Mathematical and numerical methods for physical regulations of medical images

Victoria Nymo

Master thesis in Applied and Computational Mathematics

Department of Mathematics
University of Bergen
Norway
June 2020

.

Acknowledgements

First of all I would like to thank my family and friends for all the love and support during this process. I would like to thank my supervisor Erik A. Hanson for all the advice, encouragement and endless patience. Also, I want to thank Jhabriel Varela for all the help and to Jan M. Nordbotten for helpful feedback.

.

Contents

.

1 Introduction

During the last decade, inverse problems have been growing in interest. This may be because inverse problems can be described in many ways and have many applications areas, for example medical images, seismology, etc. In our daily life we often consider inverse problems, such as if we wonder where a sound or a light comes from. These problems may be difficult to solve in our daily life but they can also challenging from a pure mathematical point of view. One reason that problems like this can be challenging is that there may exist many approximations that fit the data. For inverse problems it is essential to characterise just what solution have been obtained, how good it is in the terms of physical possibility to fit the data, and perhaps how consistent it is with other constrains.

In other studies it is a common practice to not consider the boundary conditions for physical problems. The boundary conditions are often set to zero or a random constant for simplicity. However in this thesis we will investigate the possibilities for finding the boundaries for different problems. We will look at both simple, simulated problems and real cases. The real cases we will investigate are the ability of finding the boundaries of an image series deforming over time. Where the image we will use is an MR image of the abdominal area of a human. The boundary we want to find in this image is the boundary on the top where the force from the lungs from when the person breaths affects the abdominal area. Our main goal in this thesis is to find a simulation of the breathing cycle of a human using MR images over the abdominal area in a time interval. From Figure 1 we see one of the MR image from the sequence we will examine, where the force from the lungs is acting on the top of the image.

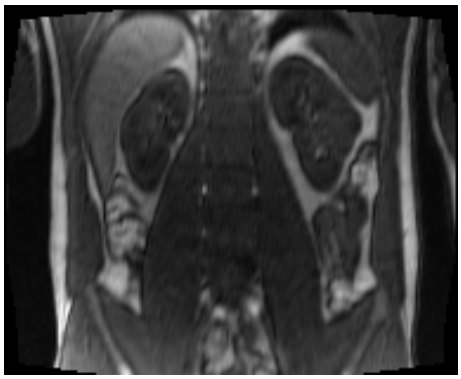


Figure 1: MR image of the abdominal area, [1]

Working towards our main goal, we build up this thesis by simulate some simple forward problems. We use two well-known equations to describe our problem, the heat equation and the Navier-Lame. These equations will be derived in Chapter 1. Starting with the heat equation because this is a smooth

equation and describes physical problems in a simple way. Then advancing to the Navier-Lame equation which is more realistic and describes linear elastic bodies well. Using different numerical methods, as the MPSA, to simulate a solution to the forward problem. Once we have the solution, we then examine the inverse problem from observations of the solution from the forward problem. Solving the inverse problem by using the optimization method gradient descent.

Then for the real case where we use a sequence of MR images over a time interval. Here the forward problem is already solved and we will only be working with the inverse problem. We solve the optimization problem using the gradient descent. With the background from what we have done earlier in the thesis, we could expect the framework to find an approximation of the solution also in this problem. To do the simulations we use the MatLab Reservoir Simulation Toolbox (MRST) and the FV-biot, https://github.com/pmgbergen/fvbiot.

The framework that will be used throughout this thesis is shown in the two equations below

$$O(u(x,t)) = f(x,t) \text{ for } x \in \Omega, t \geq 0 \tag{1.1}$$

and

$$B[u(x,t)] = \zeta(x,t) \text{ for } x \in \Gamma, t \geq 0 \tag{1.2}$$

where eq.(1.1) is the differential operator and eq.(1.2) is the boundary condition operator.

The eqs.(1.1)-(1.2) will be the background for specific equations and boundary conditions used later in this thesis. Note that the variable x does not need to be in the domain $\mathbb{R}^1$ but can be specified to be in different domain.

The thesis is structured in the following way:

Chapter 2: This chapter gives an introduction to basic theory that builds up the method used later in this thesis. Here we present properties necessary to describe the model. In addition, we introduce the equations that describes our problem, the parameters and relationship used in our model, on a general level. At the end of this chapter there will be given a summary of important equations.

Chapter 3: In this chapter we present the methods used in the model. We build up our model step by step and advancing after each step. Also here will there be given a summary of important equations from the chapter.

Chapter 4: After introducing the methods that will be used, we look at different experiments. The first experiments in this chapter is to check that the frameworks works and to get a sense of the limitations that comes with the method used. The experiments afterwards builds up to the real case using MR images over a time interval.

Chapter 5: This chapter sums up the results, limitation and conclude our work and give our final thoughts and further work.

.

2 Background theory

In this chapter we go thru some essential theory that will be used later in this thesis. We will look at theory concerning linear elasticity, inverse problem, optimization problems, ect. This chapter will give an basic understanding for the methods we will look at later in the thesis. However we will start by deriving the two equations we will use through this thesis; the heat equation and the Navier-Lame.

2.1 Introduction to the heat equation

To start we consider a simple equation, the heat equation. The first experiments in Chapter 4 will use this equation to describe the problem we will investigate. The heat equation is a good first equation because it is a smooth equation and describes physical problems well. The word heat refers to the action of transferring energy over time and/or space. In this section we derive the heat equation. The theory in this section is mainly from the books *Fourier Series and Numerical Methods for Partial Differential Equations*, [3], and *Partial differential equations*, [12].

To derive the heat equation consider the domain Ω, and let $u(x,t)$ be the temperature, for $x \in \mathbb{R}^n$. Let the heat contained in Ω be denoted as $H(t)$, where $H(t)$ is defined as

$$H(t) = \int_\Omega \rho u - \psi \, dx, \tag{2.1}$$

where ρ is the density of the material and the function $\psi = \psi(x)$ is the heat source (or sink if ψ is negative). Considering the change in heat

$$H_t = \int_\Omega \rho u_t - \psi \, dx, \tag{2.2}$$

We know that the heat moves from hot to cold regions proportionally to the temperature gradient and the heat can not leave the region besides from through the boundary. From this the heat flux across the boundary equals the change of heat energy on Γ

$$H_t = \iint_\Gamma \kappa \nabla u \cdot \mathbf{n} \, d\Gamma. \tag{2.3}$$

where κ is the heat conductivity, ∇u is the gradient of u, represented as the rate of change over space, and $\mathbf{n}$ is the normal vector.

Next we introduce a classical theorem as described in *Partial differential equations*, [12]. The proof can be found in the same book.

Theorem 1. ***Divergence Theorem***
Let Ω be a bounded spatial domain with a piecewise boundary surface Γ. Let $\boldsymbol{n}$ be the unit normal vector on Γ. Let $f(x)$ be any vector on $\bar{\Omega} = \Omega \cup \Gamma$. Then

$$\int_{\Omega} \nabla \cdot f \, d\boldsymbol{x} = \iint_{\Gamma} f \cdot \boldsymbol{n} \, d\Gamma, \tag{2.4}$$

where $\nabla \cdot f$ is the divergence of f and $d\Gamma$ is the element on surface area on Γ.

Consider the eq. (2.3). Then from Theorem 1 we get

$$H_t = \iint_{\Gamma} \kappa \nabla u \cdot \mathbf{n} \, d\Gamma = \int_{\Omega} \nabla \cdot (\kappa \nabla u) \, d\mathbf{x}, \tag{2.5}$$

Now combining the eq. (2.2) and eq. (2.5) we get

$$\int_{\Omega} \rho u_t - \psi \, d\mathbf{x} = \int_{\Omega} \nabla \cdot (\kappa \nabla u) \, d\mathbf{x}, \tag{2.6}$$

Since this equation is valid for any subdomain in Ω, then eq. (2.6) will be valid for any point in Ω such that we can write eq. (2.6) as

$$\rho u_t - \psi = \nabla \cdot (\kappa \nabla u). \tag{2.7}$$

This is equation is called the heat equation. If the variables ρ and κ are constants, the eq. (2.7) can be reduced to

$$\rho u_t = \kappa \Delta u + \psi, \tag{2.8}$$

where $\nabla \cdot (\nabla u) = \Delta u$. The operator Δ is the *Laplace operator*. The eq. (2.8) is the version of the heat equation we will use through this thesis.

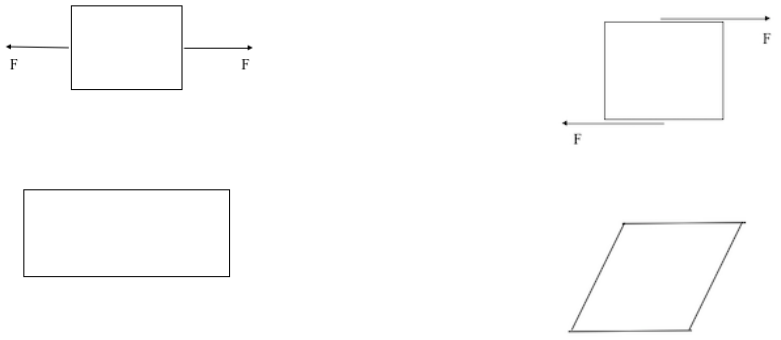
2.2 Introduction to linear elasticity and the Navier-Lame equation

In this section we will focusing on some background theory of linear elasticity as the stress and strain tensors, Hooke's law, Newtons second law of motion, etc.. Finally we derive the Navier-lame equation. For some of the experiments later in this thesis we will use the Navier-Lame equation to describe the problem. This equation gives a more realistic problem and considers important physical parameters.

The first part in this section is inspired from *Continuum Mechanics and thermodynamics* by E. B. Tadmor, R. E. Miller and R. S. Elliott, [13], *Introduction to Solid Mechanics* by J. Lubliner and P. Papadopoulos, [8].

2.2.1 Stress

To begin we describe stress. Consider a solid cube of some material and assume we use a force, F, on both ends of the cube, illustrated in Figure 2a. Then for an elastic body the force may pull the cube such that the cube stretches in the direction of the stretching. Also we could expect that the cube contracts in the direction transverse to the direction of the stretching. Now if we took the same block and the same force (in the same direction) but on the other face of the block, we could expect the block to look more like as in Figure 2b.



(a) (b)

Figure 2: a) uniaxial tension b) shear tension

The stress tensor is the generic representation of the state of stress inside the material. Consider a small cube element with faces in different directions. The stress tensor describes the forces or stresses acting on each of the faces. In a Cartesian coordinate system in $\mathbb{R}^3$ we have faces which have normal directions pointing in the $x-$ $y-$ and $z-$ directions, and on each of the faces we have $x-, y-$ and $z-$ forces. Let all the stresses act on each positive face of the body,

as in Figure (3). The stress tensor is constructed with the normal stress along the diagonal components and the shear stress on the off-diagonal components. The stress are described with two subscript, the first subscript indicates the direction of the stress vector and the second refers to the direction normal to the cross-section.

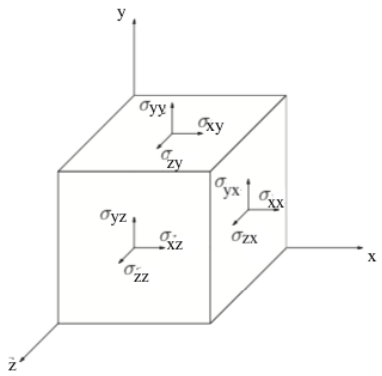


Figure 3: Stress acting on a three-dimensional isotropic volume element. This image is inspired from *Continuum Mechanics and thermodynamics*, [13]

All the components of the stress can be expressed in the stress tensor, using matrix notation

$$\sigma = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix} \tag{2.9}$$

2.2.2 Strain tensor

To construct the strain tensor we must know what deformation is. From Figure 2a and 2b we see two types of deformation. We see deformation as the materials responds on how much it has deformed. Sometimes we apply a fixed deformation and then measure the amount of force needed for that deformation and other times we apply a force, or stress, and we see how much deformations there is in the solid.

To construct the strain tensor consider the deformation, $u(x)$, of a elastic body, where the deformation is defined as the change in the shape or size of the body between two points. Let a and b be two points close to each other in the elastic body. The line from a to b is denoted as $L = b - a$. When the body is deformed the points move relative to each other to two new positions a' and b' s.t. $L' = b' - a'$. The change in the line element is a vector given as the gradient of u

$$\nabla u = u(b) - u(a) = L' - L \tag{2.10}$$

The strain is defined as the change in the position of the two points a and b, when an elastic body has had a type of deformation. The strain tensor then takes the form

$$\epsilon = \frac{1}{2}(\nabla u + (\nabla u)^T) \tag{2.11}$$

2.2.3 Poisson's ratio

As we have seen in Figure 2a when elastic bodies are stretched, the material tends to contract in the direction transverse to the direction of stretching. Or when elastic bodies are compressed, it is common to observe that the material tends to expand in the direction perpendicular to the direction of compression. The strain in the longitudinal direction is known as the longitudinal strain and the strain in the lateral direction is known as lateral strain. This is known as the Poisson effect, and the Poisson's ratio ν is a quantitative measure of this effect, [8]. Poisson's ratio is defined as

$$\nu = -\frac{\epsilon_{lat}}{\epsilon_{long}}. \tag{2.12}$$

When there is no lateral contraction the Poisson's ratio is zero, $\nu = 0$.

2.2.4 Hooke's law

Now we derive Hooke's law. this section is inspiration from the master thesis *Implementation of an MPFA/MPSA-FV Solver for the Unsaturated Flow in Deformable Porous Media* by Jhabriel Varela, [16].

Consider the isotropic volume element in Figure 3. Then strains in eq. (2.13)-(2.15) is assembled by the stress σ_{xx}, σ_{yy} and σ_{zz}

$$\epsilon_{xx} = \frac{\sigma_{xx}}{E}, \quad \epsilon_{yy} = -\nu\frac{\sigma_{yy}}{E}, \quad \epsilon_{zz} = -\nu\frac{\sigma_{zz}}{E}, \tag{2.13}$$

$$\epsilon_{xx} = -\nu\frac{\sigma_{xx}}{E}, \quad \epsilon_{yy} = -\frac{\sigma_{yy}}{E}, \quad \epsilon_{zz} = -\nu\frac{\sigma_{zz}}{E} \tag{2.14}$$

and

$$\epsilon_{xx} = -\nu\frac{\sigma_{xx}}{E}, \quad \epsilon_{yy} = -\nu\frac{\sigma_{yy}}{E}, \quad \epsilon_{zz} = \frac{\sigma_{zz}}{E}. \tag{2.15}$$

for Young's modulus E. Young's modulus defines the relationship between the stress and strain, and measures the stiffness of a solid material.

We can compress the eqs. (2.13)-(2.15) to

$$\begin{aligned}
\epsilon_{xx} &= \frac{\sigma_{xx} - \nu(\sigma_{yy} + \sigma_{zz})}{E}, \\
\epsilon_{yy} &= \frac{\sigma_{yy} - \nu(\sigma_{zz} + \sigma_{xx})}{E}, \\
\epsilon_{zz} &= \frac{\sigma_{zz} - \nu(\sigma_{xx} + \sigma_{yy})}{E}.
\end{aligned} \tag{2.16}$$

The shear strains are give by

$$\begin{aligned}\epsilon_{xy} = \epsilon_{yx} &= \frac{1+\nu}{E}\sigma_{xy},\\ \epsilon_{yz} = \epsilon_{zy} &= \frac{1+\nu}{E}\sigma_{yz},\\ \epsilon_{zx} = \epsilon_{xz} &= \frac{1+\nu}{E}\sigma_{zx}.\end{aligned} \tag{2.17}$$

As long as $\nu \neq -1$ and $\nu \neq 0.5$, we can rewrite eq. (2.16) and eq. (2.17) to a function of strains describing the stress. Introducing the shear modulus S as the ratio of shear stress to the strain

$$\begin{aligned}\sigma_{xx} &= \frac{E}{(1+\nu)(1-2\nu)}((1-\nu)\epsilon_{xx} + \nu\epsilon_{yy} + \nu\epsilon_{zz}),\\ \sigma_{yy} &= \frac{E}{(1+\nu)(1-2\nu)}((1-\nu)\epsilon_{yy} + \nu\epsilon_{xx} + \nu\epsilon_{zz}),\\ \sigma_{zz} &= \frac{E}{(1+\nu)(1-2\nu)}((1-\nu)\epsilon_{zz} + \nu\epsilon_{xx} + \nu\epsilon_{yy}).\end{aligned} \tag{2.18}$$

and

$$\begin{aligned}\sigma_{xy} &= 2S\epsilon_{xy},\\ \sigma_{yz} &= 2S\epsilon_{yz},\\ \sigma_{zx} &= 2S\epsilon_{zx}.\end{aligned} \tag{2.19}$$

We can represent the strains and the stresses as a column-matrix notation, [16], such that

$$\sigma = \begin{Bmatrix}\sigma_{xx}\\ \sigma_{yy}\\ \sigma_{zz}\\ \sigma_{yz}\\ \sigma_{zx}\\ \sigma_{xy}\end{Bmatrix}$$

and

$$\epsilon = \begin{Bmatrix}\epsilon_{xx}\\ \epsilon_{yy}\\ \epsilon_{zz}\\ 2\epsilon_{yz}\\ 2\epsilon_{zx}\\ 2\epsilon_{xy}\end{Bmatrix}$$

Now the relation between the stresses and the strain is given as Hooke's law

$$\sigma = \wp\epsilon \tag{2.20}$$

where $\wp$ is a forth order tensor given as

$$\wp = \frac{E}{1+\nu}\begin{Bmatrix} \frac{1-\nu}{1-2\nu} & \frac{\nu}{1-2\nu} & \frac{\nu}{1-2\nu} & 0 & 0 & 0 \\ \frac{\nu}{1-2\nu} & \frac{1-\nu}{1-2\nu} & \frac{\nu}{1-2\nu} & 0 & 0 & 0 \\ \frac{\nu}{1-2\nu} & \frac{\nu}{1-2\nu} & \frac{1-\nu}{1-2\nu} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \end{Bmatrix}$$

Let $\mu = \frac{E}{2(1+\nu)}$ and $\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}$, where λ and μ are the Lames parameters. Then Hooke's law can be written as

$$\sigma = \lambda\theta I + 2\mu\epsilon \tag{2.21}$$

where θ is the trace of ϵ from eq. (2.11), and I is the unit tensor.

2.2.5 Navier-Lame

The derivation of the Navier-Lame is inspired form the article *On the Propagation of Longitudinal Stress Waves in Solids and Fluids by Unifying the Navier-Lame and Navier-Stokes Equations*, [2].

Before we construct the Navier-Lame equation we must introduce Newtons second law of motion. Consider the cube in Figure (3), with volume V. The stress tensor σ, from eq. (2.9), describes the forces or stresses acting on each of the faces. Let F_V be the total surface force acting on the cube. Then F_V can be described as

$$F_V = \iint_\Gamma \sigma \cdot \mathbf{n} d\Gamma \tag{2.22}$$

By using Theorem 1 Divergence theorem, introduced in section 2.1, the equation can be rewritten as

$$F_V = \int_V \nabla \cdot \sigma dV \tag{2.23}$$

Also the total surface force can be described as

$$F_V = \int_V \xi u_{tt} dV \tag{2.24}$$

where ξ is the mass density of the body, such that Newton equation may be written as

$$\int_V (\nabla \cdot \sigma - \xi u_{tt}) dV = 0 \tag{2.25}$$

Since this equation is valid for any subdomain in V, the eq. (2.25) will be valid for any point in V such that we can write eq. (2.25) as the stress equation

$$\nabla \cdot \sigma - \xi u_{tt} = 0 \tag{2.26}$$

We combine the eq. (2.26) with Hooke's law from eq. (2.21), to obtain

$$\xi u_{tt} = \nabla \cdot (\lambda \theta I + 2\mu\epsilon) \tag{2.27}$$

Recall the properties of trace; $tr(\nabla u I) = tr((\nabla u I)^T$ and $tr(\nabla u) = \nabla \cdot u$. Then from eq.(2.27)

$$\begin{aligned}
\nabla \cdot (\theta I) &= \nabla \cdot (tr(\frac{1}{2}(\nabla u + (\nabla u)^T))I) \\
&= \frac{1}{2}\nabla \cdot (tr(\nabla u) + tr(\nabla u)) \\
&= \nabla(\nabla \cdot u) \\
\nabla \cdot (2\epsilon) &= \nabla \cdot (\nabla u + (\nabla u)^T) \\
&= \Delta u + \nabla(\nabla \cdot (u))
\end{aligned}$$

Rewriting eq. (2.27) we get the Navier-Lame

$$u_{tt} = \frac{\lambda + \mu}{\xi}\nabla(\nabla \cdot u) + \frac{\mu}{\xi}\Delta u \tag{2.28}$$

We will use this eq. (2.28) of Navier-Lame throughout the thesis. Note that this equation is only valid for small deformations. There exist other nonlinear equations which are more general, but in this thesis we consider the Navier-Lame.

2.3 Introduction to inverse problems

The purpose of this section is to introduce certain key ideas on inverse problems. Some areas of application of inverse problems are for example to wonder where a sound come from or to investigate where a light come from. One example that may be similar to what we will do in this thesis is; if a acting force compressing a sponge, then the inverse problem would be to find the force by observing the deformation of the sponge.

This section is based on some theory from *Parameter Estimation and Inverse Problems* by Aster and Thurber, [11].

We let $O(u) = u(x)$ from eq.(1.1), be the model, and $u(x)$ is a function of one variable. Let f be a set of data, then a forward problem would be to find some set of data given a model of the true parameter x_{true}

$$O(u(x_{true})) = f. \tag{2.29}$$

For the inverse problem we want to find x given f_{true}

$$f_{true} = O(u(x)). \tag{2.30}$$

If we consider the eq. (2.30), an important issue is that actual observations always contain some amount of noise, η, such that

$$f = u(x_{true}) + \eta \tag{2.31}$$

The noise may make the problem difficult to solve. If the data contain noise, there may not exist a model such that the the inverse problem can be solved. Tho there are other things that may make inverse problems difficult to solve. For example if there exists a solution, there may exist many other solution that fist the data. in other words, if a true solution exist, there may exist other solutions besides the true solution that fits the data. Another example may be that the solution is unstable. This means that if a solution is unstable a small change in the measurements can lead to a large change in the model. If this happens the problem is called ill-posed.

2.4 Introduction to optimizing problems

In this section we examine how to minimize a real-valued function, called cost function. We use a cost function to compare the given data to the approximated solution. We want to find an approximation such that the cost function is small or close to zero. If the cost function is zero or small we know that the approximated solution is close to the solution. This can be described as an optimization problem. There are multiple ways to optimize a function. In this thesis we will use traditional optimization approaches, meaning gradient based method. In some of the experiments in Chapter 4, the data we want to compare are vectors, and to compress the data we can use norms. This means that the cost function can be described as a norm function. In this chapter we will therefore introduce some norm theory and give an introduction to minimizing problems.

The theory in this section is inspired from the books *An introduction to optimization* by E.K.P. Chong, [4] and *Numerical linear algebra* by Trefethen and Bau, [15]. .

The cost functions in this thesis will be a norm function. Norms are often used to compress information about a data set to a single number. For instance we can use norm to compare results, determine the magnitude of a data point in multiple dimensions or computing the error of a predictive model. A norm is a function $||\cdot|| : \mathbb{C}^m \to \mathbb{R}$, and must fulfil the following conditions, for all vectors x and y and for all scalars $\Upsilon \in \mathbb{C}$;

$$\begin{aligned} &1.\ ||x|| \geq 0, \text{ and } ||x|| = 0 \text{ only if } x = 0, \\ &2.\ ||x + y|| \leq ||x|| + ||y||, \\ &3.\ ||\Upsilon x|| = |\Upsilon|\ ||x||. \end{aligned} \tag{2.32}$$

These conditions are collected from *Numerical linear algebra*, [15], where the conditions require that for a nonzero vector the norm is positive, the norm of a vector sum is always equal or less to the sum of the norms of the parts, and that finding the norm of a scaling vector is the same as scales the norm of the vector by the same amount.

The 2-norm is defined below

$$||x||_2 = (\sum_{i=1}^{m} |x_i|^2)^{1/2} = \sqrt{x\ x} \tag{2.33}$$

This is the norm we will use for the cost function later in this thesis.

Now we study the optimization problem. Consider a cost function $c(p)$, then the minimization problem would be given as

$$\min_p c(p) \tag{2.34}$$

For the optimization problem from eq. (2.34) the goal is to find the value of p that gives the smallest value of the cost function c. If $p*$ is a solution to this

problem, $p*$ is called minimizer of c over Ω. This minimizer may not be unique and there could exist multiple values of p that minimize the cost function.

Examine the difference between a global minimizer and a local minimiser. Given a cost function $c(p)$ where $c : \mathbb{R}^n \to \mathbb{R}$ and the vector $p \in \mathbb{R}^n$ of independent variables on a domain Ω. If $p* \in \Omega$ is a global minimizer of c over Ω, then $c(p*)$ has the smallest value than for any other $p \in \Omega$. If $p*$ is a local minimizer of c, then $c(p*)$ has the smallest value for any neighbour point of $p*$,[4].

A optimization problem is only solved when a global minimizer is found. Although global minimizers are difficult to find, we therefore often have to be satisfied with finding local minimizers. Maximize a cost function can be done in the same way as before, this is because maximizing c is equivalent to minimising $-c$.

2.5 Introduction to numerical methods

To given an introduction to the numerical method in Chapter 3 we introduce in this section, how the methods are build and demonstrate a simpler method. For some problems it may be difficult to find a solution in analytic form, and a simpler way may be to elaborate a numerical algorithm. Numerical technique finds only approximated solutions but they are much easier to execute than some of the techniques for finding the exact solutions. Numerical method is therefore a popular method for finding a solution to differential equations. Finite difference method (FDM) and Finite volume method (FVM) are a numerical techniques for solving differential equations. In this section we will use some simple methods to demonstrate the the basic idea of a general approach to construction numerical methods. Later we will specify some numerical methods that will be used for the experiments. This theory is motivated from [7], and *Numerical solution of ordinary differential equations* by , [5],

The idea of FDM is that every derivative in the differential equation, is replace with an approximation of algebraic equations. Then solving the algebraic equations gives an approximation of the solution of the differential equation.

Introducing a simple numerical method may help to illustrate the ideas in a numerical solution of a differential equation. Consider a function of one variable $u(x)$. Form eq. (1.1) let $O(u) = u_x - u$ and $f = 0$ such that

$$u_x = u \text{ for } 0 \leq x \tag{2.35}$$

The definition of the derivative is given as

$$u_x \approx \frac{u(x+v) - u(x)}{v} \tag{2.36}$$

for $v \to 0$. We can use the definition to find a numerical method approximating the derivative in eq.(2.35). In eq.(2.36) we see that the derivative is approximated by taking a small step in the positive direction of x, this can be described as

$$\frac{U_{i+1} - U_i}{v} \approx u_x + \mathcal{O}(v) \tag{2.37}$$

where U_i is an approximation of u at $x_i = iv$.

The two sources of error in FDM are the loss of precision, and truncation error. In eq.(2.37), the local truncation error is proportional to the step sizes v.

We can use the eq. (2.37) such that the eq. (2.35) can be expressed

$$\frac{U_{i+1} - U_i}{v} = U_i \tag{2.38}$$

this can be rewritten to

$$U_{i+1} = (v+1)U_i \tag{2.39}$$

If we know that $u(0) = a$ for $a \in \mathbb{R}$ then $U_0 = a$. We can then use U_0 to find the next value U_1 by using the eq. (2.39). Note that the numerical method gives a

rule for computing $U_1, U_2, ..., U_N$, one after the other, using the known values from the step before. Because of this the method is called an explicit method. To compute the following step in succession is common for many numerical methods.

For a differential equation with derivative in time we can do the same. From eq. (1.1), let $O(u) = u_t - \iota u$ and $f = 0$ such that

$$u_t = u, \ a \leq t \leq b. \tag{2.40}$$

The numerical method is given on the form

$$\frac{1}{d}(U^{k+1} - U^k) = U^k \tag{2.41}$$

where d is the step size in time. This is a forward difference in time and is called Eulers method. We can rewrite this equation such that

$$U^{k+1} = (1 + d)U^k \tag{2.42}$$

Now for a method that solves for both time and space. Here we introduce a centered difference. From eq.(1.1) we let $O(u) = u_t - u_x$ and $f = 0$ Using a forward difference in time and space, the discretization can look like

$$\frac{U_i^{k+1} - U_i^k}{d} = \frac{1}{v}(U_{i+1}^k - U_i^k) \tag{2.43}$$

where d is the step size in time and v is the step size in space.

We will use this approach later in the thesis to find an numerical method for the heat equation.

2.6 Introduction to derivative operators

In this section will examine the derivatives and finding the derivatives using finite differential methods (FDM). Finding the derivatives might be difficult to do directly for some problems. However by using FDM it might be easier to find an approximation of the derivatives than finding it directly. The theory in this section is motivated from *Finite difference methods for Ordinary and Partial Differential Equations* by R. J. LeVeque, [7] and *Analysis in Vector Spaces* by , [9]

First we consider a smooth function of one variable, $u(x)$, and the derivative of u is a bounded well-defined function

Finding the first derivative of the function u by using the definition of derivative

$$u_x = \frac{u(x+v) - u(x)}{v} \tag{2.44}$$

as $v \to 0$.

If we wanted to find the derivative by using FDM, we would let $u_x = D_+u(x)$ where D_+ is an operator such that

$$D_+u(x) = \frac{u(x+v) - u(x)}{v} \tag{2.45}$$

for some small value v. The operator is given as

$$D_+ = \frac{1}{v}\begin{bmatrix} -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -1 & 1 \\ 0 & \cdots & 0 & 0 & -1 \end{bmatrix} \tag{2.46}$$

Because of the operator is evaluating u at one side of x, eq.(2.45) is called a one-sided approximation of u_x.

The eq. (2.45) give a first order accurate approximation to u_x, and indicates that the size of the error is roughly proportional to v itself, [7].

Now for the second order derivatives. An approximations for the second derivative, u_{xx}, can be obtained by applying first order differences to derive approximations of the second order derivatives. Just as the second derivative is the derivative of u_x, we can find an operator constructed by the operator we found earlier. Consider the operator form eq. (2.46), then the second order derivative operator will be defined as

$$D_{+2} = (D_+)^T D_+ \tag{2.47}$$

Now consider a function $u(x, y)$ in a two dimensional space. In this case the function u is a matrix. Reforming the matrix to a vector, such that the columns will be lined under each other.

To illustrate, consider the example-matrix

$$Z = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

The matrix Z will take the form

$$Z = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ \vdots \\ 9 \end{bmatrix}.$$

The discretization of the derivative of the function u in x-direction may look like this

$$u_x = \frac{1}{v}(u(x + v, y) - u(x, y) \tag{2.48}$$

and the discretization in y-direction can be defined like this

$$u_y = \frac{1}{w}(u(x, y + w) - u(x, y). \tag{2.49}$$

If we consider the matrix Z, we want to find the operator in the x-direction and since Z is now a vector as stated before, the operator will then be given as

$$D_x = \frac{1}{v} \begin{bmatrix} -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

and the operator in the y-direction

$$D_y = \frac{1}{w} \begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Defining the operator for the first derivative of u in both x and y-direction as

$$D_1 = \begin{bmatrix} D_x \\ D_y \end{bmatrix}, \tag{2.50}$$

As in eq. (2.47) we construct the second derivative operator by using the first derivative operator, such that

$$D_2 = D_1^T D_1 \tag{2.51}$$

We will use this operator to find the second derivatives in section 2.1.

2.7 Overview

Here is an overview of some of the important equations form this chapter.

Heat equation

$$\rho u_t = \kappa \Delta u + \psi, \qquad \text{(2.8 revisited)}$$

Navier-Lame

$$u_{tt} = \frac{\lambda + \mu}{\xi} \nabla(\nabla \cdot u) + \frac{\mu}{\xi} \Delta u \qquad \text{(2.28 revisited)}$$

Minimization problem

$$\min_p c(p) \qquad \text{(2.34 revisited)}$$

Second derivative operator

$$D_2 = D_1^T D_1 \qquad \text{(2.51 revisited)}$$

.

3 Method

In this chapter we will investigate the methods we will use later in this thesis. With the background of theory from the chapter before the up-building of the method and the application are expected to be easily understood. This chapter contains some numerical methods and an optimization method.

3.1 Numerical method for heat equation

As stated before for the experiments in Chapter 4 we will use two framework, one for the forward problem and one for the inverse problem. For the forward problem we will solve the heat equation problem with a numerical method. In this section the numerical method for the heat equation will be discussed. This section is based on the book *Finite difference methods for Ordinary and Partial Differential Equations* by R. J. LeVeque, [7].

We start with defining $O(u)$ and f from eq. (1.1) as $O(u) = \rho u_t - \kappa \Delta u$ and $f = \psi$ such that we get the heat equation,

$$\rho u_t = \kappa \Delta u + \psi \tag{2.8 revisited}$$

for $u = u(x,t)$ and $\psi = \psi(x,t)$ where $x \in \mathbb{R}^n$. Let $\rho = 1$, for simplicity. Now from eq. (1.2) we let the boundary be specified at each end (Dirichlet boundary conditions)

$$B[u(x,t)] = \begin{cases} u(a,t) = \iota(a,t) \\ u(b,t) = \beta(b,t) \end{cases} \tag{3.1}$$

where $a, b \in \mathbb{R}^n$. We will look at the steady state where $u(x,t)$, $\psi(x,t)$, $\iota(t)$ and $\beta(t)$ are all time independent, and the solution is unchanged at later times, where $u_t = 0$. Then the eq. (2.8)

$$\Delta u(x) = g(x) \text{ for } x \in \mathbb{R}^n \tag{3.2}$$

and

$$\begin{aligned} u(a) &= \iota \\ u(b) &= \beta, \end{aligned} \tag{3.3}$$

where we introduce $g(x) = -\psi(x)/\kappa$ to avoid minus signs below.

As a first example of a numerical method we look at the one dimensional case. Let $n = 1$, and let $a = 0$ and $b = 1$ such that $0 < x < 1$. This is a simple problem and can be solved explicitly by integrating $g(x)$ twice and finding the two constants of integration so that the boundary conditions are satisfied. By solving this problem using finite differential methods (FDM) illustrates some of the most essential qualities of FDM.

Consider an approximation U_i of the solution $u(x_i)$, where U_i is a grid function consisting of values $U_0, U_1, ..., U_m, U_{m+1}$. In this case $x_i = iv$ where

$v = 1/(m+1)$ is the mesh width. The first value of the grid function will be equal to the boundary value at the left side, $U_0 = \iota$ and the last value will be equal to the boundary value at the right side, $U_{m+1} = \beta$. Now there are m unknown values that we need to compute to find an approximation of the solution, namely $U_1, ..., U_m$. A discretization of the first derivative of u can be written as

$$u_x = \frac{1}{v}(U_{i+1} - U_i). \tag{3.4}$$

We see that this is the same as in eq. (2.45) from section 2.6, and we can rewrite eq.(3.4) as

$$u_x = D_1 U \tag{3.5}$$

Now for the second derivative u_{xx}, we can use the operator D_1 from eq. (2.46) to find the second derivative operator D_2, as in eq. (2.47) such that

$$u_{xx} = D_2 U \tag{3.6}$$

where

$$D_2 U = \frac{1}{v^2}(U_{i-1} - 2U_i + U_{i+1}). \tag{3.7}$$

If we replace Δu in eq. (3.2) with the centered difference approximation from eq. 3.7, we obtain

$$\frac{1}{v^2}(U_{i-1} - 2U_i + U_{i+1}) = g(x_i) \text{ for } i = 1, 2, ..., m. \tag{3.8}$$

Be noted that the first equation, when $i = 1$ involves the value $U_0 = \upsilon$ and the last equation, $i = m$, involves the value $U_m = \beta$. We have a linear system of m equations for m unknowns. This can be expressed on the form

$$D_2 U = G \tag{3.9}$$

where U is the vector of unknowns, $U = [U_1, U_2, ..., U_m]^T$ and

$$G = \begin{bmatrix} g(x_1) - \iota/v^2 \\ g(x_2) \\ g(x_3) \\ \vdots \\ g(x_{m-1}) \\ g(x_m) - \beta/v^2 \end{bmatrix}.$$

Now consider an example of eq. (3.2) in a two dimensional space, where $n = 2$, with Dirichlet boundary conditions for $B[u(x, y)]$ from eq.(1.2).

$$\begin{aligned} &\Delta u(x, y) = g(x, y) \text{ for } (x, y) \text{ in } \Omega \\ &u = 0 \text{ on } \Gamma \end{aligned} \tag{3.10}$$

where Ω is the domain and Γ is the boundary of Ω.

Consider a grid function consisting of values $U_{i,j}$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$, where U is our approximation to the solution $u(x_i, y_j)$. Here $x_i = iv$ and $v = 1/(m+1)$, $y_j = jw$ and $w = 1/(n+1)$ the mesh width in x and y-direction. The discretization of the derivative of the function u in x-direction may look like this

$$u_x = \frac{1}{v}(U_{i+1,j} - U_{i,j}) \tag{3.11}$$

and the discretization in y-direction can be defined like this

$$u_y = \frac{1}{w}(U_{i,j+1} - U_{i,j}). \tag{3.12}$$

let $v = w$. From this we get the operators D_1 from eq. (2.50) and D_2 eq. (2.51). We can replace Δu in eq. (3.10) with $D_2 U$ s.t. we obtain

$$D_2 U = G \tag{3.9 revisited}$$

We now consider the whole heat equation for a two dimensional space

$$\rho u_t = \kappa \Delta u + \psi. \tag{2.8 revisited}$$

Let $\rho = 1$ and $\kappa = 1$ as before, and let $\psi = 0$.

From earlier we already know that $\Delta u = D_2 U$, so lets focus on the left hand side of the eq. (2.8). We notice that u_t is the first derivative of u in the t-component, and from section 2.5 we know that a discretization of the first derivative can look like

$$u_t = \frac{1}{d}(U_{i,j}^{k+1} - U_{i,j}^{k}). \tag{3.13}$$

A condition that must be satisfied is $d/v^2 \leq \frac{1}{2}$, [6]. If this condition is not satisfied the solution for the problem will oscillate. Now $U_{i,j}^k$ can be described as the identity matrix I, Such that put together we get

$$U_{i,j}^{k+1} = (dD2 + I)U_{i,j}^{k} \tag{3.14}$$

This is the numerical method we will use to solve the heat equation in the experiments in Chapter 4.

3.2 Numerical method for linear elasticity equation

For a linear elastic problem we will work with later, we need to use a method for finding the displacement. In some of the experiments in this thesis we use the displacement to find the deformed MR image from one point in time to another. To simulate the experiments with the Navier Lame equations, we use two toolboxes the Matlab Reservoir Simulation Toolbox (MRST) and FV-Biot where the FV-Biot uses a discretization named multi point stress approximation(MPSA), and the MRST solve the non-linear set of discrete equation using automatic differentiation and Newtons method. The FV-Biot can be found her; https://github.com/pmgbergen/fvbiot .

Since the MPSA is build up from the Multi Point Flux Approximation(MPFA) and the Two Point Flux Approximation(TPFA) and to get an idea of how the method works, we will give a short presentation of the MPFA and TPFA. As the name imply the TPFA scheme uses two points to approximate the flux, where the MPFA scheme uses multiple points instead of just two. This allow the MPFA to get information from multiple cells to approximate the flux across each face. These methods approximate the flux between two cells by finding the cell transmissibility. For the TPFA the transmissibility can be determined by a harmonic average of the values transmissibilities between two neighbour cells. For the MPFA we consider a grid in a multidimension, where the potential is evaluated at the center of each cell. From each cell center a line is drawn to the midpoint of the cell surface, such that each face in the grid is subdivided into a set of subfaces, where one subface per node construct the face. The transmissibility coefficients of the subinterfaces inside the interaction region, is determined by the local interaction between the cells of the interaction region. The transmissibility coefficients of the cell interfaces are found by using the information from the subinterfaces that form a cell interface.

The MPSA is a more complex method than the MPFA and TPFA, and will not be derived in this theses. The full derivation of the method consider the *Cell-centered finite volume discretizations for deformable porous media* by Nordbotten, [10], and *Implementation of an MPFA/MPSA-FV Solver for the Unsaturated Flow in Deformable Porous Media* by Jhabriel Varela, [16].

3.3 Gradient descent optimization method

There exist multiple optimization methods to find a local minima, among them the gradient descent(GD) method. GD can be used to solve a system of both linear and non-linear equations, reformulated as a minimizing problem. The gradient method is a commonly used method. In the experiments we will look at in Chapter 4, we will use this method to find the minima of an optimization problem for the cost function. We will use this method in all of the experiments in this thesis. The theory in this section is based on the *Solving systems of nonlinear equations using a globally convergent optimization algorithm* by Taheri and Mammadov, [14].

The basic idea of the GD is that with an initial guess, p_0, of the variables, an optimization method generates a sequence p_k of improved estimates until it reaches a solution. When p_k is a finite sequence, the last point is the optimal solution. If p_k is infinite, it has a limit point which is the optimal solution of the problem. The strategy is to move from one iteration to the next using the iteration from earlier. A typical behaviour of an algorithm is that the iteration p_k move steadily towards the neighborhood of a local minimum point, and then rapidly converge to that point. When a given convergence rule is satisfied, the iteration will stop.

The GD takes steps proportional to the negative of the gradient of a function at the current point. The GD method is based on the observation that if a function, $u(p)$, is defined and differentiable in a neighborhood of a point p, then $u(p)$ decreases faster if one goes from p in the direction of the negative gradient of u at p, $-\nabla u(p)$. It follows that if

$$p_{m+1} = p_m - \alpha \nabla u(p_m) \tag{3.15}$$

for $\alpha \in \mathbb{R}_+$ small enough, then $u(p_m) \geq u(p_{m+1})$, where α is the step size. The term $\alpha \nabla u(p_m)$ is subtracted from p_m because we want to move in the negative direction of the gradient and toward the minimum. One starts with a initial guess, p_0, for a local minimum of u, and considers the sequence $p_0, p_1, p_2, ...$

The step size, α, can either be a constant, which is sufficiently small, or it can be updated with a line search, by finding the locally optimal step size α_m on every iteration. By letting the step size be a constant the method will be faster but can yield poor convergence. Conversely, by updating the step size every iteration, with certain assumptions of the function $u(p)$, convergence to a local minimum point is guarantied but this can be time-consuming.

In this thesis we will not use a linesearch for α, and therefore the step size will be constant. The step size must then be small enough such that the method does not jump or zigzags near the minima. This causes the method to use a lot of steps and therefore need a high number of iterations before the method get close to the minimum point. By picking a value for α the risk will be that if the step size is very small, it would take long time to converge and become computationally expensive. On the other side if α is large it may fail to converge and overshoot the minimum.

The GD method is a commonly used method. It has the globally convergence property. However, this method suffers from the slow speed and is easy plunging into local minima.

The best alternative would be to have an analytic expression for the gradient but this may be difficult to do with the cost functions used in the experiments we will look at later. Therefore we will use the numerically expression for the gradient as in eq. (3.16). Underneath is the analytic expression of the gradient we will use later in this thesis.

3.3.1 1 variable

For a function $u(p)$, the gradient of u is found as

$$\nabla u(p) = \frac{u(p+\delta) - u(p-\delta)}{\delta}. \tag{3.16}$$

for a small δ.

3.3.2 2 variables

Now for a function of two variables, $u(p_1, p_2) = u(p)$. We now have to update two variables. For the gradient the computation is the same, but updating only one variable at a time

$$\nabla u(p) = \begin{bmatrix} \frac{u(p_1+\delta_1,p_2)-u(p_1-\delta_1,p_2)}{\delta_1} \\ \frac{u(p_1,p_2+\delta_2)-u(p_1,p_2-\delta_2)}{\delta_2} \end{bmatrix} \tag{3.17}$$

In the experiments we will look at in Chapter 4, we will use this method to find the minima of an optimization problem for the cost function. This method will be used in all of the experiments later in this thesis.

3.4 Overview

Here is an overview of some of the important equations from this chapter.

Numerical method for the heat equation

$$U_{i,j}^{n+1} = (dD2 + I)U_{i,j}^{n} \qquad \text{(3.14 revisited)}$$

Gradient for $p \in \mathbb{R}^1$

$$\nabla u(p) = \frac{u(p_m + \delta) - u(p_m - \delta)}{\delta}. \qquad \text{(3.16 revisited)}$$

Gradient for $p \in \mathbb{R}^2$

$$\nabla u(p) = \begin{bmatrix} \frac{u(p_1+\delta_1,p_2)-u(p_1-\delta_1,p_2)}{\delta_1} \\ \frac{u(p_1,p_2+\delta_2)-u(p_1,p_2-\delta_2)}{\delta_2} \end{bmatrix} \qquad \text{(3.17 revisited)}$$

.

4 Experiments

In this chapter experiments relevant to the thesis is presented. The first section of this chapter contains basic experiments with the heat equation. Then we look at the linear elasticity equation and lastly we explore the real case. In the beginning of each section in this chapter we start with a simple experiment to investigate the framework. Then advancing step by step to more realistic experiments. The goal with the experiments is to find the breathing cycle by studying a series of MR images over a time interval.

For most of the experiments we follow a pattern on how they are carried out. We begin with simulating a forward problem. We introduce an equation from Chapter 2, and the boundary conditions we will be using in the experiment. Then solving the forward problem by using a numerical method from Chapter 3. Once we have the solution of the forward problem we can start with the inverse problem. Given some of the data from the solution of the forward problem, and assume we know the boundary conditions on 3 sides(we will only be working with square domains). Construction a cost function as described in section 2.4. Then the minimizing problem is to find a solution that minimizes the cost function. For this minimizing problem we use the Gradient Descent(GD) optimizing method from section 3.3.

In some of the experiments we use a function to describe the boundary on one side, in the forward problem. Then for the inverse problems we assume we know this function as well, and the goal for the inverse problem is to find the variables used in that function for the forward problem.

For all of the experiments in section 4.1 and 4.2 we want to investigate who far the data from the forward problem can be from the boundary before the GD method can not find a good approximation of the boundary.

4.1 Experiments; Heat equation

To build up the experiments with linear elasticity equation, we start with experiments using the heat equation. This section is structured as following; First we simulate a simple problem with constant boundaries, then solving the forward problem and the inverse problem as described before. Next we advance the experiment where we let one side of the boundary be a function of two variables. Then for the inverse problem we assume we know the function and the goal is to find an approximation of the variables used in the forward problem.Then advancing again such that we only consider one part of the domain on the boundary. We also do an experiment with added noise to the solution from the forward problem.

The heat equation is solved for 100 iterations for all of the experiments in this section. All of the experiments in this section will take the same form;

First simulate a forward problem. Consider a two dimensional space $\Omega = (0,1) \times (0,1)$ and let $O(u)$ and f form eq. (1.1) be $O(u) = u_t - \kappa \Delta u$ and $f = \psi$. This is the heat equation from section 2.1, given as

$$\rho u_t = k\Delta u + \psi. \tag{2.8 revisited}$$

For a two dimensional space $\Delta u = u_{xx} + u_{yy}$. Let $\rho = 1$, $k = -1$ and $\psi = 0$. The boundary conditions from eq. (1.2) will be specified in the experiments. Then using the numerical method from section 3.2

$$U_{i,j}^{n+1} = (dD2 + I)U_{i,j}^{n} \tag{3.14 revisited}$$

with the operator D_2 from section 2.6.

For the inverse problem consider some data from the solution of the forward problem. We construct a cost function as described in section 2.4 using the data from the forward problem. Assuming we know the boundary on 3 of the sides of the domain, the minimizing problem described in eq. (2.34) is solved by using the GD method from section 3.3.

4.1.1 Ex. 1.1

As a first experiment we want to demonstrate the layout of the framework for a simple problem. As described before we use the heat equation for all of the experiments in this section.

For the forward problem let the boundary conditions from eq. (1.2) be Dirichlet boundary conditions, with zero on all of the boundaries, except for on the left side, $x = 0$. For $x = 0$ we use one, such that

$$B[u] = \begin{cases} u(x,y,t) = 0 & \text{for } y = 0, y = 1 \text{ and } x = 1 \\ u(x,y,t) = 1 & \text{for } y \in (0,1) \text{ and } x = 0 \end{cases} \tag{4.1}$$

Solving the heat equation with the given Dirichlet boundary conditions, we use the numerical method discussed in section 3.1 in eq.(2.43). The parameters

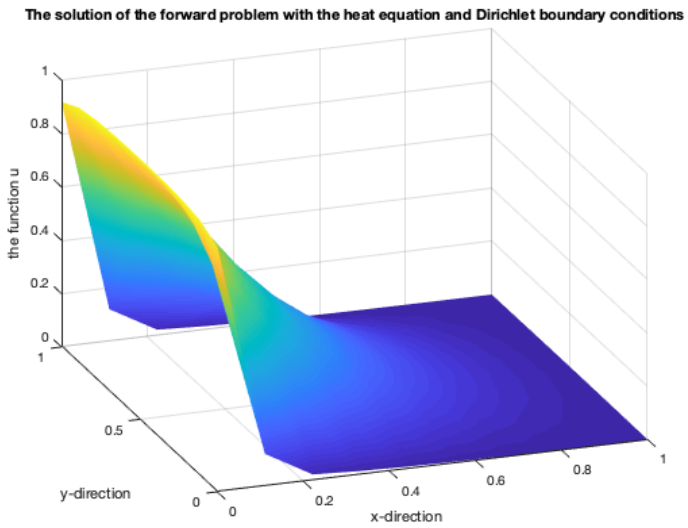


Figure 4: The solution of the heat equation with Dirichlet boundary conditions.

used in this experiment are $n = 10$, $v = 0.1$ and $d = 0.001$. In Figure 4 we see the solution of the heat equation, with the boundaries as described above.

Now for the inverse problem. Assume we know the boundary conditions from eq. (1.2) is zero on all of the boundaries, except for on the left side, $x = 0$. For the boundary condition on $x = 0$ we have to find an approximation.

From the forward problem we are only given one vector from the solution. we will investigate how far this vector can be from the boundary, before the GD method can not find a solution close to the boundary condition. We consider the vector at row h, called it P, from the forwrd problem. Then the cost function for this problem is given as $c(p) = ||P - \tau(p)||$, where the function $\tau(p)$ solved the heat equation with the given boundary condition p and returns the vector at row h in the approximated solution. From eq. (2.34), we have the minimization problem

$$\min c(p) \tag{4.2}$$

We use the GD method to solve this minimization problem, where the gradient is given as the derivative of the cost function. We find the gradient by using the eq.(3.16)

$$\nabla c = \frac{c(p + \delta) - c(p - \delta)}{\delta} \tag{4.3}$$

The parameters used are $\delta = 0.01$, $n = 10$, the initial guess is a random value, $p_{guess} = 5$, the step size must be small enough to not zigzag, $\alpha = 0.5$, the maximum iterations is $maxiter = 500$ and the tolerance is $tol = 0.001$.

As the GD is explained in section 3.4, we use the GD to minimize the minimizing problem and to find an approximation of the boundary. First the method will use the initial guess as p, then using the approximated boundary to solve the heat equation. Next we compare the vector at row h in the approximated solution and the given vector from the forward problem, in the cost function. This is repeated until the cost function is small or when the GD have used maximum iteration.

we can see from the table underneath that when the data from forward problem is further away form the boundary the GD method gets more inaccurate

h	The difference between the true and the approximated solution
3	0.0729
5	0.1703
8	0.9215

4.1.2 Ex. 1.2

Now we want to advance the experiment from before. Instead of a boundary of a constant we use a sinus function on $x = 0$. We use the same predictions as we did in Experiment 1.1.

As described, the forward problem use the heat equation from eq. (2.8). For a vector $P \in \mathbb{R}^2$, $P = [P1, P2]$, let the boundary conditions from eq. (1.2) be given as

$$B[u] = \begin{cases} u(x, y, t) = 0 & \text{for } y = 0, y = 1 \text{ and } x = 1 \\ u(x, y, t) = K(P) & \text{for } y \in (0, 1) \text{ and } x = 0 \end{cases} \tag{4.4}$$

where $K(P)$ is a function $\mathbb{R}^2 \to \mathbb{R}^n$, given as $K(P) = P1 \sin(P2y)$. Let $P = [1, 3\pi]$. The other parameters the same as in Experiment 1.1. Then using the numerical method from eq.(2.43) in section 3.1, with the operator D_2 from eq. (2.51) to solve the heat equation with the boundary conditions described above. From Figure 5 we see the solution from the forward problem.

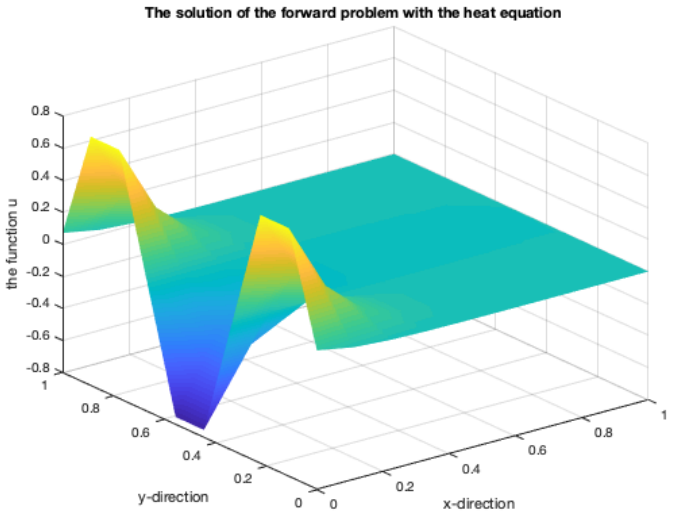


Figure 5: The solution of the forward problem with the heat equation for a boundary given as a sinus function.

Now for the inverse problem, assuming we know the boundaries for $y = 0, y = 1$ and $x = 1$. Also assuming that we know the boundary is a sinus function as described K for $x = 0$, and we only want to find the amplitude and the period (represented in P), for the boundary on the left side.

As before we are only given one vector from the solution of the forward problem. Also in this experiment we will investigate how far this vector can be from the unknown boundary, before the method can't find a solution close

to the true boundary condition. We consider the vector at row h, called P from the forward problem. Then the cost function for this problem is given as $c(p) = ||P - \tau(K(p))||$. Now the function $\tau(K(p))$ solves the heat equation the the given amplitude and period, and returns the vector at row h from the approximated solution. From eq. (2.34), we have the minimization problem

$$\min_p c(p) \tag{4.5}$$

We use the GD method from section 3.3 to solve this minimization problem, where the gradient is given as the derivative of the cost function. We find the gradient by using a centered numerical as in eq.(3.17), where $\delta = 0.1$. The other parameters used are the $n = 10$, the initial guess is a vector, $p_{guess} = [2\ 2\pi]$, the step size $\alpha = 0.1$, the maximum iterations $maxiter = 500$ and the tolerance $tol = 0.1$. Here we use the GD to find an approximation of p, instead of the whole boundary, such that $p_{m+1} = p_m - \alpha\nabla c$. Recall from section 3.3 that we need to update the gradient for one variable of p_m at a time.

In the table underneath we see the difference between the solution and the approximated solution when the information we get is from the vector at row h in the forward problem;

Row h	The difference between the true and the approximated boundary
3	0.5082
5	2.9979
8	3.2969

Also n this experiment we see that when the data from the forward problem is further away form the boundary the method gets more inaccurate. We also see that the difference is larger than in the experiment from before.

4.1.3 Ex. 1.3

When we do the experiments in section 4.3 with the MR images over a time interval we only consider one side of the abdominal are. This is because we could expect that the left lung and right lung would have the same breathing cycle.

To simulate a simpler experiment before considering the experiments with the linear elasticity equation, we examine an experiemetn with the heat equation and only half of the domain Ω for the cost function. In this experiment we will see how the framework for the inverse problem, effects the result when only a part of the domain Ω is considered in the cost function.

Consider the same boundary conditions as in Experiment 1.2, for a vector $P \in \mathbb{R}^2$, $P = [P1, P2]$, the eq. (1.2) be given as

$$B[u] = \begin{cases} u(x,y,t) = 0 & \text{for } y = 0, y = 1 \text{ and } x = 1 \\ u(x,y,t) = K(P) & \text{for } y \in (0,1) \text{ and } x = 0 \end{cases} \tag{4.6}$$

where $K(P)$ is the same function as in Experiment 1.2. Also in this experiment let $P = [1, 3\pi]$. Solving the heat equation with the given Dirichlet boundary conditions, we get the same result as in Figure 5

As in Experiment 1.2 we assume we know the boundary conditions is zero on all of the boundaries, except for $x = 0$, and that the boundary is the function $K(P)$ and we only want to find the amplitude and the period, for the boundary on the left side.

From the forward problem we are only given one vector at row h, from the solution. Here denoted as P_{sub}. Where P_{sub} is the subvector of the vector at row h in the forward problem, for the interval $[0, 0.5]$ in the y-direction.

Then the cost function for this problem is given as $c(p) = ||P_{sub} - \tau(K(p))||$. From eq. (2.34), we have the minimization problem

$$\min_p c(p) \tag{4.7}$$

We use the GD method to solve this minimization problem, where the gradient is given as the derivative of the cost function, from eq. (3.17). The parameters are the same as in Experiment 1.2.

In the table underneath we see the difference between the solution and the approximated solution when the information we get is from the vector at row h in the forward problem;

Row h	The difference between the true and the approximated boundary
3	1.6081
5	2.6918
8	3.2969

We can see from this table that when we only look at half of the vector from the forward problem we have less information and therefore we may get a larger

error than before when the vector form the forward problem is close to the boundary. However of we look at a vector further away from the boundary the error is almost the same as from Experiment 1.2.

From Figure 6 we see how the approximated solution is. The approximation is close to the solution for the subdomain considered, but for the other half the there is not enough information for the approximated solution to be accurate.

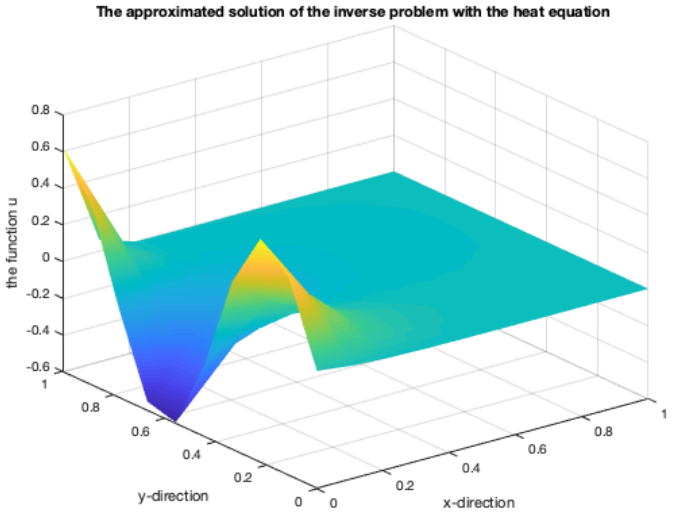


Figure 6: The approximated solution of the heat equation for a vector at row $h = 5$ in a subdomain.

4.1.4 Ex. 1.4

To make the experiments more realistic we can add noise to the data from the solution of the forward problem described in eq. (2.31). In this experiment we see how noise in the cost function effects the result. We would expect the difference between the approximated solution and the solution from the forward problem, to get larger when added noise to the given data when investigate the inverse problem.

We use the same predictions as we did in experiment 1.3, and reuse the same forward problem. We get the same solution as in Figure 5.

Investigating the inverse problem, We already know the boundary conditions from eq. (1.2) be zero on all of the boundaries, except for $x = 0$, and given the function K. We consider the vector at row h, called P, from the forward problem. Let the cost function be defined as in section 2.4

$$\min_p c(p) = ||P_\eta - \tau(K(p))|| \tag{4.8}$$

We add 10% noise to P. Then P_η is P with added noise, $P_\eta = P + \eta$ where $\eta = 10P\ rand(n, 1)/100$;.

We use the GD method to solve this minimization problem, where the gradient is given as in eq. (3.17). We use the same parameters as in Experiment 1.3.

In the table underneath we see the difference between the solution and the approximated solution when the information we got from the vector at row h in the forward problem, with added noise;

Row h	The difference between the true and the approximated boundary
3	1.5756
5	2.6902
8	3.2969

As expected the difference between the true solution and the approximation gets larger with a percentage of noise. We can see from the table that when the vector we get the information from is further away form the boundary the method gets more inaccurate. We also see that the difference is larger than in the experiment from before.

From Figure 7 we see that for added noise to the given data impact the approximated solution and makes it more inaccurate.

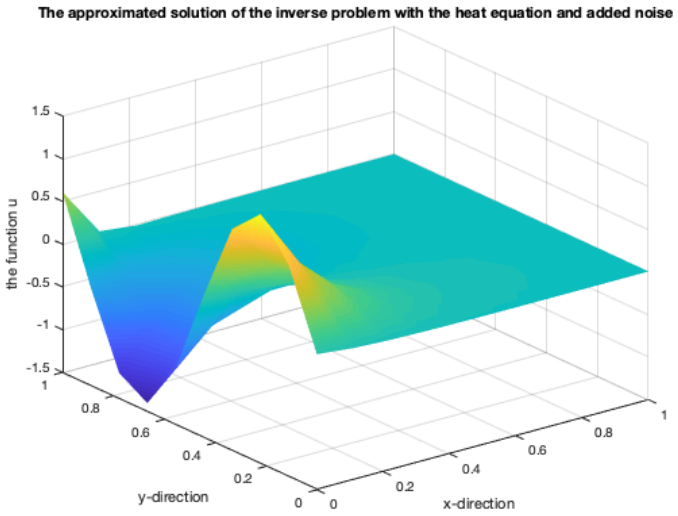


Figure 7: The approximated solution of the heat equation and added noise to the vector at row $h = 5$ from the forward problem.

.

4.2 Experiments; Linear elasticity equation

In this section we will repeat some of the experiments from section 4.1 only now we use the linear elasticity equation from section 2.2. this section is structured the same way as section 4.1; An experiment whit constant boundaries, then an experiment with a function describing one side of the boundaries. Then advancing the inverse problems by only considering a part of the domain of the data from the forward problem, and lastly we examine an experiment where we add noise to the data from the forward problem, for the inverse problem.

All of the experiments in this section is build the same way

For the forward problem we consider the domain of the unit square, $\mathbb{R}^2$, $\Omega = (0,1) \times (0,1)$ and the Navier-Lame equation discussed in section 2.2

$$u_{tt} = \frac{\lambda + \mu}{\xi} \nabla(\nabla \cdot u) + \frac{\mu}{\xi} \Delta u \qquad \text{(2.28 revisited)}$$

Form eq. (1.2) let the boundary conditions be Neumann boundary conditions on the sides, for $y = 0$ and $y = 1$, and Dirichlet on the bottom and top, $x = 0$ and $x = 1$, such that

$$B[u] = \begin{cases} \nabla u = 0 & \text{for } y = 0, y = 1 \\ u = 0 & \text{for } x = 0 \end{cases} \tag{4.9}$$

The top boundary condition will be specified in the experiments. Then using the MPSA method, shortly introduced in section 3.2, to find the displacement to the forward problem.

For the inverse problem consider some data from the solution of the forward problem, also assume we know the boundaries on 3 sides of the domain. Construction a cost function as described in section 2.4 by using the data from the forward problem. Then the minimizing problem is as described in eq. (2.34). We use the GD method form section 3.3 to solve the minimization problem.

4.2.1 Ex. 2.1

In the first experiment for the linear elasticity equation we want to illustrate the layout of the framework for the forward problem and for the inverse problem with Navier-Lame equation.

For the forward problem consider the boundary conditions described in the introduction of this section, and for the top boundary condition we let the y-component for the faces on the left side of the top of the grid be $uy = 1$.

Constructing a grid with the given boundary condition. Then solve the forward problem by using the MPSA, to find the displacement. From Figure 8 we see the magnitude of the displacement of the forward problem.

From the forward problem we are only given two vectors from the solution, the displacement in the x-direction, UT and in the y-direction VT, at the row h. In this experiment, as we did in the last, we investigate how far these vectors

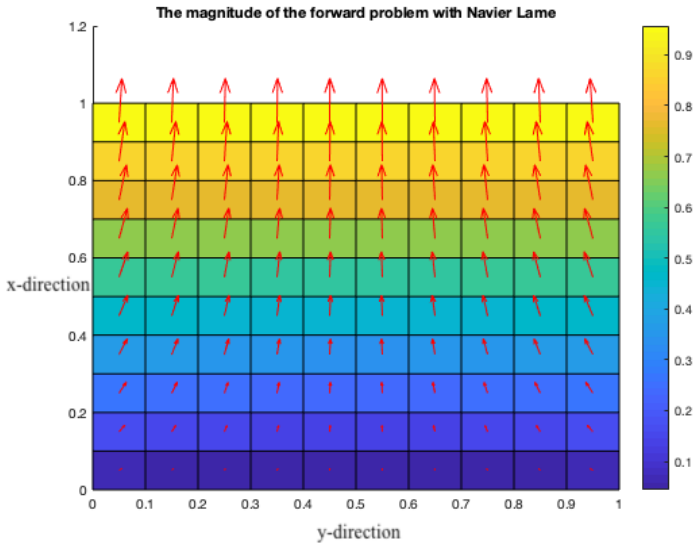


Figure 8: The magnitude for the forward problem with Navier-Lame with mixed boundary condition.

can be from the boundary before the GD optimization method can not find a solution close to the true boundary condition.

Assuming we know the boundary on the left and right side and on the bottom as described in the introduction of this section. We also assume that we know that the boundary on the top is a Dirichlet condition. We want to find an approximation of the y-component for the faces on the left side of the top of the grid.

We construct the cost function described in section 2.4 as

$$c(p) = ||(UT - U(p))^2 + (VT - V(p))^2|| \tag{4.10}$$

where $[U, V]$ is the displacement of the approximation p of the y-component for the faces on the left side of the top of the grid, at row h in the approximated solution.

The parameters used are $n = 10$, the initial guess - a random value, $p_{guess} = -3$, the step size - must be small enough to not zigzag, $\alpha = 0.01$, the maximum iterations - $maxiter = 500$ and the tolerance - $tol = 0.1$.

From eq. (2.34) we get the optimization problem of the cost function

$$\min_p c(p) \tag{4.11}$$

We use the GD method to solve the optimization problem where the gradient is the derivative of the cost function. Here we use a approximation of the derivative as in eq.(3.16)

$$\Delta c = \frac{c(p+\delta) - c(p-\delta)}{\delta} \tag{4.12}$$

Solving the inverse problem using the GD method. The iterative method stops when the cost function is small or when the method have taken maximum number of iteration.

The result we get can be seen in the table underneath

h	The difference between the true and the approximated solution
3	$7.8515 * 10^{-13}$
5	$4.9840 * 10^{-6}$
8	0.9476

In the table we see the result from the experiment. Note the results are much better in this experiment than from all of the experiment from earlier.

4.2.2 Ex. 2.2

Advancing Experiment 2.1 for the linear elasticity equation. We now want to use a sinus function for the boundary instead of a constant. We will use the same predictions as in Experiment 2.1.

For the top boundary condition for the eq. (1.2), we let the y-component for the faces on the left side of the top of the grid be $uy = K(P)$ for a vector $P = [P1, P2]$. Where the function $K(P)$ is the same function from Experiment 1.2, and is denoted as $K(P) = P1sin(P2y)$. Let $P = [1, 3\pi]$

The grid size is $(n \times n)$ where $n = 10$. Constructing the grid and solve the forward problem by using the MPSA from section 3.2, and find the displacement. From Figure 9 we see the magnitude of the displacement of the forward problem.

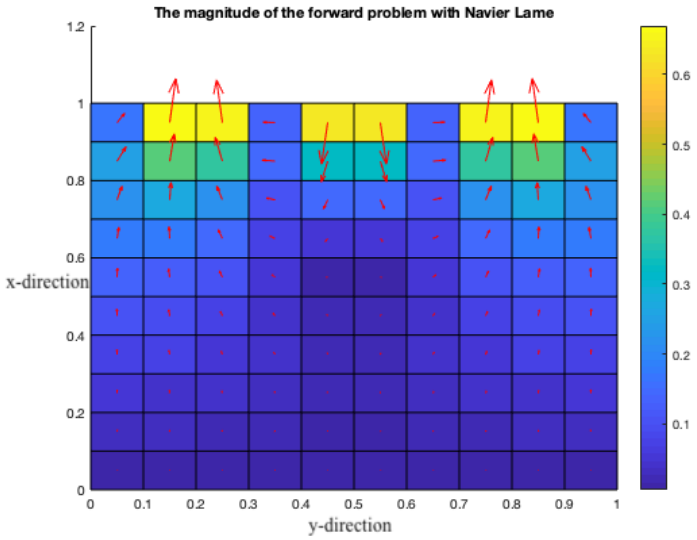


Figure 9: The magnitude for the forward problem with Navier-Lame and a sinus function on one side of the boundary.

In the inverse problem we want to find an approximation of the amplitude and the period of the y-component for the faces on the left side of the top of the grid, assuming we know the boundary conditions on the other sides and the function K.

We construct the cost function described in section 2.4 as

$$c(p) = ||(UT - U(K(p)))^2 + (VT - V(K(p)))^2|| \tag{4.13}$$

where $[U, V]$ is the displacement from the approximated solution at row h.

From eq. (2.34) we get the optimization problem of the cost function. We use the GD method from section 3.3 to solve the optimization problem, where

the gradient is the derivative of the cost function. Here we use an approximation of the derivative as in eq. (3.17). The parameters used are $n = 10$, the initial guess is $p_{guess} = [2, 2\pi]$, the step size must be small enough to not zigzag, $\alpha = 0.01$, the maximum iterations $maxiter = 500$ and the tolerance $tol = 0.1$.

The result we get after the GD method can be seen in the table underneath

h	The difference between the true and the approximated solution
3	2.0928
5	2.7685
8	3.2969

As expected we see that the error is larger in this experiment than in Experiment 2.1.

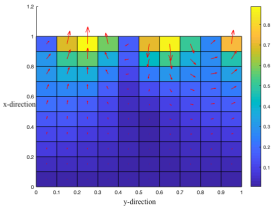


Figure 10: h=3

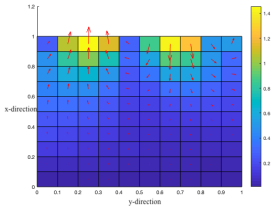


Figure 11: h=5

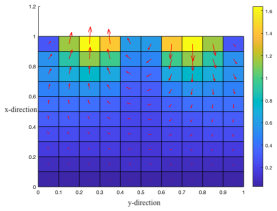


Figure 12: h=8

In Figure 10-12 we see the magnitude of the approximated solution for when the data from the forward problem is from row $3, 5$ and 8. There is not much difference between the figures, however these figures are different from the Figure 9.

4.2.3 Ex. 2.3

Advancing Experiment 2.2 by consider only half of the domain Ω for the cost function.

We consider the same forward problem as in Experiment 2.2, and we get a solution as in Figure 9

As before we want to find an approximation of the amplitude and the period for the y-component for the faces on the left side of the top of the grid. From the forward problem we have the displacement $[UT, VT]$ from row h.

We construct the cost function described in section 2.4 as

$$c(p) = ||(UT_{sub} - U(K(p)))^2 + (VT_{sub} - V(K(p)))^2|| \tag{4.14}$$

where $[U, V]$ is the displacement at row h from the approximated solution and $[UT_{sub}, VT_{sub}]$ are the subvectors of the displacement from the forward problem. The subdomain is $(0, 0.5)$.

From eq. (2.30) we get the optimization problem of the cost function. We use the GD method to solve the optimization problem where the gradient is the derivative of the cost function as in eq. (3.17). Using the same parameters an in Experiment 2.2. The result we get after the GD method can be seen in the table underneath

h	The difference between the true and the approximated solution
3	2.9130
5	3.2829
8	3.2969

the difference between the solution from the forward problem and the approximated solution is large. As from Experiment 1.3 we see the same happens here.

4.2.4 Ex. 2.4

To make the Experiment 2.3 more realistic we can add noise as in eq. (2.31). In this experiment we will look at how added noise in the displacement in the data from the forward problem will affect the result.

We repeat the forward problem as in Experiment 2.1, with the same domain and the same boundary conditions as before. The solution from the forward problem can be sen in Figure 9.

As before we want to find an approximation of the amplitude and period of the y-component for the faces on the left side of the top of the grid.

We construct the cost function described in section 2.4 as

$$c(p) = ||(UT_\eta - U(p))^2 + (VT_\eta - V(p))^2|| \tag{4.15}$$

where $[U, V]$ is the displacement of the approximated solution and $[UT_\eta, VT_\eta]$ are the subvectors of the displacement at row h from the forward problem with added noise. As before we add 10% noise such that $[UT_\eta = 10UT\ rand(n,1)/100, VT_\eta = 10VT\ rand(n,1)/100]$

From eq. (2.30) we get the optimization problem of the cost function and use the GD method to solve the optimization problem where the gradient is given as in eq.(3.17). We use the same parameters as in Experiment 2.3.

The result we get after the GD method can be seen in the table underneath

h	The difference between the true and the approximated solution
3	2.9157
5	3.2969
8	3.2969

The are not any difference from using a vector form row $h = 5$ and $h = 8$ of the forward problem. This is because the difference is too small that far away from the boundary.

From Figure 13 we see the magnitude of the displacement of the approximated solution of the boundary when the data from the forward problem is from row $h = 5$. As we can see it is not too similar to the solution from the forward problem, in Figure 9.

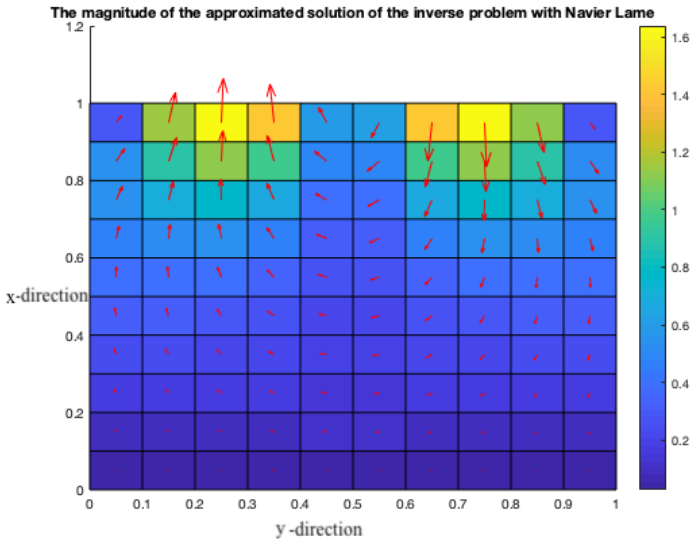


Figure 13: The magnitude for the approximated solution for the inverse problem with Navier-Lame with mixed boundary condition. This is the approximated solution for the data at row $h = 5$ from the forward problem.

.

4.3 Experiments: Real cases

This section will investigate real MR images. All of the MR images we use in this thesis are representing the abdominal area. In this section we start by simulating a problem with one MR image. We explore an experiment with a constant boundary and then an experiment with a boundary described with a function. Lastly we consider a real case with MR images deforming over a time interval. In this experiment we only consider the inverse problem, this is because in a real case we do not know the solution to the forward problem. However with the background from all of the experiments before we could expect the GD method to find an approximation almost as good as from the experiments before.

We will continue using the linear elasticity equation described in section 2.2.The first two experiments in this section is build the same way

For the forward problem we consider the Navier-Lame equation discussed in section 2.2

$$u_{tt} = \frac{\lambda + \mu}{\xi} \nabla(\nabla \cdot u) + \frac{\mu}{\xi} \Delta u \qquad \text{(2.28 revisited)}$$

Form eq. (1.2) let the boundary conditions be the same as before, Neumann boundary conditions on the sides, for $y = 0$ and $y = 1$, and Dirichlet on the bottom and top, $x = 0$ and $x = 1$, such that

$$B[u] = \begin{cases} \nabla u = 0 & \text{for } y = 0, y = 1 \\ u = 0 & \text{for } x = 0 \end{cases} \tag{4.16}$$

where top boundary condition will be specified in the experiments. Also in these experiments we use the MPSA method, from section 3.2, to find the displacement to the forward problem.

For the inverse problem consider some data from the solution of the forward problem, assuming we know the boundaries on 3 sides of the domain. Construction a cost function as described in section 2.4 by using the data from the forward problem. Then the minimizing problem is as described in eq. (2.34). We use the GD method form section 3.3 to solve the minimization problem.

The MR images are

4.3.1 Ex. 3.1

For the experiments with MR images, consider the image shown in Figure 14. Here we consider a image $M0$. For the forward problem we use a force(boundary condition) to deform the image to another image called M. Then for the inverse problem we want to use the image $M0$ and M to find the force acting on $M0$ such that it deforms to M.

As from the experiments before we will first simulate a forward problem using the Navier-Lame eq. (2.28). Then solving the forward problem using the numerical method from section 3.2. For the inverse problem we use the data from the forward problem and the framework from section 2.3. construction a

cost function as described in section 2.4 and then minimize the optimization problem by using the GD method from section 3.3.

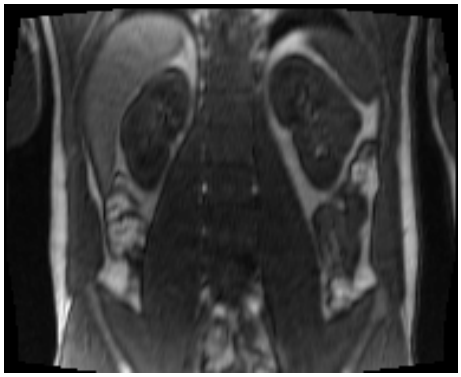


Figure 14: MR image of the abdominal area, [1]

To simplify, we will look at one side of the spine (the left side). Also, since the image has $[192 \times 156]$ pixels, the method will use a lot of time if we where to use the image with the original pixel. Therefore we resize the image to have (50×50) pixels. To avoid local minimum points we will use a Gauss filter on the images.

For the forward problem let $M0$ be the MR image from Figure 14. Form eq. (1.2) let the boundary conditions for the top of the image be described with the y-component for the faces on the left side of the top of the grid, be $uy = 1$.

Construction a grid with the given boundary condition and $n = 50$. Solving the forward problem using the MPSA from section 3.3 to find the displacement. Then using the displacement to find the deformed image M.

For the inverse problem we use the deformed image M. As before we assume that we know the boundaries on the sides and the bottom. Also assume we know that the boundary on the top is a Dirichlet boundary. For the inverse problem we want to find an approximation p of the y-component for the faces on the left side of the top of the grid.

What we want to do in the inverse problem, is to use M and $M0$ to find p, such that the two images, M and $M0$, will be the same. This means that we want $p \approx uy$ or ideally $p = uy$.

We construct the cost function as described in section 2.3

$$c(p) = ||M0 - M(p)|| \tag{4.17}$$

We get the minimization problem from the eq. (2.34)

$$\min_p c(p) \tag{4.18}$$

We will use the GD to find the minimum, as before, where the gradient is

given as the derivative of the cost function

$$\Delta c = \frac{c(p+\delta) - c(p-\delta)}{\delta} \tag{4.19}$$

with $\delta = 0.1$. The other parameters used in this experiment are $n = 50$, initial guess is a random number $p_{guess} = 2$, step size $\alpha = 0.01$, maximum iteration $maxiter = 500$ and the tolerance -$tol = 0.1$. The Gradient descent will stop after a given number of iterations, $maxiter$, or if the cost function is smaller than a tolerance, tol. The result from the method is $p = 0.9293$ which makes the difference between the true solution and the approximated solution 0.0707. Comparing the two images we see the result in Figure 15. In Figure 15 we see

Figure 15: Comparing the two images, $M0$ and the image for the approximated solution M.

that the images are almost identical.

4.3.2 Ex. 3.2

We advance Experiment 3.1. We want to use a sinus function for the boundary instead of a constant.

As from the experiments before we will first simulate a forward problem using the framework from section 2.3. and the Navier-Lame eq. (2.28). Then solving the forward problem using the numerical method from section 3.3. For the inverse problem we use the data from the forward problem and the framework from section 2.3. construction a cost function as described in section 2.4 and then minimize the optimization problem by using the GD method from section 3.4.

We use the same predictions as in Experiment 3.1

For the forward problem consider the domain of the unit square, $\Omega = (0,1) \times (0,1)$. Let $O(u)$ and f form eq. (1.1) be $O(u) = u_{tt} - \frac{\lambda+\mu}{\rho}\nabla(\nabla \cdot u) - \frac{\mu}{\rho}\Delta u$ and $f = 0$. This gives us the Navier-Lame equation

$$u_{tt} = \frac{\lambda+\mu}{\xi}\nabla(\nabla \cdot u) + \frac{\mu}{\xi}\Delta u \qquad \text{(2.28 revisited)}$$

Form eq. (1.2) let the boundary conditions be Neumann boundary conditions on the sides, for $y = 0$ and $y = 1$, and Dirichlet on the bottom and top, $x = 0$ and $x = 1$, such that

$$B[u] = \begin{cases} \Delta u = 0 & \text{for } y = 0, y = 1 \\ u = 0 & \text{for } x = 0 \end{cases} \tag{4.20}$$

for the top boundary condition we let the y-component for the faces on the left side of the top of the grid be $uy = K(P)$ for a vector $P = P1, P2$. Where the function $K(P)$ is $\mathbb{R}^2 \to \mathbb{R}^n$ and is denoted as $K(P) = P1sin(P2y)$. Let $P = [1, 3\pi]$

Construction a grid with the given boundary condition and $n = 50$. Solving the forward problem using the MPSA from section 3.3 to find the displacement. Then using the displacement to find the deformed image.

For the inverse problem we use the deformed image M. As before we assume that we know the boundaries on the sides and the bottom. Also assume we know that the boundary on the top is a Dirichlet boundary. For the inverse problem we want to find an approximation of the y-component for the faces on the left side of the top of the grid, denoted as p.

What we want to do in the inverse problem, is to use M and $M0$ to find p, such that the two images, M and $M0$, will be the same. The parameters used in this experiment are $n = 50$, initial guess is $p_{guess} = [11.5\pi]$, step size $\alpha = 0.01$, maximum iteration $maxiter = 500$ and the tolerance $tol = 0.1$.

We construct the cost function from section 2.4

$$c(p) = ||M - M(p)|| \tag{4.21}$$

We get the minimization problem from eq. (2.34)

$$\min_p c(p) \tag{4.22}$$

We will use the GD to find the minimum, as before, where the gradient is given as the derivative of the cost function

$$\Delta c = \frac{c(p+\delta) - c(p-\delta)}{\delta} \tag{4.23}$$

with $\delta = 0.1$.The Gradient descent will stop after a given number of iterations or if the cost function is small.

The result from the method is $[0.4524, 4.7356]$. This gives a difference 0.5481 when using the norm of the difference between the solution from the forward problem and the approximated solution.

Comparing the two images we see the result in Figure 16

Figure 16: Comparing the two images, M and the image for the approximated solution.

4.3.3 Ex. 3.3

Finally, we will look at the real case with multiple MR images over a time interval. From the experiments before we know that the GD method works well for data from the forward problem that are not so far away from the boundary, although when the data is far away from the boundary the GD method is not capable to fond a approximation close to the solution. In this experiment we don't know the solution to the forward problem. However, form all of the experiments we have looked at before we could expect that also in this experiment the method will find an approximation not too far away from the solution.

In this inverse problem we use the framework from section 2.3 and the Navier-Lame eq. (2.28). Starting with construction a cost function by using the start image $M0$, and the deformed image M. Then solving the minimization problem using the GD method.

In this experiment we consider the first MR image from the sequence, denoted as $M0$, and the next MR image $M1$. We want to find the force from the lungs such that the the image $M0$ is deformed to $M1$ by using the GD method. Then considering the MR image after $M1$, called $M2$. We use the GD method to find the force that deformed then $M0$ to $M2$, and so on. In this experiment we will look at 27 MR images. We will resize the images to (50×50) and use a Gauss filter as in Experiment 3.2.

As in Experiment 3.2 we assume that we know the boundaries on the sides and the bottom such that the eq. (1.2) is Neumann boundary conditions on the sides, for $y = 0$ and $y = 1$, and Dirichlet on the bottom, $x = 0$, such that

$$B[u] = \begin{cases} \Delta u = 0 & \text{for } y = 0, y = 1 \\ u = 0 & \text{for } x = 0 \end{cases} \tag{4.24}$$

where u is the solution to the forward problem. Also assume we know that the boundary on the top is a Dirichlet boundary and that the boundary on the left side of the top of the grid is described with a function $K(P) = P(1)\sin(P2y)$.

For the inverse problem we want to find an approximation of the y-component for the faces on the left side of the top of the grid, denoted as p. For each of the images we use the approximated solution from the image before as the initial guess for that image. We let the first initial guess be $p_{guess} = [1, 1.5\pi]$, $n = 10$ and the step size be $\alpha = 0.01$. Let the maximum iterations for the GD method be $maxiter = 100$ and the tolerance for the cost function $tol = 0.1$. Let the grid size be $(n \times n)$.

Construction the cost function

$$c(p) = ||M0 - M(K(p))|| \tag{4.25}$$

where $M0$is the first image of the sequence. we then get the minimization problem described in eq. (2.34)

$$\min_p c(p) \tag{4.26}$$

As before we use the GD method to minimize eq.(4.26). We do the iteration for all the images creating a vector W, where $W_0 = p_{guess}$. This vector will be an expression for the berating cycles/amplitude and period of the breathing cycle.

We use GD to find the approximated solutions for the whole sequence.

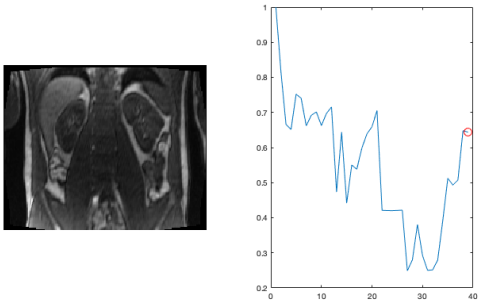


Figure 17: The last image of the MR image sequence and the amplitude of the approximated solution.

The function K with the approximated solution in W is an approximation of the breathing cycle. In Figure 17 we plot the approximated solution for the amplitude and period. Since we do not know the solution to this inverse problem we can not know if the approximated solution is close to the true solution. However, from the Figure 17 we can see that the result is moving up and down. We could expect the vector p to move down for when the person is exhaling and moving up when the person is inhaling. If we look at the images from Figure 18, we see that there is not much deformation from each image. However if we consider the approximated solution alongside the MR images, we see that the approximated solution matches the breathing cycle at some point. As expected we see that there is a delay or disruptions. This can be because of the step size or lack of a linesearc or the interval of the brute force. Also from the experiments before we know that if the image is fare away from the starting image we know that the method does not find an approximation close to the solution.

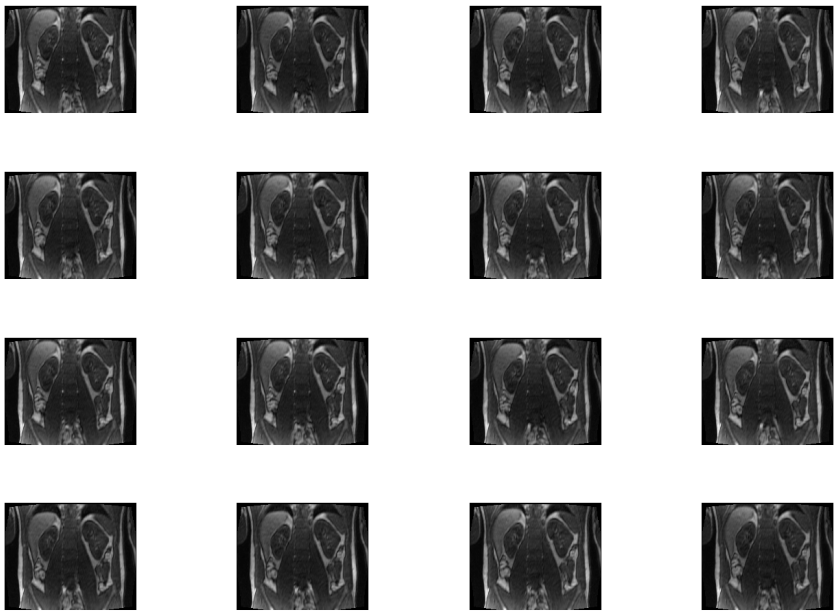


Figure 18: the first 16 images in the sequence of the 27 MR images

.

5 Discussion

Here we discuss the results from the experiments and some of the limitations with the framework in this thesis. Finally we conclude the work and comment on further work.

In general for the experiments in section 4.1 with the heat equation, some of the advantages and limitations with the GD method is shown clearly. The proposed framework is fast and accurate for problems where the data from the forward problem is some what close to the boundary we want to find. However when the data is further away from the boundary the error gets large fast. This may be because there exist multiple solutions that fits the model as discussed in section 2.3.

The framework we use in this thesis have the same struggles for the experiments in section 4.2, using the Navier-Lame and the MPSA to find the displacement, as for in the experiments in section 4.1. Also when we make the experiments more realistic by adding noise as described in section 2.3, the results gets more inaccurate and struggles with finding an approximation even close to the solution.

In section 4.3 we expected the framework to struggles more with the MR image and therefore we resized the images and added a Gauss filter. This is because the GD method would have used a lot of time for a image with pixels $[192 \times 156]$. We added the Gauss filter to avoid local minimum points. In spite of these precautions, we see from the Experiment 3.3, that the approximated solution does not represent the berating cycle as good as hoped. This may be because the images has a lot of noise or there exist multiple solutions that fits the model to represent the data.

in general, result from the experiments where we know the true solution, we see that the error varies a lot for the experiments in Chapter 4 and the framework finds both approximations close to the true solution and not so close. For the experiment using the MR image sequence, Experiment 3.3, where we don't know the true solution, there is no guarantee that we find an approximated close the true solution.

Some of the limitation to the experiments in this thesis are that we do not have an analytic expression of the gradient. We use a numerical approximation of the gradient for all of the experiments in Chapter 4. Another point to emphasize is that if the function is almost flat, the gradient is small and the GD method will use a lot of steps to get closer to the minimum point. This could be fixed with a larger step size. However if the step size is to large the GD method can zigzags or jumps. If this happens the GD method may not come close to the solution. Another limitation in this thesis is that in section 4.3 with the experiments with the MR images, we resize the images to have fewer pixels. This could also have an impact on the accuracy of the approximated solution.

5.0.1 Conclusion

All in all the framework in this thesis has potential. The proposed framework is fast and finds an approximated solution close to the true solution if the conditions are good. It is clear that the proposed method have some limitations which makes it slow or inaccurate for some of the experiments. This reflects in the experiments with the MR images. To overcome some of these obstacles we implement a Gauss filter and resized the images. This seems to help on the experiment where we look at one MR image, and the approximated solution is close to the true solution. The experiment with the sequence of MR images over a time interval, where we don't know the true solution, there is no guarantee of finding the true solution or get close to it. We see that the approximation is representing the images well but but it has some sort of delay or disruptions. This could be because of some of the limitations the proposed framework have or it could be because of some of the assumptions, f.ex. resizing the images or not using a linesearch.

To improve the experiments in Chapter 4 we could implement a linesearc or choose a smaller step size and a higher number for the maximum iterations. The next approach would maybe be to consider some of the improvements above or to use a different method than GD. The experiments in this thesis has a lot of potential and some improvements can make them more accurate.

References

[1] *Image data: Courtesy of Assoc.Prof. Erik A. Hanson. Made available true personal communication.*

[2] Ahmad Barzkar and Hojatollah Adibi. On the propagation of longitudinal stress waves in solids and fluids by unifying the navier-lame and navier-stokes equations. *Mathematical Problems in Engineering*, 2015:1–9, 2015.

[3] R.A. Bernatz. *Fourier Series and Numerical Methods for Partial Differential Equations.* Hoboken, NJ, USA: John Wiley and Sons, Inc., 2010.

[4] E. K. P. Ching. *An introduction to optimization.* Wiley Series in Discrete Mathematics and Optimization. Wiley, 2013.

[5] W. Han K. E. Atkinson and D. Stewart. *Numerical Solution of Ordinary Differential Equations.* Pure and Applied Mathematics, Numerical Solution of Ordinary Differential Equations. Hoboken, NJ, USA: John Wiley and Sons, Inc., 2011.

[6] David Kincaid and Ward Cheney. *Numerical Analysis: Mathematics of Scientific Computing.* Pure and applied undergraduate texts: v. 2. Library of Congress Cataloging-in-Publication Data, 2002.

[7] Randall J. LeVeque. *Finite Difference Method for Ordinary and Partial Differential equations.* SIAM, Society of Industrial and Applied Mathematics, 2007.

[8] Jacob Lubliner and Panayiotis Papadopoulos. *Introduction to Solid Mechanics.* An Integrated Approach. Springer, 2014.

[9] Paul F. A. Bartha Mustafa A. Akcoglu, Dzung Minh Ha and Paul F. A. Bartha. Analysis in vector spaces. *Wiley Series in Probability and Statistics*, 319(1), 2011.

[10] J. M. Nordbotten. Cell-centered finite volume discretizations for deformable porous media. *International Journal for Numerical Methods in Engineering*, 100(6):399–418, 2014.

[11] Brian Borchers Richard C. Aster and Clifford H. Thurber. *Parameter Estimation and Inverse Problems.* Elsevier Academic Press, 2005.

[12] Walter A. Strauss. *Partial Differential Equations.* Laurie Rosatone, 2008.

[13] Ronald E ; Elliott Ryan S Tadmor, Ellad B ; Miller. *Continuum Mechanics and Thermodynamics From Fundamental Concepts to Governing Equations.* Cambridge: Cambridge University Press, 2011.

[14] Sona Taheri and Musa Mammadov. Solving systems of nonlinear equations using a globally convergent optimization algorithm. *Global Journal of technology and optimization*, 3, 2012.

[15] Lloyd N. Trefethen and David Bau. *Numerial Linear Algebra*. Society for Industrial Applied Mathematics, U.S., 1997.

[16] Jhabriel Varela. Implementation of an mpfa/mpsa-fv solver for the unsaturated flow in deformable porous media. 2018.

Nomenclature

α	step size
β	boundary condition
Δ	Laplace operator
ϵ	train tensor
η	noise
Γ	edge of domain, $\partial\Omega$
ι	boundary condition
κ	heat conductivity
λ	Lames first parameter
μ	Lames second parameter
∇	gradient
ν	Poisson's ratio
Ω	domain
ψ	heat source/sink
ρ	heat density
σ	stress tensor
$\mathbf{n}$	unit normal vector
θ	trace of ϵ
Υ	scalar
$\wp$	stiffens tensor (4th order)
ξ	mass density
ζ	boundary conditions
B	Boundary condition operator
c	cost function
D	(D_+, D_-, D_1, D_2)derivative operators

d	step size in time
E	Young's modulus
F	a force
f	differential condition
F_V	total surface force
g	function for heat eq.
G_i	vector of g_i
$H(t)$	heat
I	unit tensor
L, L'	line between two points, a,b and a',b'
n, m	dimension of vectors
O	Differential operator
p_k	sequence of variables
S	shear modulus
U	approximation of u
V	volume
v	step size in the x-direction
w	step size in y-direction