration

choose a starting guess $U_h^{(0)}$ and a tolerance value ϵ
for $k = 0,1,2,\ldots$ **do**
  Assemble $a^k(\phi_j, \phi_i)$ and $L^k(\phi_i)$
  solve the system $a^k \alpha^k = L^k$
  update $U^{k+1} = U_h^k + \partial U_h^k$ where $\partial U_h^k = \sum_{j=1}^{n_i} \alpha_j^k \phi_j$
  {check stopping criterion}
  if $||\partial U_h^k|| < \epsilon$ **then**
    Break
  end if
end for

2.4.2 Finite Element method for the force field equation

Equation (2.4) is solved by a pseudo-time stepping approach to a steady state solution.

$$\begin{aligned} \frac{\partial \mathbf{U}}{\partial t} &= \alpha \nabla^2 \mathbf{U} - (\mathbf{U} - \nabla f)|\nabla f|^2 \quad \text{in } \Omega \\ \mathbf{U} &= 0 \quad \text{on } \partial\Omega \end{aligned} \tag{2.31}$$

The method of discretization is done by semi-discrete approach. In this method, equation (2.31) is first discretized in space using a finite element method then a discretization in time. Multiplying equation (2.31) by a test function $\mathbf{v}$ (which is a vector, depending only on space and satisfies the Dirichlet boundary condition $\mathbf{v} = 0$ on $\partial\Omega$) and integrating, gives the weak form

$$(\frac{\partial \mathbf{U}}{\partial t}, \mathbf{v})_\Omega = (\alpha \nabla^2 \mathbf{U}, \mathbf{v})_\Omega - (\mathbf{U}, \mathbf{v}|\nabla f|^2)_\Omega + (\nabla f, \mathbf{v}|\nabla f|^2)_\Omega \tag{2.32}$$

Consider the first term on the right hand side of equation (2.32)

$$\begin{aligned} (\alpha \nabla^2 \mathbf{U}, \mathbf{v}) &= \int_\Omega \alpha \nabla^2 \mathbf{U} \cdot \mathbf{v}\, d\Omega \\ &= \alpha[(\nabla \mathbf{U} \cdot \mathbf{n}) \cdot \mathbf{v}|_{\partial\Omega} - \int_\Omega \nabla \mathbf{U} \cdot \nabla \mathbf{v} d\Omega]. \end{aligned} \tag{2.33}$$

Since there is homogeneous Dirichlet boundary conditions for $\mathbf{v}$, the first term of equation (2.33) disappears, and the equation becomes

$$\begin{aligned} (\alpha \nabla^2 \mathbf{U}, \mathbf{v})_\Omega &= -\alpha \int_\Omega \nabla \mathbf{U} \cdot \nabla \mathbf{v} d\Omega \\ &= -\alpha (\nabla \mathbf{U}, \nabla \mathbf{v})_\Omega \end{aligned} \tag{2.34}$$

Substituting equation (2.34) into equation (2.32) one gets

$$(\frac{\partial \mathbf{U}}{\partial t}, \mathbf{v})_\Omega = -\alpha (\nabla \mathbf{U}, \nabla \mathbf{v})_\Omega - (\mathbf{U}, \mathbf{v}|\nabla f|^2)_\Omega + (\nabla f, \mathbf{v}|\nabla f|^2)_\Omega. \tag{2.35}$$

For stability reasons, the backward Euler method is used for discretization in time of equation (2.35). Let $\mathbf{U}_k$, $\mathbf{U}_{k-1}$ and Δt denote the solution at the current time, the solution at the previous time and the time step (assume constant) respectively, then

$$\begin{aligned} (\frac{\partial \mathbf{U}}{\partial t}, \mathbf{v})_\Omega &= (\frac{\mathbf{U}_k - \mathbf{U}_{k-1}}{\Delta t}, \mathbf{v})_\Omega \\ &= \frac{1}{\Delta t}(\mathbf{U}_k, \mathbf{v})_\Omega - \frac{1}{\Delta t}(\mathbf{U}_{k-1}, \mathbf{v})_\Omega \end{aligned} \tag{2.36}$$

Using equation (2.36) in equation (2.35), yields the following

$$\frac{1}{\Delta t}(\mathbf{U}_k, \mathbf{v})_\Omega - \frac{1}{\Delta t}(\mathbf{U}_{k-1}, \mathbf{v})_\Omega = -\alpha (\nabla \mathbf{U}_k, \nabla \mathbf{v})_\Omega - (\mathbf{U}_k, \mathbf{v}|\nabla f|^2)_\Omega + (\nabla f, \mathbf{v}|\nabla f|^2)_\Omega \tag{2.37}$$

Multiplying both sides with Δt and putting all terms with $\mathbf{U}_k$ on the left side and terms without $\mathbf{U}_k$ on the right side gives

$$(\mathbf{U}_k, \mathbf{v})_\Omega + \Delta t \alpha (\nabla \mathbf{U}_k, \nabla \mathbf{v})_\Omega + \Delta t (\mathbf{U}_k, \mathbf{v}|\nabla f|^2)_\Omega = (\mathbf{U}_{k-1}, \mathbf{v})_\Omega + \Delta t (\nabla f, \mathbf{v}|\nabla f|^2)_\Omega \tag{2.38}$$

Equation (2.38) is implemented in the software FEniCS-hpc (see Chapter 3). The finite element method for equation (2.38) reads, find a $\mathbf{U}_k^h \in V_h$ such that

$$(\mathbf{U}_k^h, \mathbf{v}) + \Delta t\alpha(\nabla \mathbf{U}_k^h,\, \nabla \mathbf{v}) + \Delta t(\mathbf{U}_k^h, \mathbf{v}|\nabla f|^2) = (\mathbf{U}_{k-1}^h, \mathbf{v}) + \Delta t(\nabla f, \mathbf{v}|\nabla f|^2),\ \ \forall\, \mathbf{v} \in V_h. \tag{2.39}$$

Where V_h is a finite element space. With the bilinear and linear forms;

$$\begin{aligned} a_{GVF}(\mathbf{U}, \mathbf{v}) &= (\mathbf{U}_k^h, \mathbf{v}) + \Delta t\alpha(\nabla \mathbf{U}_k^h,\, \nabla \mathbf{v}) + \Delta t(\mathbf{U}_k^h, \mathbf{v}|\nabla f|^2) \\ L_{GVF}(\mathbf{v}) &= (\mathbf{U}_{k-1}^h, \mathbf{v}) + \Delta t(\nabla f, \mathbf{v}|\nabla f|^2) \end{aligned} \tag{2.40}$$

The abstract problem can be defined as, find $\mathbf{U}_k^h \in V_h$ such that $a_{GVF}(\mathbf{U}, \mathbf{v}) = L_{GVF}(\mathbf{v})$, $\forall \mathbf{v} \in V_h$. Both forms depend on f.

Stability analysis

To ensure that the computed solution of equation (2.32) is stable, we perform the following stability analysis. From equation (2.35), choosing $\mathbf{v} = \mathbf{U}$

$$(\frac{\partial \mathbf{U}}{\partial t}, \mathbf{U})_\Omega = -\alpha(\nabla \mathbf{U},\, \nabla \mathbf{U})_\Omega - (\mathbf{U}, \mathbf{U}|\nabla f|^2)_\Omega + (\nabla f, \mathbf{U}|\nabla f|^2)_\Omega \tag{2.41}$$

Using the definition of the L_2 norm , we have that $\|u\|^2 = \|u\|_{\mathrm{L}_2}^2 = (u,\, u)$ and that

$$\begin{aligned} (\frac{\partial \mathbf{U}}{\partial t}, \mathbf{U})_\Omega &= \int_\Omega \frac{\partial \mathbf{U}}{\partial t} \cdot \mathbf{U} d\Omega \\ (\text{but } \mathbf{U} \cdot \frac{\partial \mathbf{U}}{\partial t} &= \frac{1}{2}\frac{\partial \mathbf{U}^2}{\partial t}) \\ &= \int_\Omega \frac{1}{2}\frac{\partial \mathbf{U}^2}{\partial t} d\Omega \\ &= \frac{1}{2}\frac{\partial}{\partial t}\int_\Omega \mathbf{U}^2 d\Omega \\ &= \frac{1}{2}\frac{\partial}{\partial t}(\mathbf{U},\, \mathbf{U})_\Omega \\ &= \frac{1}{2}\frac{\partial}{\partial t}\|\mathbf{U}\|^2 \end{aligned} \tag{2.42}$$

Using equation (2.42) in equation (2.41)

$$\begin{aligned} \frac{1}{2}\frac{\partial}{\partial t}\|\mathbf{U}\|^2 &= -\alpha\|\nabla \mathbf{U}\|^2 - (\mathbf{U}, |\nabla f|^2 \mathbf{U})_\Omega + (|\nabla f|^2 \nabla f,\, \mathbf{U})_\Omega \\ \frac{1}{2}\frac{\partial}{\partial t}\|\mathbf{U}\|^2 &= -\alpha\|\nabla \mathbf{U}\|^2 - (|\nabla f|\mathbf{U}, |\nabla f|\mathbf{U})_\Omega + (|\nabla f|\nabla f,\, \mathbf{U}|\nabla f|)_\Omega \\ \int_0^T \frac{1}{2}\frac{\partial}{\partial t}\|\mathbf{U}\|^2 dt + \alpha\int_0^T \|\nabla \mathbf{U}\|^2 dt + \int_0^T \||\nabla f|\mathbf{U}\|^2 dt &= \int_0^T (|\nabla f|\nabla f,\, \mathbf{U}|\nabla f|)_\Omega dt \\ &= \int_0^T \int_\Omega |\nabla f|\, \nabla f\, \cdot \mathbf{U}\, |\nabla f|\, d\Omega dt \end{aligned} \tag{2.43}$$

Applying Cauchy's inequality ($\int g \cdot f dx \leq (\int |f|^2 dx)^{\frac{1}{2}} (\int |g|^2 dx)^{\frac{1}{2}}$) on the term on the right hand side of equation (2.43)

$$\int_0^T \int_\Omega |\nabla f| \nabla f \cdot \mathbf{U} |\nabla f| d\Omega dt \leq \left(\int_0^T \int_\Omega | \, |\nabla f| \nabla f|^2 d\Omega dt \right)^{\frac{1}{2}} \left(\int_0^T \int_\Omega | \, \mathbf{U} |\nabla f| \, |^2 d\Omega dt \right)^{\frac{1}{2}} \tag{2.44}$$

For $a, b, \epsilon \in \mathbb{R}^+$ then $ab \leq \frac{1}{2\epsilon} a^2 + \frac{\epsilon}{2} b^2$, for any $\epsilon > 0$. To prove this, use the fact that $0 \leq (a - \epsilon b)^2$ then

$$\begin{aligned} 0 &\leq a^2 - 2\epsilon ab + \epsilon^2 b^2 \\ 2\epsilon ab &\leq a^2 + \epsilon^2 b^2 \\ ab &\leq \frac{1}{2\epsilon} a^2 + \frac{\epsilon}{2} b^2, \end{aligned} \tag{2.45}$$

which is Young's inequality. Applying equation (2.45) on equation (2.44) gives

$$\begin{aligned} \int_0^T \int_\Omega |\nabla f| \nabla f \cdot \mathbf{U} |\nabla f| d\Omega dt &\leq \frac{1}{2\epsilon} \int_0^T \int_\Omega | \, |\nabla f| \nabla f \, |^2 d\Omega dt + \frac{\epsilon}{2} \int_0^T \int_\Omega |\mathbf{U}| \nabla f| \, |^2 d\Omega dt \\ \int_0^T \int_\Omega |\nabla f| \nabla f \cdot \mathbf{U} |\nabla f| d\Omega dt &\leq \frac{1}{2\epsilon} \int_0^T \| \, |\nabla f| \nabla f \, \|^2 \, dt + \frac{\epsilon}{2} \int_0^T \| \, |\nabla f| \mathbf{U} \, \|^2 \, dt. \end{aligned} \tag{2.46}$$

Equation (2.43) can now be written as

$$\begin{aligned} \int_0^T \frac{1}{2} \frac{\partial}{\partial t} \|\mathbf{U}\|^2 \, dt + \alpha \int_0^T \|\nabla \mathbf{U}\|^2 \, dt + \int_0^T \| \, |\nabla f| \mathbf{U}\|^2 \, dt \leq & \frac{1}{2\epsilon} \int_0^T \| |\nabla f| \nabla f\|^2 \, dt \\ & + \frac{\epsilon}{2} \int_0^T \| |\nabla f| \mathbf{U}\|^2 \, dt \end{aligned} \tag{2.47}$$

$$\frac{1}{2} \|\mathbf{U}(T)\|^2 + \alpha \int_0^T \|\nabla \mathbf{U}\|^2 \, dt + \int_0^T \| |\nabla f| \mathbf{U}\|^2 (1 - \frac{\epsilon}{2}) dt \leq \frac{1}{2\epsilon} \int_0^T \| \, |\nabla f| \, \nabla f\|^2 \, dt + \frac{1}{2} \|\mathbf{U}(0)\|^2$$

Choosing $0 < \epsilon < 2$ the term $\int_0^T \| \, |\nabla f| \mathbf{U} \, \|^2 (1 - \frac{\epsilon}{2}) dt$ will be a positive quantity. Thus, at the final time T the solution will always be less than the initial condition plus a positive finite quantity $\int_0^T \| \, |\nabla f| \nabla f \, \|^2 \, dt$ (from the definition of f, it is a bounded function).

Error estimation

The error estimation is formulated in terms of *a posteriori* error estimation based on duality. Equation (2.31) is known as the primal problem and can be written as

$$\begin{aligned} \frac{\partial \mathbf{U}}{\partial t} - \alpha \nabla^2 \mathbf{U} + \mathbf{U} |\nabla f|^2 &= \nabla f |\nabla f|^2 \\ (\frac{\partial}{\partial t} - \alpha \nabla^2 + |\nabla f|^2) \mathbf{U} &= \nabla f |\nabla f|^2 \\ \mathbf{A}\mathbf{U} &= g \end{aligned} \tag{2.48}$$

where $A = \frac{\partial}{\partial t} - \alpha \nabla^2 + |\nabla f|^2$ and $g = \nabla f |\nabla f|^2$.
Introducing the dual problem with dual solution ϕ continuous over the time interval $\mathbf{T} = [0, \, t_N]$

$$\begin{aligned} -\frac{\partial \phi}{\partial t} - \alpha \nabla^2 \phi + \phi |\nabla f|^2 = \psi \qquad & \text{in } \Omega \times \mathbf{T} \\ \phi = 0 \qquad & \text{on } \Gamma \times \mathbf{T}, \end{aligned} \tag{2.49}$$

with initial condition $\phi(., t_N) = 0$. Equation (2.49) can be written as

$$A^*\phi = \psi, \tag{2.50}$$

where $A^* = -\frac{\partial}{\partial t} - \alpha\nabla^2 + |\nabla f|^2$, and ψ is data.
The quantity (e, ψ) gives a measure of the error in the domain . If $\psi = 1$ then $(e, 1)$ becomes the measure of the average error over the whole domain. Let M be a function of e (where $e = \mathbf{U} - \mathbf{U}_h^k$, with $\mathbf{U}$ representing the exact solution of equation (2.48)). The estimation of M gives the bound on the error. To build an adaptive framework, M can be used to identify parts (elements) of the mesh that need refinement based on high errors that those parts produce.
For simplicity a two dimensional case is considered which can be extended to a three dimensional case. Let the time interval $\mathbf{T}$, be partitioned such that $0 = t_1 < t_2 < t_3 < \ldots < t_N = T$ and let $T_l = [t_{l-1}, t_l]$ be a representation of a sub-interval within the given time interval. Defined M as

$$\begin{aligned} M(e) &= \int_T (e, \psi) dt \\ &= \int (e, -\frac{\partial \phi}{\partial t} - \alpha\nabla^2\phi + \phi|\nabla f|^2) dt \\ &= \sum_{l=1}^{N} \int_{T_l}\int_\Omega -\frac{\partial \phi}{\partial t} e - \alpha\nabla^2\phi e + e\phi|\nabla f|^2 d\Omega dt \end{aligned} \tag{2.51}$$

But

$$\int_{T_l}\int_\Omega \frac{-\partial\phi}{\partial t} e d\Omega dt = \int_\Omega \left[-\phi e|_{T_l} + \int_{T_l} \phi\frac{\partial e}{\partial t} dt\right] d\Omega$$

$$\int_{T_l}\int_\Omega \frac{-\partial\phi}{\partial t} e d\Omega dt = \int_\Omega -\left[\phi e\right]_{l-1}^{l} d\Omega + \int_{T_l}\int_\Omega \phi\frac{\partial e}{\partial t} d\Omega dt$$

and

$$\int_{T_l}\int_\Omega -\alpha e\nabla^2\phi d\Omega dt = \int_{T_l}\int_\Omega \alpha\nabla e \cdot \nabla\phi d\Omega dt$$

Substituting the above expression in equation (2.51)

$$M(e) = \sum_{l=1}^{N} \int_{T_l}\int_\Omega \phi\frac{\partial e}{\partial t} + \alpha\nabla e \cdot \nabla\phi + e\phi|\nabla f|^2 d\Omega dt - \int_\Omega \left[\phi e\right]_{l-1}^{l} d\Omega, \tag{2.52}$$

where $\left[\phi e\right]_{l-1}^{l} = (\phi e)_l - (\phi e)_{l-1}$. That is the difference between ϕe evaluated at the boundary of each time sub-interval T_l. Applying the Galerkin orthogonality together with linear interpolation ($\pi\phi \in V_h$) in time and space

$$\begin{aligned} M(e) = &\sum_{l=1}^{N} \int_{T_l}\int_\Omega (\phi - \pi\phi)\frac{\partial e}{\partial t} + \alpha\nabla e\nabla(\phi - \pi\phi) \\ &+ e(\phi - \pi\phi)|\nabla f|^2 d\Omega dt + \int_\Omega \left[(\phi - \pi\phi)e\right]_{l-1}^{l} d\Omega \end{aligned} \tag{2.53}$$

Applying integration by parts on each element in Ω and using $e = \mathbf{U} - \mathbf{U}_h^k$, equation (2.53) becomes

$$\begin{aligned} M(e) =& \sum_{l=1}^{N}\sum_{i=1}^{m}\int_{T_l}\int_{\Omega_i}(\phi-\pi\phi)(\dot{\mathbf{U}}-\dot{\mathbf{U}}_h^k) - (\phi-\pi\phi)\alpha\nabla^2(\mathbf{U}-\mathbf{U}_h^k) \\ &+ (\mathbf{U}-\mathbf{U}_h^k)(\phi-\pi\phi)|\nabla f|^2 d\Omega dt + \int_{T_l}\left[(\phi-\pi\phi)\alpha\nabla(\mathbf{U}-\mathbf{U}_h^k)\right]_{\partial\Omega_i} dt \\ &+ \int_{\Omega}\left[(\phi-\pi\phi)(\mathbf{U}-\mathbf{U}_h^k)\right]_{l-1}^{l} d\Omega \end{aligned} \tag{2.54}$$

Also, let the contribution from the facets of the cells in the mesh be $J(\mathbf{U}_h^k)$, with $J(\mathbf{U}_h^k) = \int_{T_l}\left[(\phi-\pi\phi)\alpha\nabla(\mathbf{U}-\mathbf{U}_h^k)\right]_{\partial\Omega_i} dt$. Note that in one dimension, $J(\mathbf{U}_h^k) = 0$, since $\phi - \pi\phi = 0$, at the facets (nodes) but in two dimensions it is non zero. However we consider the diffusive term $\alpha\nabla^2\mathbf{U}$ to be of less importance, and to simplify the estimate we omit the $J(\mathbf{U}_h^k)$ term in the estimate. Thus equation (2.54) becomes

$$\begin{aligned} M(e) =& \sum_{l=1}^{N}\sum_{i=1}^{m}\int_{T_l}\int_{\Omega_i}(\phi-\pi\phi)\left[\dot{\mathbf{U}}-\dot{\mathbf{U}}_h^k - \alpha\nabla^2\mathbf{U} + \alpha\nabla^2\mathbf{U}_h^k - (\mathbf{U}-\mathbf{U}_h^k)|\nabla f|^2\right] d\Omega dt \\ &+ \int_{\Omega}\left[(\phi-\pi\phi)(\mathbf{U}-\mathbf{U}_h^k)\right]_{l-1}^{l} d\Omega \end{aligned}$$

$$\begin{aligned} M(e) =& \sum_{l=1}^{N}\sum_{i=1}^{m}\int_{T_l}\int_{\Omega_i}(\phi-\pi\phi)\left[\dot{\mathbf{U}} - \alpha\nabla^2\mathbf{U} + \mathbf{U}|\nabla f|^2 - (\alpha\dot{\mathbf{U}}_h^k - \nabla^2\mathbf{U}_h^k + \mathbf{U}_h^k|\nabla f|^2)\right] d\Omega dt \\ &+ \int_{\Omega}\left[(\phi-\pi\phi)(\mathbf{U}-\mathbf{U}_h^k)\right]_{l-1}^{l} d\Omega \end{aligned}$$

Using equation (2.48)

$$\begin{aligned} M(e) =& \sum_{l=1}^{N}\sum_{i=1}^{m}\int_{T_l}\int_{\Omega_i}(\phi-\pi\phi)\left[g - (\alpha\dot{\mathbf{U}}_h^k - \nabla^2\mathbf{U}_h^k + \mathbf{U}_h^k|\nabla f|^2)\right] d\Omega dt \\ &+ \int_{\Omega}\left[(\phi-\pi\phi)(\mathbf{U}-\mathbf{U}_h^k)\right]_{l-1}^{l} d\Omega \end{aligned}$$

Also the residual is given by $R(\mathbf{U}_h^k) = g - (\alpha\dot{\mathbf{U}}_h^k - \nabla^2\mathbf{U}_h^k + \mathbf{U}_h^k|\nabla f|^2)$

$$\begin{aligned} M(e) =& \sum_{l=1}^{N}\sum_{i=1}^{m}\int_{T_l}\int_{\Omega_i}(\phi-\pi\phi)R(\mathbf{U}_h^k) d\Omega dt \\ &+ \int_{\Omega}\left[(\phi-\pi\phi)(\mathbf{U}-\mathbf{U}_h^k)\right]_{l-1}^{l} d\Omega \end{aligned} \tag{2.55}$$

Since $\mathbf{U}$ is a continuous function in time, we have $[\mathbf{U}]_{l-1}^{l} = 0$,
therefore $\left[\mathbf{U}-\mathbf{U}_h^k\right]_{l-1}^{l} = \left[-\mathbf{U}_h^k\right]_{l-1}^{l}$.
Applying Cauchy-Schwarz inequality on equation (2.55)

$$\begin{aligned} M(e) \leq& C\sum_{l=1}^{m}\sum_{i=1}^{m}\left\|R(\mathbf{U}_h^k)\right\|_{L^2(J_{il})}\left\|\phi-\pi\phi\right\|_{L^2(J_{il})} \\ &+ \left\|\left[\mathbf{U}_h^k\right]_{l-1}^{l}\right\|_{L^2(\Omega)}\left\|(\phi-\pi\phi)_{l-1}^{l}\right\|_{L^2(\Omega)} \end{aligned} \tag{2.56}$$

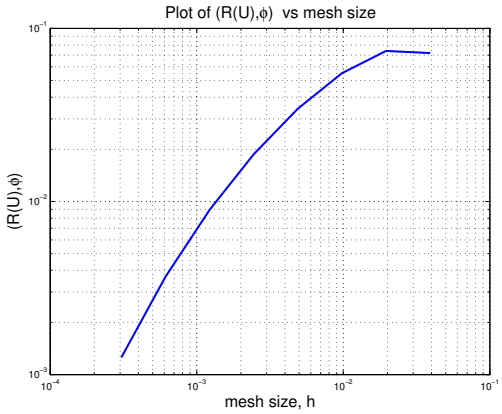
Where $J_{il} = \Omega_i \times T_l$ is a space time element. It can be shown that (see [6, 15])

$$\|\phi - \pi\phi\|_{L^2(J_{il})} \quad \leq \quad C(k_l \left\|\dot{\phi}\right\|_{L^2(J_{il})} + h_i^2 \left\|\nabla^2\phi\right\|_{L^2(J_{il})}) \tag{2.57}$$

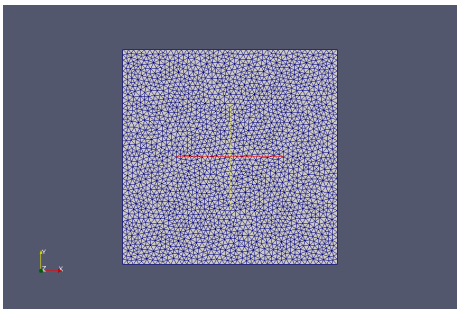
Applying equation (2.57) to equation (2.56), the following is obtained

$$\begin{aligned} M(e) \leq C \sum_{l=1}^{N} \sum_{i=1}^{m} & \left\|R(\mathbf{U}_h^k)\right\|_{L^2(J_{il})} \left(k_l \left\|\dot{\phi}\right\|_{L^2(J_{il})} + h_i^2 \left\|\nabla^2\phi\right\|_{L^2(J_{il})}\right) \\ & + \left\|\left[\mathbf{U}_h^k\right]_{l-1}^{l}\right\|_{L^2(\Omega_i)} h_i^2 \left\|\left[(\phi)''\right]_{l-1}^{l}\right\|_{L^2(\Omega_i)}, \end{aligned} \tag{2.58}$$

where $k_l = t_l - t_{l-1}$.

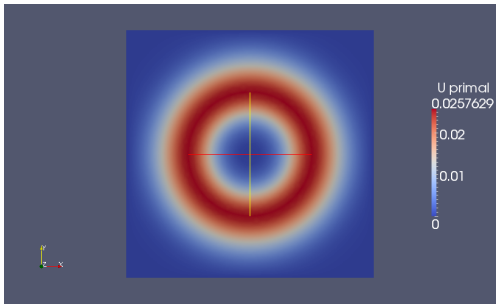


(a) *Error estimation with respect to mesh size.*

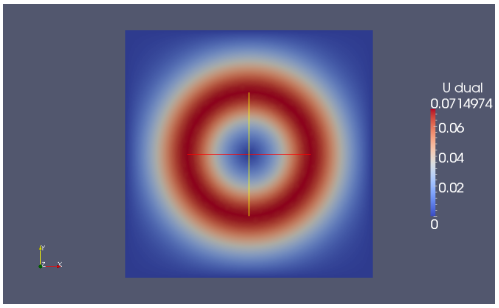


(b) *Square mesh with number of vertices,* $v = 2215$*, made up off triangular elements.*

Figure 2.1: *A plot of* $(R(U), \phi)$ *against mesh size* h *in Figure (2.1a). The mesh used to compute the error estimate is shown in Figure (2.1b).*



(a) *Computed primal solution*



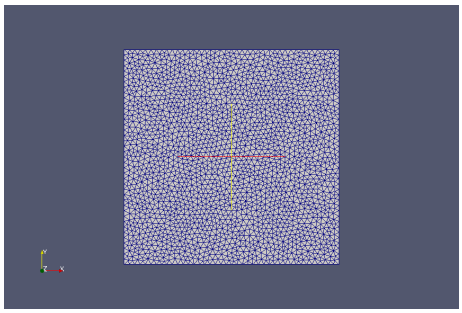
(b) *Computed dual solution*

Figure 2.2: *Figure shows the primal computed solution and the dual computed solution in a square domain.*

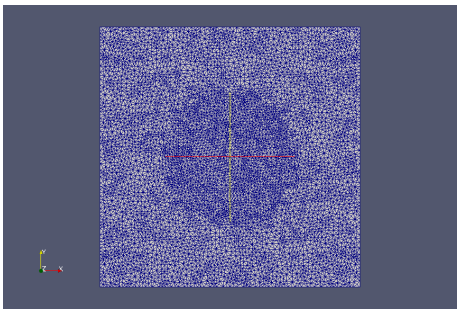
To illustrate the *a posteriori* error estimate, the dual solution ϕ, is computed from equation (2.49) using a finite element method with quadratic basis functions and the primal solution is computed with linear basis function as shown in Figure (2.2b) and Figure (2.2a). f was chosen as

$$f(x,y) = \frac{1}{\bar{\sigma}\sqrt{2\pi}} \exp - \left(\frac{0.3 - \sqrt{x^2 + y^2}}{2\bar{\sigma}^2} \right),$$

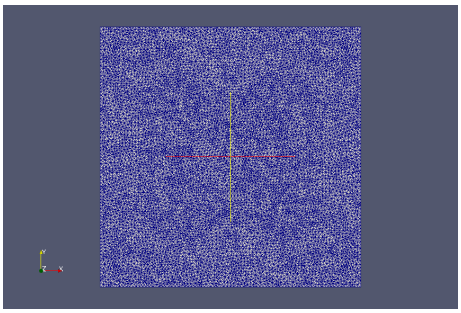
with $\bar{\sigma} = 0.5$ and $\alpha = 0.2$ on a square domain as shown in Figure (2.1b). The reason for choosing different basis functions for the dual solution and primal solution is to avoid having a zero a posteriori error estimate. From Figure (2.1), $(R(\mathbf{U}_h^k), \phi)$ decreases as the mesh is uniformly refined . To test the concept of adaptive refinement, the cells of the mesh where the error is bigger than a tolerance value set as 0.001 were located and refined. In this case, refinements were done locally. Figure (2.3) shows how some portions of the mesh were refined when the error $(R(\mathbf{U}_h^k), \phi)$ was bigger than 0.001.



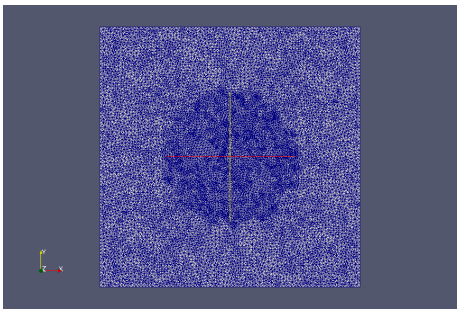
(a) *number of vertices,* $v = 2215$



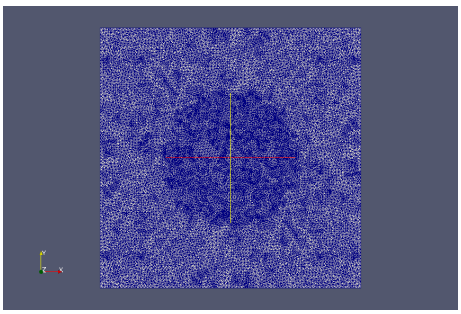
(b) *number of vertices ,* $v = 8697$



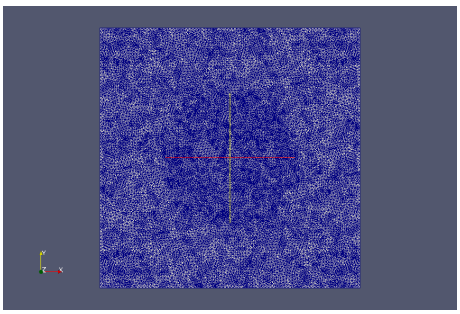
(c) *number of vertices,* $v = 34465$



(d) *number of vertices* $v = 137217$



(e) *number of vertices,* $v = 547585$



(f) *number of vertices,* $v = 2187780$

Figure 2.3: *The figure shows a square mesh with successive refinements.*

2.4.3 Finite Element method for the elasticity equation

To maintain the same time stepping scheme with the external field computed above, equation (2.14) has to be rewritten as a coupled system. Using the variable $\mathbf{u}$ which represents the displacement velocity as before, that is $\mathbf{u} = \frac{d\mathbf{w}}{dt}$, equation (2.14) with its given boundary and initial conditions as before, becomes

$$\dot{\mathbf{u}} = \nabla \cdot \sigma + \mathbf{U} \tag{2.59a}$$

$$\frac{d\mathbf{w}}{dt} = \mathbf{u} \tag{2.59b}$$

The velocity is first computed using equation (2.59a) and then the displacement is computed using equation (2.59b) with the known velocity at each time step. That is equation (2.59b) is discretized as follows;

$$\begin{aligned} \mathbf{w}_n - \mathbf{w}_{n-1} &= \Delta t \mathbf{u}_n \\ \mathbf{w}_n &= \mathbf{w}_{n-1} + \Delta t \mathbf{u}_n, \end{aligned} \tag{2.60}$$

where $\mathbf{u}_n$ is the finite element solution obtained from solving equation (2.59a), $\mathbf{w}_n$ and $\mathbf{w}_{n-1}$ are the current and previous displacements.
The semi-discrete approach on equation (2.59a) is as follows;
Multiplying equation (2.59) by a test function $\mathbf{v}$, with $\mathbf{v} \in V_h$, depending on space only, and integrating,

$$(\dot{\mathbf{u}},\, \mathbf{v})_\Omega = (\nabla \cdot \sigma,\ \ \mathbf{v})_\Omega + (\mathbf{U},\, \mathbf{v})_\Omega \tag{2.61}$$

Note that

$$\nabla \cdot \sigma = \sum_{i,\, j=1}^{3} \frac{\partial \sigma_{ij}}{\partial x_j}$$

Substituting in equation (2.61), it becomes

$$(\dot{\mathbf{u}},\ \ \mathbf{v})_\Omega = (\sum_{i,\, j=1}^{3} \frac{\partial \sigma_{ij}}{\partial x_j},\ \ \mathbf{v})_\Omega + (\mathbf{U},\ \ \mathbf{v})_\Omega \tag{2.62}$$

Applying integration by parts on the first term on the right hand side

$$\begin{aligned} (\sum_{i,\, j=1}^{3} \frac{\partial \sigma_{ij}}{\partial x_j},\ \ \mathbf{v}_i)_\Omega &= \left(\int \sum_{i\, j=1}^{3} \frac{\partial \sigma_{ij}}{\partial x_j} d\Omega \right)_{\partial\Omega} \cdot \mathbf{v}_i|_{\partial\Omega} - \int_\Omega \left(\int \sum_{i,\, j=1}^{3} \frac{\partial \sigma_{ij}}{\partial x_j} d\Omega \right) \frac{\partial \mathbf{v}_i}{\partial x_j} d\Omega \\ &= \int_{\partial\Omega} \sum_{i,\, j=1}^{3} \sigma_{ij} \cdot n_j\, \mathbf{v}_i \, d\Gamma - \int_\Omega \sum_{i,\, j=1}^{3} \sigma_{ij} \frac{\partial \mathbf{v}_i}{\partial x_j} d\Omega \\ &= \sum_{i,\, j=1}^{3} (\sigma_{ij} \cdot n_j\, ,\, v_i)_{\partial\Omega} - \sum_{i,\, j=1}^{3} (\sigma_{ij},\, \frac{\partial v_i}{\partial x_j})_\Omega \end{aligned} \tag{2.63}$$

Substituting equation (2.63) in equation (2.62)

$$(\dot{\mathbf{u}},\ \ \mathbf{v})_\Omega = \sum_{i,\, j=1}^{3} (\sigma_{ij} \cdot n_j,\, v_i)_{\partial\Omega} - \sum_{i,\, j=1}^{3} (\sigma_{ij},\, \frac{\partial \mathbf{v}_i}{\partial x_j})_\Omega + (\mathbf{U},\ \ \mathbf{v})_\Omega \tag{2.64}$$

Reverting to the notation without indices, equation (2.64) becomes

$$(\dot{\mathbf{u}},\ \mathbf{v})_\Omega = (\sigma \cdot \mathbf{n}\,,\, \mathbf{v})_{\partial\Omega} - (\sigma,\, \nabla\mathbf{v})_\Omega + (\mathbf{U},\ \mathbf{v})_\Omega \tag{2.65}$$

Since σ and $\nabla\mathbf{v}$ are matrices the contraction property which is defined as; Given any two 3×3 matrices A and B then

$$A : B = \sum_{i,\,j=1}^{3} A_{ij}B_{ij} \tag{2.66}$$

can be used on σ and $\nabla\mathbf{v}$. Thus

$$\begin{aligned} (\sigma,\, \nabla\mathbf{v})_\Omega &= \int_\Omega \sum_{i,\,j=1}^{3} \sigma_{ij}\mathbf{v}_{i,\,j}\, d\Omega \\ \text{where } \nabla\mathbf{v} &= \mathbf{v}_{i,\,j} \\ &= \int_\Omega \sigma \colon \nabla\mathbf{v}\, d\Omega \\ &= (\sigma \colon \nabla\mathbf{v})_\Omega \end{aligned} \tag{2.67}$$

Substituting equation (2.67) in equation (2.65)

$$(\dot{\mathbf{u}},\ \mathbf{v})_\Omega = (\sigma \cdot \mathbf{n}\,,\, \mathbf{v})_{\partial\Omega} - (\sigma : \nabla\mathbf{v})_\Omega + (\mathbf{U},\ \mathbf{v})_\Omega \tag{2.68}$$

Equation (2.68) becomes

$$(\frac{\partial\mathbf{u}}{\partial t},\ \mathbf{v})_\Omega = (\sigma \cdot \mathbf{n},\, \mathbf{v})_{\partial\Omega} - (\sigma : \nabla\mathbf{v})_\Omega + (\mathbf{U},\ \mathbf{v})_\Omega \tag{2.69}$$

Let $\mathbf{u}^h$ be the finite element solution, then the finite element method for equation (2.69) reads;
Find $\mathbf{u}^h \in V_h$ such that

$$(\frac{\partial\mathbf{u}^h}{\partial t},\ \mathbf{v})_\Omega = (\sigma \cdot \mathbf{n},\, \mathbf{v})_{\partial\Omega} - (\sigma : \nabla\mathbf{v})_\Omega + (\mathbf{U},\ \mathbf{v})_\Omega,\ \forall\mathbf{v} \in V_h \tag{2.70}$$

Equation (2.70) is discretized in time using backward Euler and given that it is a non linear equation, for each time step the Newton's method is applied to the spatial part to obtain the solution. To illustrate the method, the notation $D_{x_j} f = \frac{\partial f}{\partial x_j}$ will be used [10].

Define F to be

$$F(\mathbf{u}^h) := -(\frac{\partial\mathbf{u}^h}{\partial t},\ \mathbf{v})_\Omega + (\sigma \cdot \mathbf{n}\,, \mathbf{v})_{\partial\Omega} - (\sigma :\ \nabla\mathbf{v})_\Omega + (\mathbf{U},\ \mathbf{v})_\Omega \tag{2.71}$$

and also let

$$D_t\mathbf{u}^h = -(\frac{\partial\mathbf{u}^h}{\partial t},\ \mathbf{v})_\Omega$$

and

$$G(\mathbf{u}^h) = (\sigma \cdot \mathbf{n}\,,\, \mathbf{v})_{\partial\Omega} - (\sigma \colon\ \nabla\mathbf{v})_\Omega + (\mathbf{U},\ \mathbf{v})_\Omega$$

Observe that G contains differential operators in space only. Then equation (2.71) becomes

$$F(\mathbf{u}^h) = -D_t\mathbf{u}^h + G(\mathbf{u}^h) \tag{2.72}$$

Generally Newton's method is given by

$$\mathbf{u}_n^h = \mathbf{u}_{n-1}^h - (F'(\mathbf{u}_{n-1}^h))^{-1} F(\mathbf{u}_{n-1}^h), \tag{2.73}$$

where $F'(\mathbf{u}_{n-1}^h)$ is a square matrix with elements that are partial derivatives of the components of F. To derive the Newton's method for equation (2.72), first apply a differential operator, $D_{\mathbf{u}^h}$ on both sides of equation (2.72)

$$\begin{aligned} D_{\mathbf{u}^h}(F(\mathbf{u}^h)) &= -D_{\mathbf{u}^h}(D_t\mathbf{u}^h + G(\mathbf{u}^h)) && (2.74)\\ &= -D_{\mathbf{u}^h}(D_t\mathbf{u}^h) + D_{\mathbf{u}^h}(G(\mathbf{u}^h)) && (2.75)\\ &\quad D_{\mathbf{u}^h} \text{ and } D_t \text{ commute} \\ &= -D_t(D_{\mathbf{u}^h}\mathbf{u}^h) + D_{\mathbf{u}^h}(G(\mathbf{u}^h)) && (2.76) \end{aligned}$$

Multiply both sides of equation (2.76) with a function W

$$\begin{aligned} D_{\mathbf{u}^h}(F(\mathbf{u}^h))W &= -D_t(D_{\mathbf{u}^h}\mathbf{u}^h)W + D_{\mathbf{u}^h}(G(\mathbf{u}^h))W && (2.77)\\ &\quad \text{but } D_{\mathbf{u}^h}\mathbf{u}^h = 1 \\ &\quad \text{and } D_{\mathbf{u}^h}(G(\mathbf{u}^h)) = G'(\mathbf{u}^h) \\ D_{\mathbf{u}^h}(F(\mathbf{u}^h))W &= -D_tW + G'(\mathbf{u}^h)W && (2.78)\\ F'(\mathbf{u}^h)W &= -D_tW + G'(\mathbf{u}^h)W && (2.79) \end{aligned}$$

Let $W = \mathbf{u}_n^h - \mathbf{u}_{n-1}^h$, that is the difference between the current and previous solution to equation (2.73). Substituting it in equation (2.79)

$$\begin{aligned} F'(\mathbf{u}_{n-1}^h)(\mathbf{u}_n^h - \mathbf{u}_{n-1}^h) &= -D_t(\mathbf{u}_n^h - \mathbf{u}_{n-1}^h) + G'(\mathbf{u}_{n-1}^h)(\mathbf{u}_n^h - \mathbf{u}_{n-1}^h) \\ &= -D_t\mathbf{u}_n^h + D_t\mathbf{u}_{n-1}^h + G'(\mathbf{u}_{n-1}^h)(\mathbf{u}_n^h - \mathbf{u}_{n-1}^h) \\ F'(\mathbf{u}_{n-1}^h)(\mathbf{u}_n^h - \mathbf{u}_{n-1}^h) - D_t\mathbf{u}_{n-1}^h &= -D_t\mathbf{u}_n^h + G'(\mathbf{u}_{n-1}^h)(\mathbf{u}_n^h - \mathbf{u}_{n-1}^h) && (2.80) \end{aligned}$$

But from equation (2.72),

$$\begin{aligned} F(\mathbf{u}_{n-1}^h) &= -D_t\mathbf{u}_{n-1}^h + G(\mathbf{u}_{n-1}^h) \\ \Rightarrow \quad D_t\mathbf{u}_{n-1}^h &= -F(\mathbf{u}_{n-1}^h) + G(\mathbf{u}_{n-1}^h) \end{aligned}$$

Substitute in equation (2.80)

$$F'(\mathbf{u}_{n-1}^h)(\mathbf{u}_n^h - \mathbf{u}_{n-1}^h) + F(\mathbf{u}_{n-1}^h) - G(\mathbf{u}_{n-1}^h) = \quad -D_t\mathbf{u}_n^h + G'(\mathbf{u}_{n-1}^h)(\mathbf{u}_n^h - \mathbf{u}_{n-1}^h) \tag{2.81}$$

$$F'(\mathbf{u}_{n-1}^h)\left[\mathbf{u}_n^h - \mathbf{u}_{n-1}^h + \frac{F(\mathbf{u}_{n-1}^h)}{F'(\mathbf{u}_{n-1}^h)}\right] - G(\mathbf{u}_{n-1}^h) = \quad -D_t\mathbf{u}_n^h + G'(\mathbf{u}_{n-1}^h)(\mathbf{u}_n^h - \mathbf{u}_{n-1}^h) \tag{2.82}$$

The first term on the left hand side of equation (2.82) is zero inferred from equation (2.73). Therefore the Newton's method for equation (2.72) is

$$-D_t\mathbf{u}_n^h + G'(\mathbf{u}_{n-1}^h)(\mathbf{u}_n^h - \mathbf{u}_{n-1}^h) = -G(\mathbf{u}_{n-1}^h), \quad \text{for} \quad n = 1, 2, 3.. \tag{2.83}$$

Writing equation (2.83) concisely as

$$D_t\mathbf{u}_n^h = \left[\mathbf{u}_n^h - \mathbf{u}_{n-1}^h + \frac{G(\mathbf{u}_{n-1}^h)}{G'(\mathbf{u}_{n-1}^h)}\right] G'(\mathbf{u}_{n-1}^h) \quad \text{for} \quad n = 1, 2, 3.. \tag{2.84}$$

From equation (2.84), it can be seen that for each time step a non linear algebraic system is solved. This equation is implemented in Unicorn in a class known as the ElasticSmoother. For more about Unicorn see the next chapter and [17].
At this point the elastic object can now be deformed since the external field can be computed from equation (2.39) and the deformation is described by equation (2.84). Algorithm 2 presents a holistic approach of this deformation.

Algorithm 2 Model deformation algorithm

1: Let the solution from Algorithm 1 be U_{EK}
2: Define the bilinear and linear Forms from equation (2.40) to depend on U_{EK}, dt, U_{old}, where U_{old} is the solution at the previous time step, on a mesh.
3: Let these Forms be $a(U_{EK}, dt, \alpha)$ and $L(U_{old}, dt, U_{old})$
4: Assemble $a(U_{EK}, dt, \alpha)$ and $L(U_{old}, dt, U_{old})$
5: Declare **ElasticSmoother** class which contains the implementation of equation 2.84
6: **while** $t < T$ **do**
7: Assemble $a(U_{EK}, dt, \alpha)$ into a matrix A
8: Assemble $L(U_{old}, dt, U_{old})$ into a vector b
9: Apply boundary conditions
10: Solve the system $AU_{ext} = b$ for U_{ext}
11: Set $U_{old} = U_{ext}$
12: Apply ElasticSmoother which depends on the external field U_{ext}
13: Check for convergence by examining the change in volume (area) of the mesh
14: **if** no change in volume (area) **then**
15: break
16: **end if**
17: $t+ = timestep$
18: **end while**

Chapter 3

Software

The software used in implementing the equations derived in this project is Dolfin-hpc [13, 16] and Unicorn [9] which are part of the FEniCS-hpc project. Dolfin-hpc is a high performance computing branch of Dolfin, consisting of several C++/Python library functions (classes) which have been parallelized. The parallelization is carried out with the following; MPI, OpenMP, PGAS. Thus only a small amount of effort is needed to parallelize a new low level algorithm which involves looping over a mesh, and the algorithms developed in this project are executed in parallel. For example, reading a mesh in a program, the mesh is automatically distributed on the available number of cores. Each core has a separate part of the mesh and the meshes on each of these cores are glued together using a class known as **MeshDistributedData**. There is also a mapping from local to global indices and vice versa making it easy to keep track of the vertices and cells of the whole mesh. Some classes used in this project will be mentioned. One of those classes is the **Function** class. Functions are defined by a mesh and degrees of freedoms. Since the mesh is distributed across the cores, so is any function defined with the function class. Also, since partial differential equations are used as building blocks for this project, a gain in parallel performance is achieved. Another class is the **MeshEditor**. It is used for creating a simplicial mesh (meshes consisting of intervals, triangles or tetrahedra) by specifying the vertices and the cells.

FFC (FEniCS Form Compiler)
FFC is another component of FEniCS-hpc which translates a high level representation of the weak form into low-level source code. As an example consider the weak form of equation (2.39), its implementation in a form file is as shown in listing 3.1.

Listing 3.1: *Source code for bilinear and linear forms for one time step with Newton's method of equation (2.39)*

```
cell = "triangle"
element = FiniteElement("Lagrange", cell, 1)

K3 = FiniteElement("Discontinuous Lagrange", cell, 0)
element1 = VectorElement("Lagrange", cell,1)

v = TestFunction(element1)
u = TrialFunction(element1)
f = Function(element)

u_old = Function(element1) #solution computed at the previous time step

alpha = Function(K3)
dt = Function(K3)

T = dot(grad(f),grad(f))

def A(u,v):
    return dot(grad(u),grad(v))

L = dt*dot(mult(T, grad(f)),v)*dx  + dot(u_old,v)*dx
a = dt*alpha*A(u,v)*dx + dt*dot(mult(T, u),v)*dx +  dot(u,v)*dx
```

Unicorn: Unicorn is made up of a collection of finite element solver implementations for continuum mechanics and models. Amongst them is the ElasticSmoother, already mentioned in Chapter 2. In the ElasticSmoother class, its implementation is based on deforming a mesh towards a weighted optimal quality. The highest quality that a cell can have is set to one, which corresponds to a scaled equilateral reference cell. By searching for the best quality for cells in the mesh, a time-dependent non-linear elasticity problem is solved. For cells with a bad quality, an extra stiffness is attached to it.

Chapter 4

Results

4.1 Introduction

To validate the methods developed in Chapter 2, a sphere is used as a test case to be deformed to an ellipsoid, that is the ellipsoid is the target geometry. The sphere is in the form of a volume mesh with 18596 tetrahedrons, 3827 vertices, radius of 1.0 and a computational volume of 4.16916. Computational volume means that we sum all the individual volumes of the tetrahedrons in the mesh. The axis of the ellipsoid are chosen as $a = 0.7$, $b = 0.8$, $c = 0.6$ to observed a significant displacement of the points in the mesh. The volume of the ellipsoid (target geometry) is 1.41. A force field is computed from the target geometry. The vertices at the boundary of the mesh (sphere) are allowed to move towards the target (ellipsoid) boundary. For this test case, two implementations of the elasticity PDE are tested. These implementations are; when extra stiffness is assigned to cells with poor quality making them very stiff and when no extra stiffness is assigned, but there is a uniform stiffness across the cells. The parameters E and ν were taken as 1.0 and 0.0 respectively. We observe effects of a parameter p, which is a power of the cell quality (see equation(4.2)), in the deformation for both implementations of the PDE. In this project, we chose p = 0, for the case of uniform stiffness and p = 2 when extra stiffness is assigned to cells with poor quality. The visualization of the results is with Paraview [1]. The results were performed using 12 cores on a NUMA (Non Uniform Memory Access) machine (Hydra) at KTH.

4.2 Results

The implementation of equation (2.39) in FEniCS-hpc is with an analytical function given by

$$f(x,y,z) = \frac{1}{\bar{\sigma}\sqrt{2\pi}} \exp\left(-\frac{1 - \left(\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} \right)}{2\bar{\sigma}^2} \right), \tag{4.1}$$

where $a = 0.7$, $b = 0.8$, $c = 0.6$, $\bar{\sigma} = 0.5$ and $\alpha = 0.2$. Using a sphere with radius $r = 1$ and the target geometry as an ellipsoid with dimensions given as a, b, and c as above, the computed solution of equation (2.39) is shown in Figure 4.1. From the figure, it can be seen that the properties of the gradient are

- ∇f has vectors pointing towards the boundary and are normal to the boundary at the boundaries. This will cause the model's boundary undergoing deformation to converge to the target boundary.

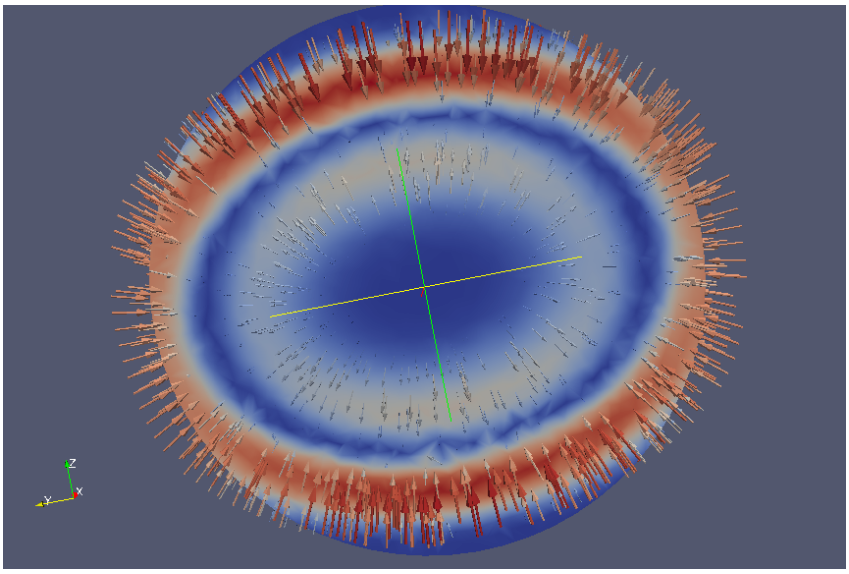


Figure 4.1: *Computed solution of equation (2.39) before deformation. The figure shows that the gradients are highest at the edge in the domain and near zero at the centre of the domain. The domain is a spherical mesh with tetrahedral elements and vertices=3827*

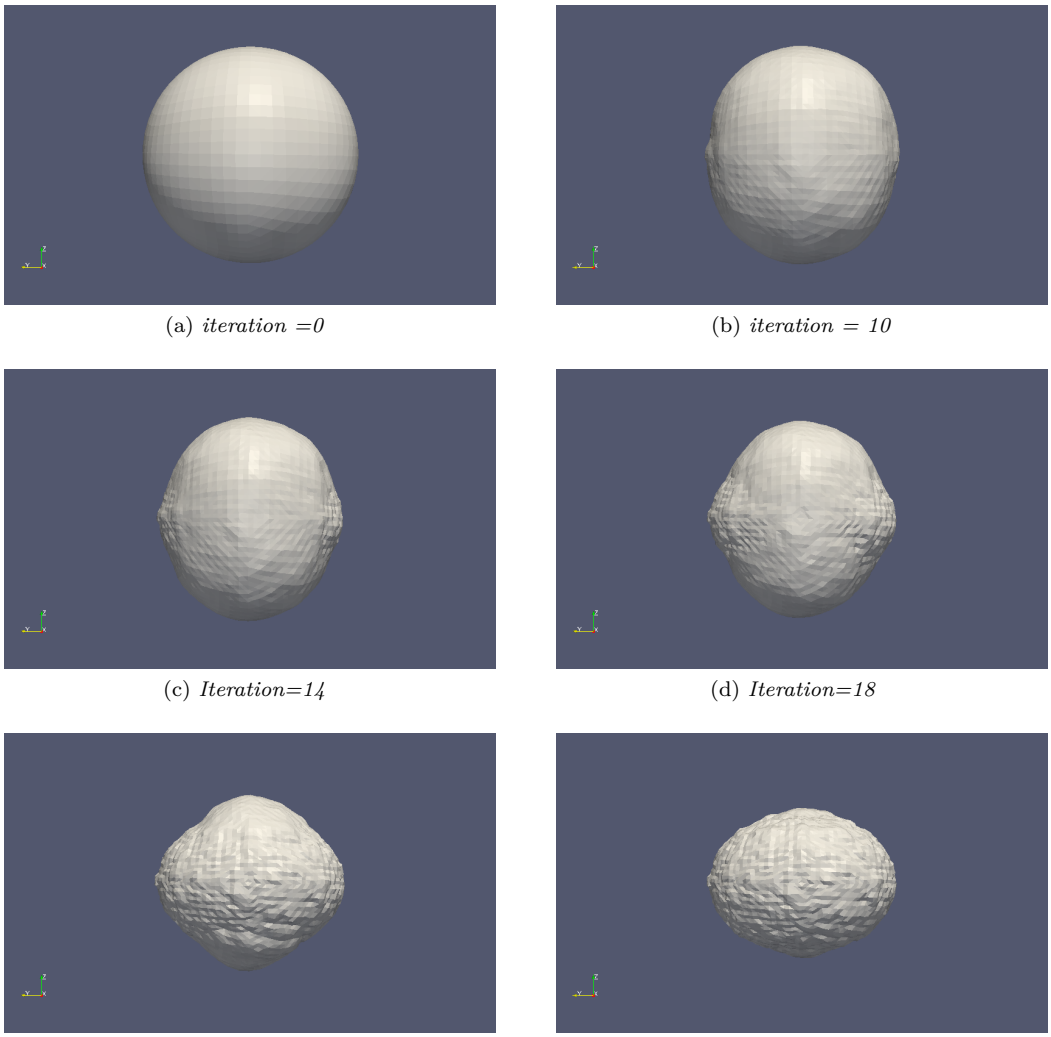
- ∇f has large magnitudes only in the immediate vicinity of the boundary.
- In a homogeneous region (for example, the inner part of the left ventricle), ∇f is almost zero.

The deformation is first implemented with the case when the elasticity PDE assigns extra stiffness to cells with poor quality. A weighted stiffness is given to each cell in the mesh, in the form

$$\text{weight} = \frac{1}{Q^p} \tag{4.2}$$

where Q is the quality and p is a coefficient. When the quality Q reduces, the stiffness for that cell increases. The inequality $0 < Q \leq 1$ holds for each cell.
Figure (4.2) shows how the sphere deforms to an ellipsoid over the number of iterations shown. As the iteration progresses, the sphere shrinks until equilibrium is reached. At equilibrium, a stopping criterion is satisfied. The stopping criteria compares the current volume of the mesh and the previous volume of the mesh. If the comparison is less than a given tolerance (of 10^{-4}) then the deformation stops. The volume of the mesh is calculated by summing the volume of each cell in the mesh. Since the execution is done in parallel, each core has just a portion of the mesh and computes a partial volume. At each iteration step, the total volume of the complete mesh is calculated by gathering (MPI_Allgather) the individual volume that each core calculates. This total volume is what is used in the comparison of previous and current volume, not the local volume on each core. This prevents some cores from calling MPI_Finalize when their local current volume compared with their previous local volume is less than the given tolerance, while other cores are still computing.

The roughness in Figure (4.2) is explained by the fact that the cells that have been deformed, some turn to have poor quality and thus extra stiffness is assigned to them resulting in no



(a) *iteration =0*

(b) *iteration = 10*

(c) *Iteration=14*

(d) *Iteration=18*

(e) *Iteration=26*

(f) *Iteration=34*

Figure 4.2: *A sphere undergoing deformations in number of steps shown into an ellipsoid with* p = 2

movement of the vertices associated to the cell while the other cells with higher quality, their vertices can move. This is the observed effect of the parameter, p (p = 2). To smooth out the ellipsoid, a Laplacian mesh smoother is used. The Laplacian mesh smoother in this case iterates over all the nodes of the mesh and moves each node to the geometric center of its neighbours. The smoothing is carried out with a small number of iteration. A high number of iterations with Laplacian smoothing will change the geometry of the mesh. That is, the mesh will become elongated. With the addition of a smoothing routine (Laplacian smoothing which results in Figure (4.4)), the result is not as good as compared to Figure (4.5), where p = 0 is used in equation (4.2). This case correspond to the case when the PDE uses uniform stiffness.

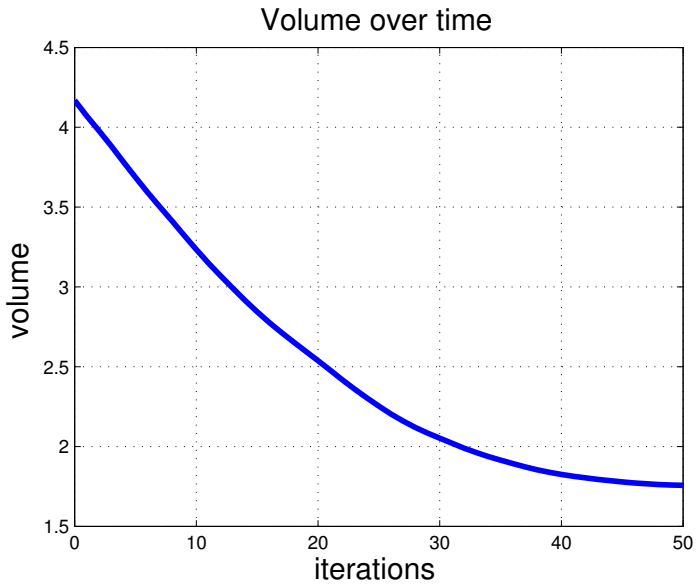


Figure 4.3: *A plot showing how the size of the volume of the sphere changes in each time step. This figure represents the volume in figure 4.2 with a time step of* 0.0001 *with* p = 2. *The target volume is 1.41.*

A plot of volume over number of iterations for p = 0 and p = 2 is shown in Figure (4.6) and Figure (4.3) respectively. From these figures, the deformation for p = 0 does somewhat better than that for p = 2. The case when p = 2 performs poorer, is due to the fact that at some point during the deformation, some cells at the surface will end up with low quality and high stiffness making their vertices non movable. The next iteration will lead to more cells having low quality on the surface increasing the number of non movable vertices. This leads to an early attainment of the stopping criterion as compared to the case when p = 0.

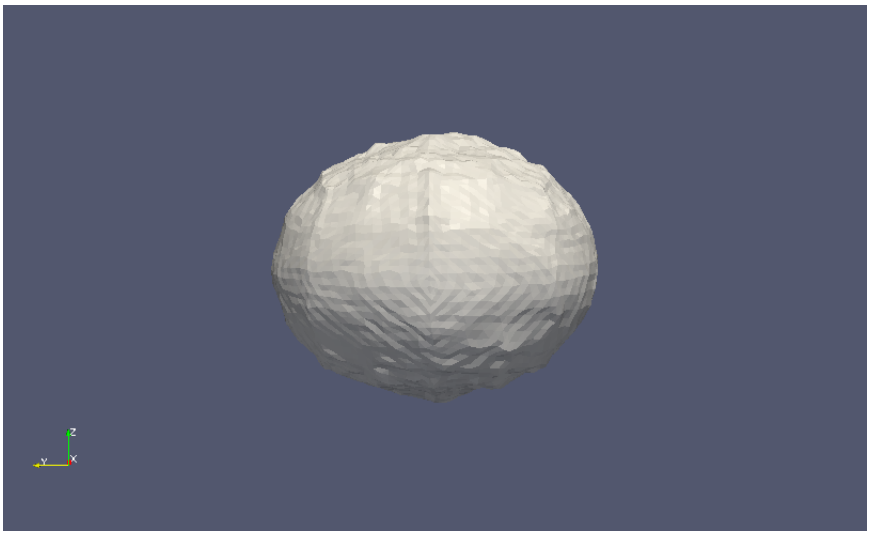
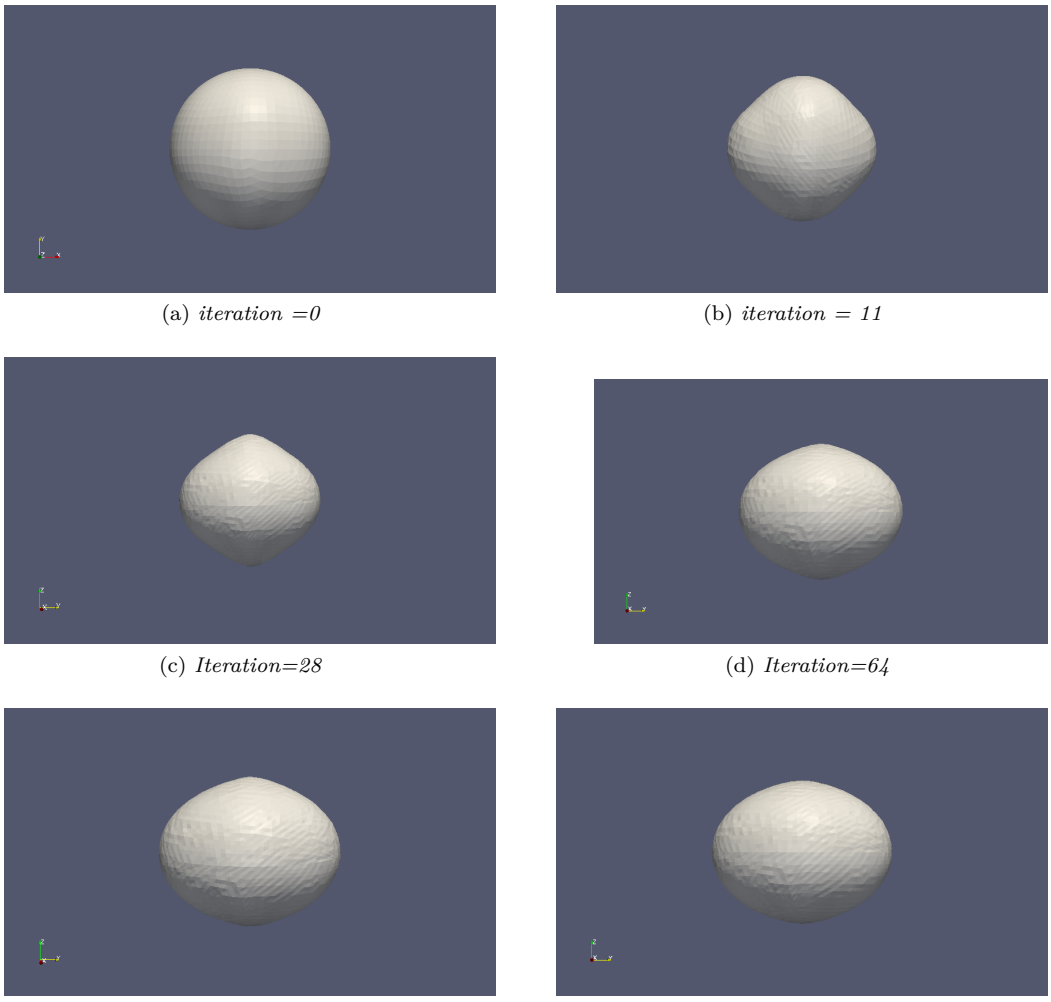


Figure 4.4: *Using local averaging to smooth the mesh with five iterations.*

Figure (4.7) shows a plot of the minimum mesh quality over iteration for a spherical mesh undergoing deformation. Figure (4.7a) is with $p = 0$. As shown the quality decreases as the iteration is progresses. Figure (4.7b) depicts the minimum mesh quality over iteration for the case with extra stiffness attached to cells with low quality. As shown, the quality improves up till iteration $= 10$ then shortly after that, drops a little bit. It stops here because the stopping criterion imposed has been met. From these plots, it is important to find a good balance between choosing a tolerance value for the stopping criterion, number of iterations and mesh quality for the deformation.



(a) *iteration =0*

(b) *iteration = 11*

(c) *Iteration=28*

(d) *Iteration=64*

(e) *Iteration=70*

(f) *Iteration=179*

Figure 4.5: *A sphere undergoing deformations in number of steps shown into an ellipsoid. The p value has been set to zero and* $\sigma = 0.5$

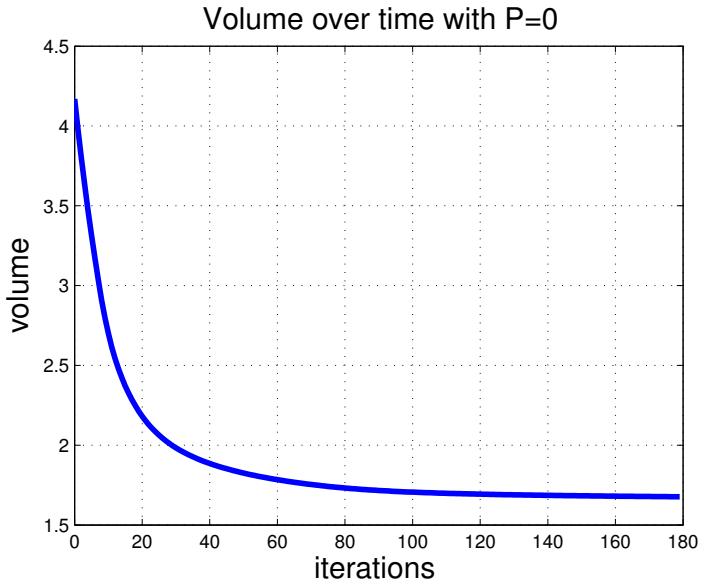
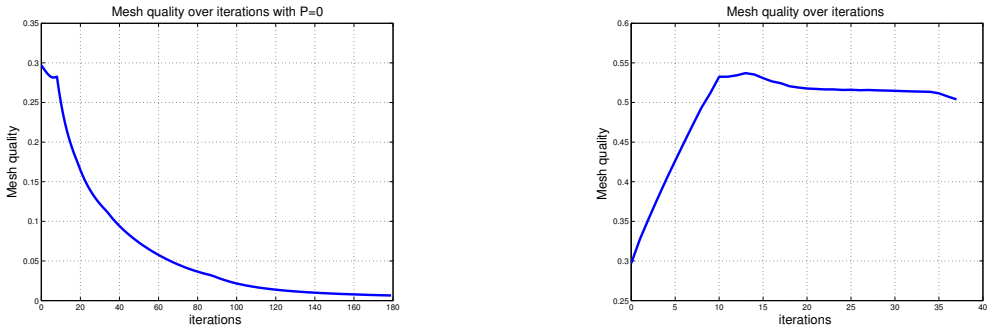


Figure 4.6: *A plot showing how the size of the volume of the sphere changes in each iteration. This figure represents the volume in figure 4.5 with a pseudo time step of* 0.0001 *with* $p = 0$*. The target volume is 1.41.*



(a) *Mesh quality over iteration with* $p = 0$

(b) *Mesh quality with* $p = 2$.

Figure 4.7: *This figures shows how the quality of the mesh with vertices=27593 varies with number of iterations with* $\bar{\sigma} = 0.5$

Chapter 5

Application

In this chapter, the application of the method developed in the previous chapter to the Philips data set is shown. To begin the illustration, the Philips data set is described and how to extract the relevant part of the data is also outlined. The relevant part in this context is the inner wall of the left ventricle (LV) which will later be used with the heart solver code mentioned earlier. The solver works with volume meshes of the left ventricle and the data set consists of surface meshes which are not embedded, therefore a conversion from surface mesh to volume mesh has to be made. Having the extracted mesh ready, the gradient vector flow computation is then carried out on the mesh which will then be used in deforming a generic model. The pipeline for the procedure is shown in Figure (5.1).

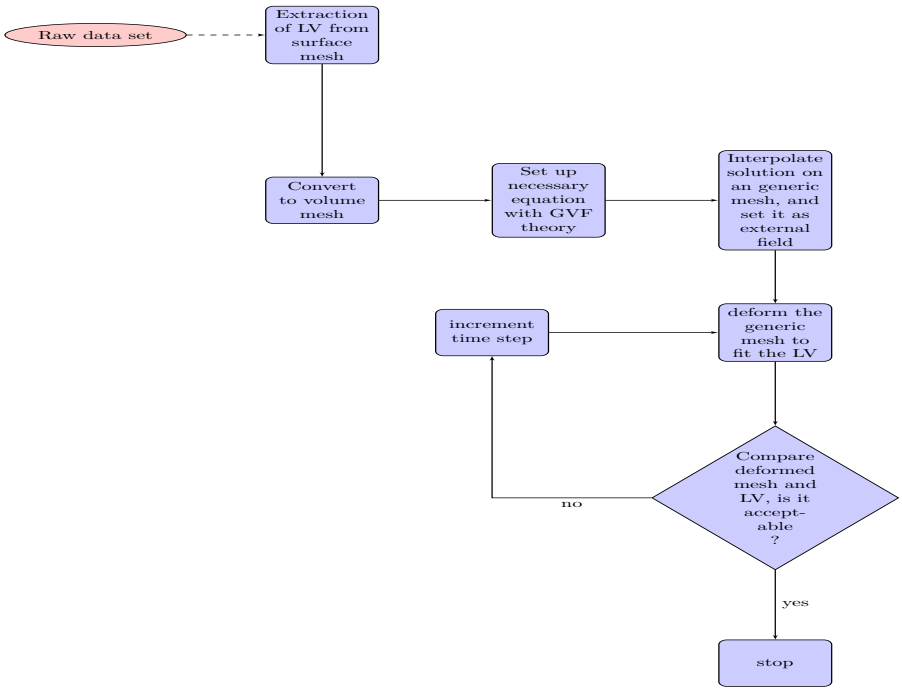
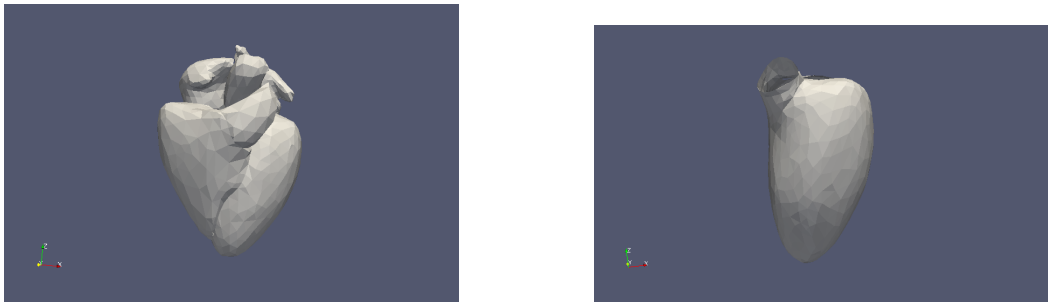


Figure 5.1: *Flow diagram showing the procedure of moving from the raw data to generating a patient specific model*



(a) *The whole heart as a surface mesh*

(b) *Extracted left ventricle*

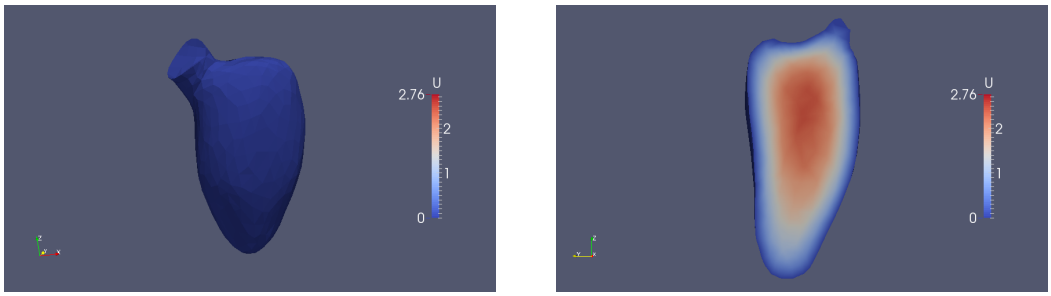
Figure 5.2: The Philips heart model

5.1 Automated patient specific data extraction

The data set for this project is provided by Philips. It consists of time ($t = 0 : 1 : 33$) snapshots of the whole heart in the form of a surface meshes in object file format (**.off**). For more details about object file format see [2]. The first time snapshot is considered for the application of the developed method of the previous chapter. For the remaining time snapshots, the same method, as applied to the first time snapshot holds. The **.off** file is read and the **MeshEditor** is used in building the surface mesh. The result from this building is depicted in Figure (5.2a). For the purpose of this project the left ventricle, as shown in Figure(5.2b) is extracted by means of the following Algorithm 3.

Algorithm 3 Extraction of left ventricle from the whole heart

```
1: Declare maps: new_verts, new_cells, old_cells
2: Load the mesh
3: int local_cell = 0;
4: for each cell in the mesh do
5:    if cell index is in between 619 and 1717 then
6:       int j =0
7:       for each vertex in a cell do
8:          set global_id ← vertex.index()
9:          if global_id is not in verts array then
10:            verts[global_id] = local_id
11:            insert in new_verts map local_id and points of the vertex
12:            increment local_id
13:         end if
14:         old_cells[cell →index()][j] = global_id
15:         new_cells[local_cell][j] = verts[global_id]
16:         increment j
17:      end for
18:      increment local_cell
19:   end if
20: end for
21: Using mesh editor to build extracted mesh
22: set num_verts as the size of new_verts map
23: set size_of_cells as the size of new_cells map
24: MeshEditor editor
25: Declare a new mesh, mesh1
26: editor.open(mesh1, CellType::triangle, 2, 3)
27: editor.initVertices(num_verts)
28: editor.initCells(size_of_cells)
29: for k= 0 : num_verts-1 do
30:    set Point p ← new_verts[k]
31:     editor.addVertex(k, p[0], p[1], p[2])
32: end for
33: Get cell connectivities
34: for k= 0: num_verts-1 do
35:    set point p ← new_cells[k]
36:    editor.addCell(k, p[0], p[1], p[2]);
37: end for
38: editor.close()
39: Write new mesh to a file
40: File f(leftventricle.pvd )
41: f<< mesh1;
```



(a) *Solution of the Eikonal equation on the whole of the left ventricle.*

(b) *Cross section of the left ventricle with the computed solution of the Eikonal equation.*

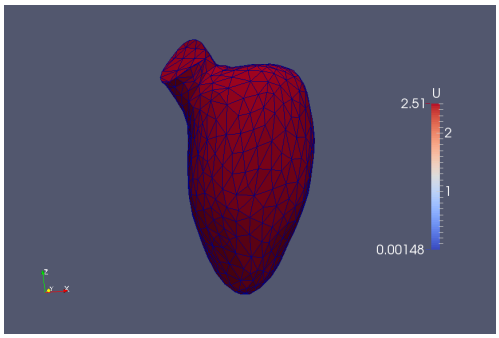
Figure 5.3: *Numerical solution of the Eikonal equation on the left ventricle.*

5.2 Mesh generation

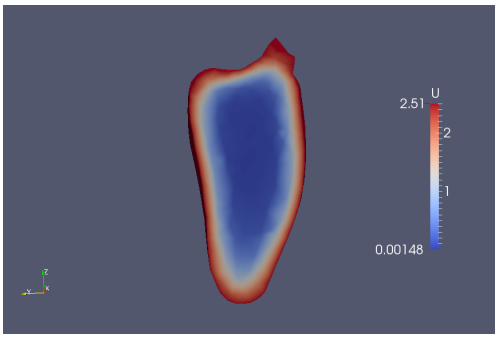
The challenge in this project is owing to the fact that there are two dissimilar meshes involved. A surface mesh from the available data and a *a priori* model in the form of a volume mesh. The task is to generate the external field from the surface mesh and use it to deform the model. A direct computation of the GVF field on the surface mesh can not be used in deforming the model since the two meshes are not compatible. An approach of connecting these meshes has to be developed. As a first test of the method developed in Chapter 2, a conversion of the extracted inner wall of the left ventricle to a volume mesh is done by using ANSA software [3]. The GVF is computed on the generated volume mesh which will be used to deform the model.

5.3 Gradient vector flow computation

To begin the GVF computation, the Eikonal equation is first solved on the left ventricle. The result is shown in Figure (5.3)



(a) *Edge map function computed on the left ventricle.*



(b) *Cross section of the left ventricle showing the edge map function computed on the left ventricle.*

Figure 5.4: *Edge map function computed on the left ventricle.*

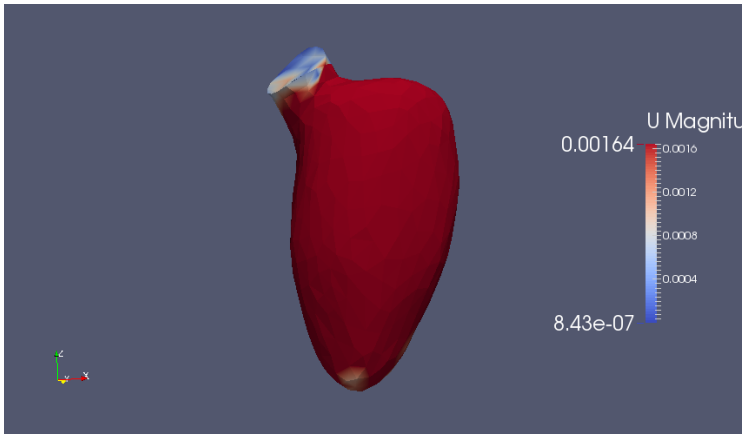


Figure 5.5: *Gradient vector flow computed on the left ventricle.*

From Figure (5.3b), it is observed that the solution of the Eikonal equation, which says that the distance should be zero at the boundary and has a maximum value at the farthest point from the boundary.
The next step is to calculate the edge map function using the solution of the Eikonal equation. Recall that the edge map function is given by

$$f(x,y,z) = \frac{1}{\bar{\sigma}\sqrt{2\pi}} \exp\left(\frac{-g(x,y,z)^2}{2\bar{\sigma}^2}\right)$$

where $g(x,y,z)$ is the solution of the Eikonal equation, with $\bar{\sigma} = 0.5$. The results of this computation is shown in Figure (5.4). From this figure, the edge map has maximum value on and near the boundary which is a desired property, as already mentioned in Chapter 2. Computing the GVF on the heart yields the result shown in Figure (5.5).
Now at this point the generic model, in this case a sphere is to be deformed. The solution obtained from solving the GVF equation on the left ventricle has to be projected on the spherical mesh since the information about the shape of the ventricle has to be mapped on to the spherical mesh. Figure (5.6) shows the cross section of the sphere with the projected solution. The regions in the sphere with non zero values are then set as the target geometry

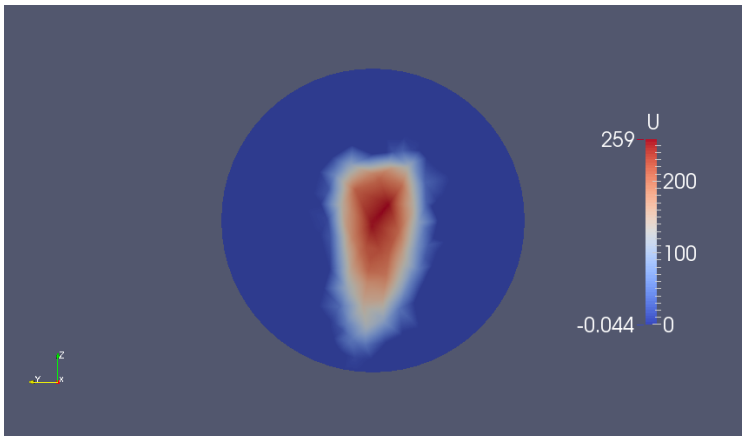
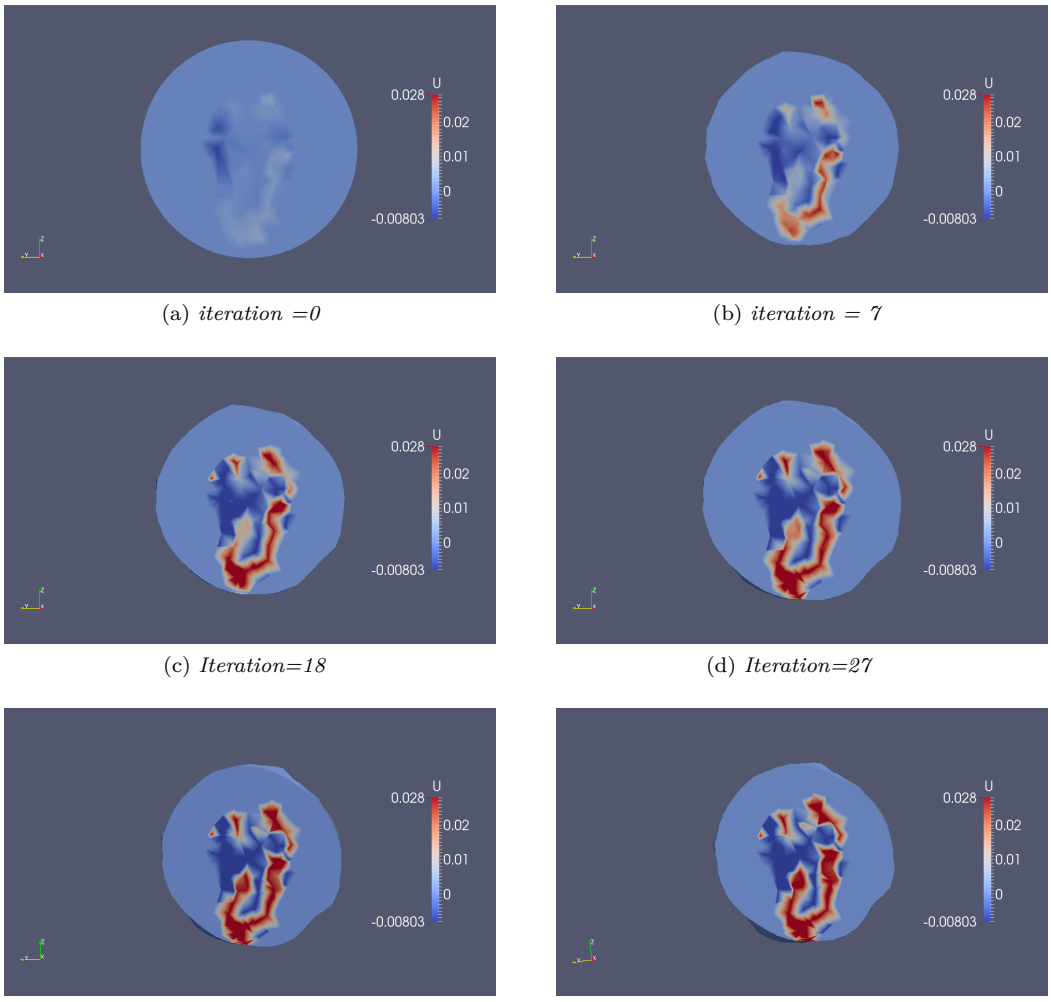


Figure 5.6: *Interpolated solution on the spherical mesh*

for the sphere to be deformed into. In Figure (5.7), it shows some deformation of the sphere to the target. The deformation is not enough. This is due to the fact that the projected force exist only within certain regions (left ventricles) resulting only in pulls from within the region and no push externally. To solve this problem, the vertices of the surface mesh should be projected (mapped) onto the sphere and the Eikonal equation will then be computed for the whole domain. This ensures that the external force is in the whole domain. By doing this, the projection of the GVF solution computed on the extracted left ventricle from the data set is avoided and converting the extracted inner wall of the left ventricle to a volume mesh is also avoided.



(a) *iteration =0*

(b) *iteration = 7*

(c) *Iteration=18*

(d) *Iteration=27*

(e) *Iteration=40*

(f) *Iteration=50*

Figure 5.7: *A sphere undergoing deformations in number of steps shown into the left ventricle. The deformation is not enough, which can be seen in the figure.*

Chapter 6

Conclusion

In this project, algorithms and software tools have been developed which are shown in Figure (5.1), as a pipeline for automating the transformation of patient specific model of the human heart from Philips data set to *a priori* model that mimics the left ventricle. In the process of establishing this pipeline, the following was studied;

The gradient vector flow (GVF) was used as a foundation for computing the external force field used in the deformation of the generic model. The GVF was formulated in terms of the edge map function.

Another aspect that was studied, was the ElasticSmoother in Unicorn. The ElasticSmoother was modified to take into account the external force field used in the deformation. Two implementations were studied. That is, the case when the extra stiffness is assigned to cells of poor quality and the case when there is uniform stiffness. The three dimensional case for uniform stiffness produced expected results as seen in [24]. In [24], they used a sphere and transformed it to an ellipsoid. The problem that arose with the ElasticSmoother for the case with assigning extra stiffness , was the roughness for three dimensional cases , and further investigations will be carried out.
When the developed methods were applied to the Philips data set, the deformation was not enough due to the projected force being limited to certain parts of the domain. To overcome this problem, project the vertices of the surface mesh on to the volume mesh and compute the Eikonal equation. By doing this, the external force is available on the whole domain.
The method developed in this project for generating a mesh can be improved upon to include learning algorithms such as Bayesian learning algorithm, whereby for an unseen data set it can adjust itself and produced the required mesh [19]. This is an advantage for having a fully automated routine. On the other hand, there are three partial differential equations that require solutions to them, making this method computationally intensive.

Bibliography

[1] http://www.paraview.org/.

[2] http://www.cs.princeton.edu/courses/archive/spr09/cos426/assn2/formats.html.

[3] http://www.beta-cae.gr/ansa.htm.

[4] Chandrajit Bajaj, Samrat Goswami, Zeyun Yu, and Yongjie Zhang. Patient specific heart models from high resolution ct. *International Journal for numerical methods in Biomedical engineering*, 29:850–869, 2013.

[5] Niyazi Cem Degirmenci, Johan Jansson, and Johan Hoffman. Fluid-structure interaction model of vocal folds. *EUNISON*.

[6] K Eriksson, D Estep, P Hanbo, and C Johnson. *Computational Differential Equations*. Cambridge University Press, UK, 1996.

[7] Rafael C. Gonzalez and Richard E. Woods. *Digital image processing*. Prentice Hall.

[8] J. Hoffman and *et al*. Unicorn: Parallel adaptive finite element simulation of turbulent flow and fluid-structure interactio for deforming domains and complex geometry. *Computers and fluids*, 80:310–319, 2013.

[9] Johan Hoffman, Johan Jansson, Rodrigo Vilela de Abreu, Niyazi Cem Degirmenci, Niclas Jansson, Kaspar Müller, Murtazo Nazarov, and Jeannette Hiromi Spühler. Unicorn: Parallel adaptive finite element simulation of turbulent flow and fluid-structure interaction for deforming domains and complex geometry. *Computers & Fluids*, 2012.

[10] Johan Hoffman, Johan Jansson, and Michael Stockli. Unified continuum modeling of fluid-structure interaction. *Mathematical Models and Methods in Applied Sciences*, 21(3):491–513, 2011.

[11] Johan Hoffman and Claes Johnson. *Computational Turbulent Incompressible Flow*. Springer, Germany, 2007.

[12] J Jansson and J Hoffman. Simulation-Visualization-Interaction. `http://www.kth.se/en/csc/forskning/forskningsnyheter/simulation-visualization-interaction-1.396026`, 2013. [Online].

[13] Niclas Jansson. *High Performance Adaptive Finite Element Methods: With Applications in Aerodynamics*. PhD thesis, KTH Royal Institute of Technology, 2013.

[14] S.S Khalafvand, E.Y.N Ng, and L. Zhong. CFD simulation of flow through the heart: a perspective review. *Computer Methods in Biomechanics and Biomedical Engineering*, 14(1):113–132, Febuary 2011.

[15] Mats G Larson and Fredrik Bengzon. *The Finite element method: Theory, Implementation and Applications*, volume 10. Springer-Verlag Berlin Heidelberg, 2013.

[16] A. Logg and G. N. Wells. Dolfin: Automated finite element computing. *ACM Trans. Math. Softw.*, pages 1–28, 2010.

[17] Anders Logg, Kent-Andre Mardal, and Garth N. Wells. *Automated Solution of Differential Equations by the Finite Element Method*, volume 84. Springer-Verlag, 2011.

[18] Q Long, R Merrifield, X Y Xu, P Kilner, D N Firmin, and G-Z Yang. Subject-specific computational simulation of left ventricle flow based on magnetic resonance imaging. *Journal of Engineering in Medicine*, 222:475–485, 2008.

[19] Stephen Marsland. *Machine Learning: An Algorithmic Perspective*. CRC, New York, 2009.

[20] Maxwell Lewis Neal and Roy Kerckhoffs. Current progress in patient specific modeling. *Briefings IN Bioinformatics*, II(1):111–126, 2009.

[21] J.T Ottesen, M.S Olufsen, and J.K Larsen. *Applied Mathematical Models in Human Physiology*. Roskilde University Denmark, 2003.

[22] Markus Persson, Jan Erik Solem, Karin Markenroth, Jonas Svensson, and Anders Heyden. *Phase Contrast MRI Segmentation Using Velocity and Intensity*, volume 3459. Springer Berlin Heidelberg, 2008.

[23] Q. C. Pham, F. Vincent, P. Clarysse, P. Croisille, and I. E Magnin. A fem based deformable model for the 3d segmentation and tracking of the heart in cardiac mri. pages 250–254.

[24] Youssef Rouchdy, Jerome Pousin, Joel Schaerer, and Patrick Clarysse. A nonlinear elastic deformable template for soft structure segmentation. application to the heart segmentation in mri. *Inverse problems*, 3:1–23, 2007.

[25] Shankar P. Sastry, Jibum Kim, Suzanne M. Shontz, Brent A.Craven, Frank C. Lynch, Keefe B. Manning, and Than Panitanarak. Patient-specific model generation and simulation for pre-operative surgical guidance for pulmonary embolism treatment. *Lecture Notes in Computational Vision and Biomechanics*, 3:223–249, 2013.

[26] Jung Hee Seo, Vijay Vedula, Theodore Abraham, and Rajat Mittal. Multiphysics computational models for cardiac flow and virtual cardiography. *International Journal for numerical methods in Biomedical engineering*, 29:850–869, 2013.

[27] David A Steinman. Image-based computational fluid dynamics modeling in realistic arterial geometries. *Annals of Biomedical Engineering*, 30:483–497, 2002.

[28] Boyang Su, Liang Zhong, Xi-Kun Wang, Jun-Mei Zhang, Ru San Tan, John Carson Allen, Soon Keat Tan, Sangho Kim, and Hwa Liang Leo. Numerical simulation of patient specific left ventricle model with both mitral and aortic valves by fsi approach. *Computer methods and Programs in Biomedicine, Elsevier*, 113(2):474–482, 2013.

[29] Clarence Wilbur Taber. *Taber's Cyclopedic Medical Dictionary*. F a Davis Company, 2009.

[30] The Frankline Institute. The human heart. `www.fi.edu/learn/heart/index.html`. [Online].

[31] J Weese, J Peters, C Meyer, I Wächter, R Kneser, H Lehmann, O Ecabert, H Barschdorf, R Hanna, F M Weber, O Dössel, and C Lorenz. *The Generation of Patient Specific Heart Models for Diagnosis and Interventions*, volume 113. Springer-Verlag.

[32] Chenyang Xu and Jerry L Prince. Snakes, shapes and gradient vector flow. *IEEE Transactions on Image Processing*, 7(3):359–369, 1998.

TRITA-MAT-E 2014:67
ISRN-KTH/MAT/E—14/67-SE

www.kth.se