Multi-Level Inversion Based On Mesh Decoupling

Benny Shachor

TU Delft

Multi-Level Inversion Based On Mesh Decoupling

by

Benny Shachor

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday December 9, 2019 at 9:00 AM.

Student number: 4748573
Thesis committee: Dr. Domenico Lahaye, TU Delft, supervisor
Dr. Hadi Hajibeygi, TU Delft, supervisor
Dr. Ir. Femke Vossepoel, TU Delft

This thesis is confidential and cannot be made public until December 9, 2019.

TU Delft

Abstract

Understanding the permeability of the subsurface is a crucial step to simulate fluid flow in the subsurface. A parameter estimation problem for the flow equations can be solved to find the permeability. The robust identification of material parameters remains a significant challenge. In classical approaches, the non-linear least-squares problem is formulated as a non-linear optimization problem in which the partial differential equation governing the permeability field acts as a constraint. These approaches lead to large scale problems and are, therefore, computationally challenging. This thesis proposes a new approach based on mesh decoupling of state and design variables. This approach allows treating the design variables on various scales of resolution without comprising the accuracy of the state and adjoint solver. The method is implemented on a one-dimensional and two-dimensional Poisson problem taken from the literature. The first numerical results show that the multilevel approach is able to accelerate the initial stages of the search procedure significantly.

Acknowledgements

I would like to use this opportunity to thank my supervisors Dr. Domenico Lahaye and Dr. Hadi Hajibeygi. Without their guidance, supervision, insights, and help, I would not be able to do this work. I thank them for believing in me and willing to accept the challenge of combining pure mathematical research with petroleum engineering aspects to explore new ideas that can benefit both aspects of science. During the work, both have let me the flexibility to test my thoughts, some time to fail, all to be a better student and researcher for later stages. This approach has taught me a lot, and I thank them for that.

I thank Dr. Ir. Femke Vossepoel for her interest in my work and for being a committee member for my thesis defense. During my studies, the courses and interactions with professor Vossepoel have encouraged me to find interest in geostatistics and different interpolation methods.

I want to thank the TU Delft university staff for the different courses, classes, presentations, and fieldwork. All that prepared me to be a better scientific researcher, petroleum engineer, and a co-worker.

Last but not least, I would like to thank my dear family and especially my dear mother for the constant support in the challenging times and for encouraging me to go abroad and pursue an MSc.

Contents

List of Figures

List of Tables

1 Introduction

The flow of fluid in porous media arises in different fields of science, such as flow in the subsurface, flow of blood in veins. To be able to model the physical phenomena of fluid flow in the subsurface, one needs to be able to understand several characteristics of the subsurface and the fluid. One of the characteristics of the subsurface is permeability. Measuring the permeability is complicated because the subsurface is not reachable. In the best case, the subsurface is reached on limited locations one would like to investigate.

Modeling of oil flow in the subsurface is a crucial aspect of the oil and gas industry. Accurate modeling will allow the engineer to decide on the right spots to place the well, what kind of surface facilities is needed, what are the economic circumstances and the projection of the expected production. Therefore one needs to understand the reservoir characteristics and among them the permeability.

Mainly, the estimation of the subsurface permeability is being done by taking core samples while drilling the wells. In the laboratory, the rock properties will be estimated. Using the rock properties and different correlations, the permeability can be approximated. The calculated permeability is valid for the location of the wells. Then the permeability field can be interpolated/extrapolated to different areas of the field. The main methods used to do the interpolations are geostatistical methods such as kriging, object-based modeling, etc. The disadvantage of using these procedures is the large uncertainty that is involved in each of the processes. Starting from the seismic acquisition and finishing in the different geostatistical methods. Although disadvantages exist, the petroleum industry works with these methods and manage to reduce the uncertainty to produce oil and gas economically.

Inverse problems arise when one tries to estimate a parameter that is unreachable using measurements and observations of a process that was caused by the existence of these parameters. The formal studies of the mathematical theory of inverse problems started in the 20th century. Since then, they are widely investigated due to the significant importance of these methods to science. Among the application of inverse problems are Medical Imaging, Acoustics, Remote sensing, Astronomy, Geosciences, etc. Using inverse problem methods, one can estimate the permeability of the subsurface.

The current research will try to extend the known methods of inverse problems with the use of the Newton Trust Region method to do the optimization and will include the Multi-Level approach to reach more accurate and faster results. The use of MultiLevel methods will help solve the implications of the large scale problems that arise in the inverse problem. Those problems are formulated to a non-linear least squares problem, which is governed by non-linear behavior due to the elliptic equations that govern the permeability field.

The research will use the Julia programming language to emphasize the capabilities and benefits of using it in the geosciences computation applications.

Inverse problem for elliptic equations is described at [19], [9], where a basic 1D examples are being illustrated including guidelines to solve this kind of inverse problems. The FEM to help one discretize PDE's is described by the work of [18]. Regularization is a key to solve the illposedness of Inverse problems and is described at the work of [2], [10], [19], [9]. The inverse problem parameter estimation can be stated as a least squares problem which is described at [12], [3]. Numerical Optimization and the Newton with Trust Region method are described at [4],[15]. Multi-Level appraoches help reduce computation and yet stay robust and similar ideas are introduced in [5],[14],[17]. The use of Julia and applications is described at [1], [6], [13]

In chapter 1, an introduction to the topic is mentioned. In Chapter 2, the parameter identification problem will be introduced, including the methods used these days by the oil and gas industry to estimate the

subsurface permeability and what will be the benchmark problem that is tested in this thesis. Chapter 3 will explain the concepts used in the scope of this thesis from numerical mathematics. Chapter 4 describes how the parameter estimation was done, including stating the problem. Chapter 5, presents state of the art Multi-Level method and algorithm, chapter 6 shows the results and analysis reached during the work of this thesis. Chapter 7 will conclude the work of this thesis, primary findings, and future work on the topic.

2

Parameter Estimation

Parameter estimation problems arise when one conducts an experiment. In the experiment, the source and outputs can be captured. The problem arises when one would like to know what type of parameters causes the process. Parameter estimation problems arise in many physical problems such as Tomography, Radar, Acoustics, Petroleum engineering, etc. For example, in Tomography applications, one tries to estimate the shape of the human body internals without doing any penetration. In Geosciences, one would like to know what are the characteristics and structure of the subsurface where the limitation is that the subsurface can be reached only for limited locations.

Different methods are used to estimate the various parameters of the subsurface. The main methods are by Remote Sensing and Seismic acquisition.

Parameter estimation is important to any forward modeling being done. When one wants to calculate the flow of oil in the subsurface, a permeability field is needed. The simulation results are sensitive to the permeability map. In fig. 2.1 the difference in the permeability field has a different affect on the oil saturation and pressure fields of the simulation.

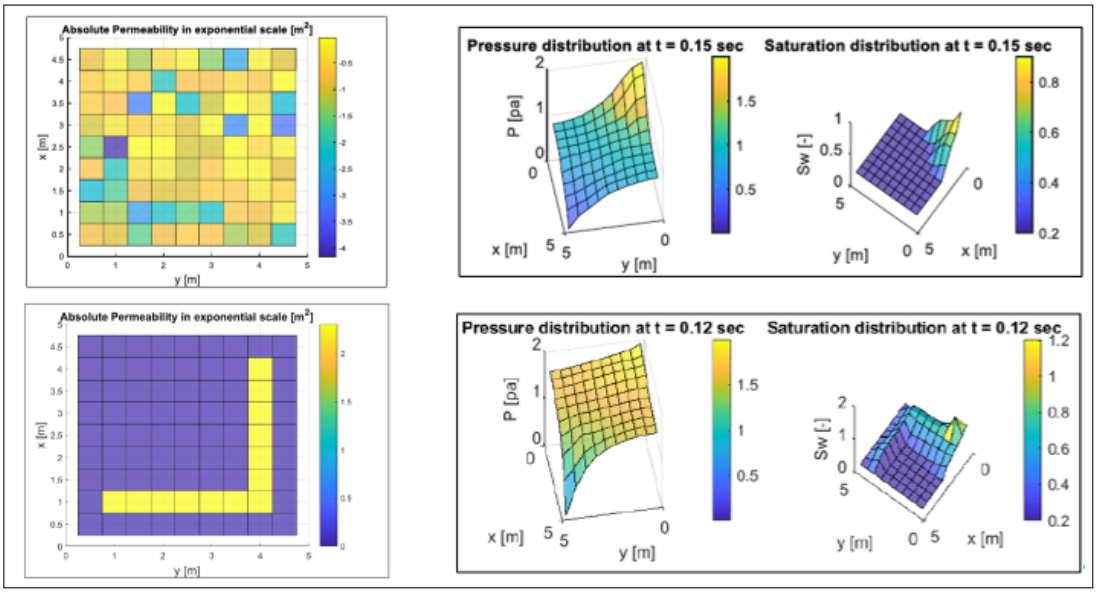


Figure 2.1: To the left there are two different permeabilty fields and to the right is each permeability field affect on the oil flow in the reservoir

The parameter estimation methods allow one to link physical phenomena to the data using equations that describe the process. An example of parameter estimation in geophysics is when one would like to learn the structure of the subsurface. Drilling a well in the subsurface is very expensive and therefore is done only on limited locations. A solution to this problem can not be reached without using parameter estimation

methods that allow the user not to penetrate the area of investigation. The existing methods to estimate the permeability field are as follows: Using seismic acquisition to build the structure of the subsurface. From the cores taken from the wells, a relationship between the porosity and permeability is derived. In parallel using the well logs, a geological model is built. Using geostatistical tools, the porosity is interpolated and extrapolated to the whole extent of the field. Then to conclude, the porosity - permeability correlation is used to create a permeability map of the field.

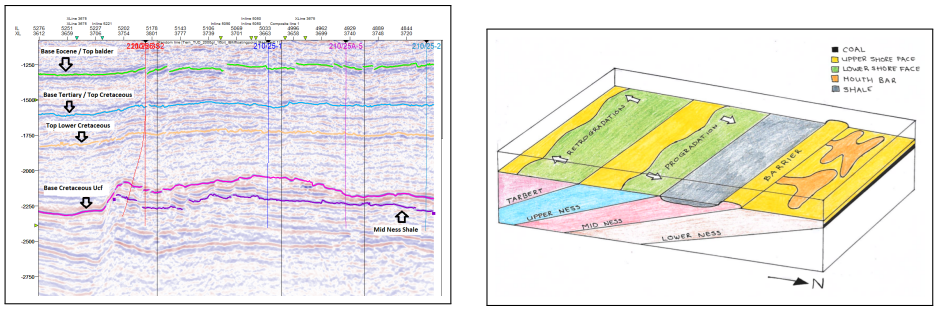
For example some methods used in geostatistics as in [11], [16] are:

Object-Based modeling (OBM) is a stochastic method that allows the creation of realistic geological objects inside the reservoir model. It uses the outcrop analogies to build the necessary objects. The main rule that leads to object-based modeling is erosion rules. Older strata are in greater depth than younger strata. One will also use different geological concepts such as the same object can't intersect the same object. The user will determine the shapes of the objects. Each object is modeled by a function that allows its creation. The model can be worked in different scales so different objects can be introduced in the same model. The OBM will take into consideration seismic, well, production, and interpretation information. For example salt bodies that were seen in the seismic will be an input for the model. In that case, the fluvial object will not be modeled in the same location as the salt body.

Sequential Indicator Simulation (SIS) using Ordinary Kriging, one can get the best linear unbiased estimate where one gets a minimum variance. Nevertheless, the Ordinary Kriging estimation has no randomness element, and therefore for each simulation, the result of the estimation will be the same. On the other hand, the subsurface includes a lot of uncertainty. This uncertainty can be modeled using different sequential simulations, which involves random processes for building the estimated grid. A setback for this approach is that this kind of procedure will have to run for a long time since it involves a lot of mathematical operations.

Kriging methods allows one to get the best unbiased with minimum variance interpolator. The kriging method creates weight for each point of data and then interpolates for different regions. Different kriging methods exist, and each has its benefits. The kriging can be used in the gaussian simulation to create a more random behavior to the interpolation.

An illustration of the different sources used to build the permeability field are in Fig. 2.2, an example for the permeability results coming from that sources is presented in fig. 2.3.



(a) Seismic data the shows the structure of the subsurface. The seismic also includes the drilled wells

(b) A geological model example

Figure 2.2: Different sources are combined in order to get a permeabilty field

As part of this thesis, to reduce the uncertainty that exists in the used methods, a different approach is used. The flow equations will represent the process of fluid flow in the subsurface. The flow occurs due to changes in pressures between the reservoir and the subsurface. The study will try to understand the structure of the reservoir and the permeability field in it using an Inverse problem technique. When one tries to solve the non-linear inverse problem, a lot of difficulties arise. One significant obstacle is the large scale of the problem. The Mult-Level approach introduced in this work tries to solve part of this difficulty.

The main limitations of the this thesis approach in solving parameter estimation problem is:

- The flow equation used for the research is a steady-state, incompressible equation which doesn't represent many cases of oil and gas flow problems
- The methods use regularization that depends on the user. Different methods will have different solutions for the solution

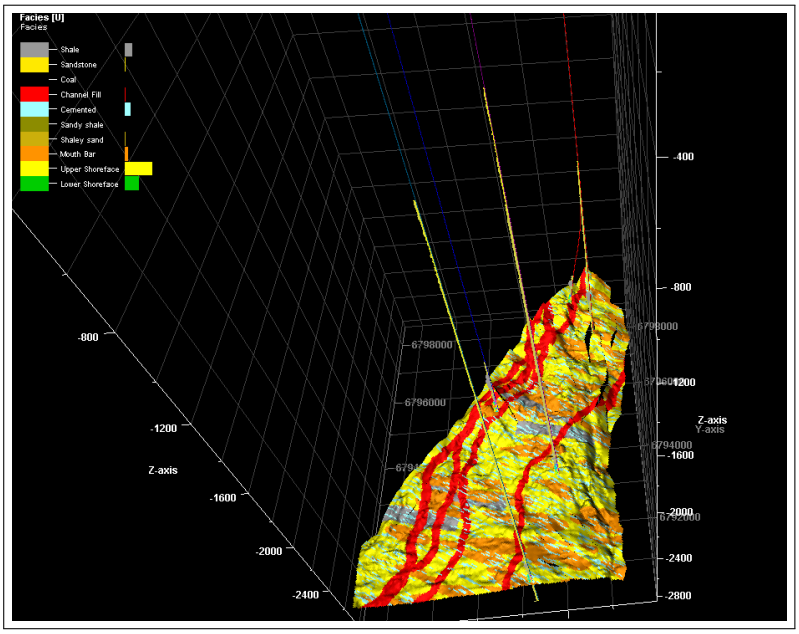
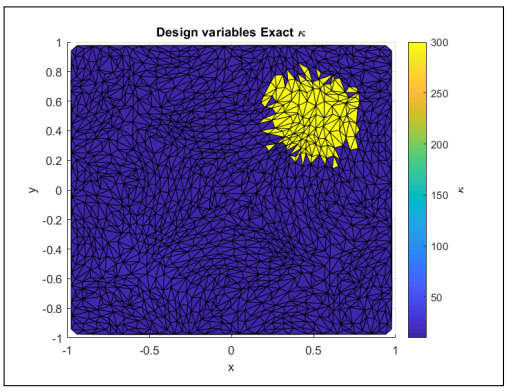


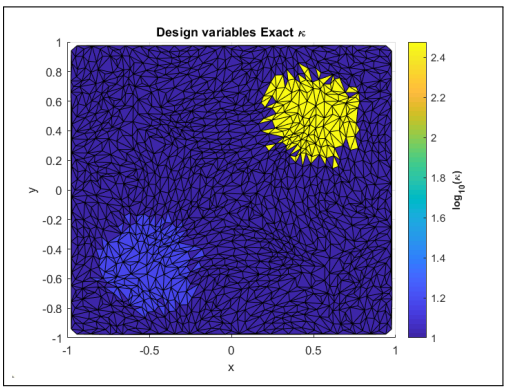
Figure 2.3: A permeabilty field which contains different objects and facies

- In what cases the Multi-Level approach fails is not clear
- Non-linear SVD analysis doesn't exist yet. The analysis can give a lot of information about the type of the regularization needed

The final problem to be solved as part of this thesis is described below. The problem comes as an example as in [8], where two bodies exist in the subsurface. One with very high permeability (can be, a lobe of sand). The second with a minor increase in permeability.



(a) Permeability field



(b) Permeability field in a log scale

Figure 2.4: A field with constant permeability which includes two objects. One has a very high permeability and the other has a slightly higher permeability compared to the permeability of the filed

3

Concepts from Numerical Mathematics

3.1. Introduction

In this chapter, the Theoretical concepts of the Finite Element Method (FEM) for solving the elliptic equation are introduced. The weak formulation and spatial discretization used in this thesis will be presented, as well. The basis function used for the FEM will be piece-wise linear. Later, the Quadratic Newton and Newton Trust Region methods for solving optimization problems are explained. This thesis consists of a comparison of the two methods.

The notations used in the chapters are:

∇ is the gradient operator, $\cdot$ is the dot operator, $<,>$ is the dot product, Ω is the domain and Γ is the domain boundary, $||x||^2$ is the Euclidean norm and is described as $||x||^2 = \sum_{i=1}^{n} (x_i)^2$, $Hess$ is the Hessian operator

3.2. Finite Element Elements for the Poisson Equation: weak form and spatial discretization

3.2.1. Finite Element Method

Weak Form and Spatial discretization:
The Elliptic problem is described as:

$$-\nabla \cdot (k \nabla u) = f$$

where k,u,x are function of space $\underline{x}$.

Multiplying by a function v and integrating over the domain Ω to get:

$$\int_{\Omega} (-\nabla \cdot (k \nabla u)\, v \, \mathrm{d}\Omega = \int_{\Omega} f\, v \, \mathrm{d}\Omega$$

v is a function of space $\underline{x}$ as well.

Discretization of the domain to node locations x_i for $i = 1....n$.
In 1 dimension the discretization is of points on a line as in the next figure:

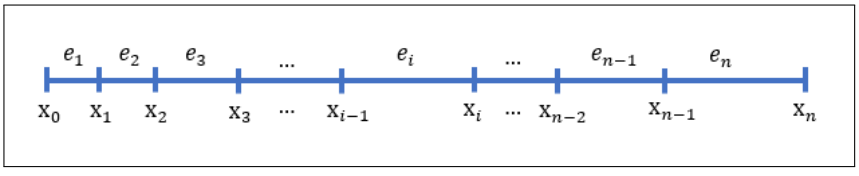


Figure 3.1: The nodes are denoted as x_i, $0 \le i \le n$. The elements are denoted as e_i, $1 \le i \le n$. The boundary nodes are x_0, x_n.

In 2 dimensions the discretization looks like:

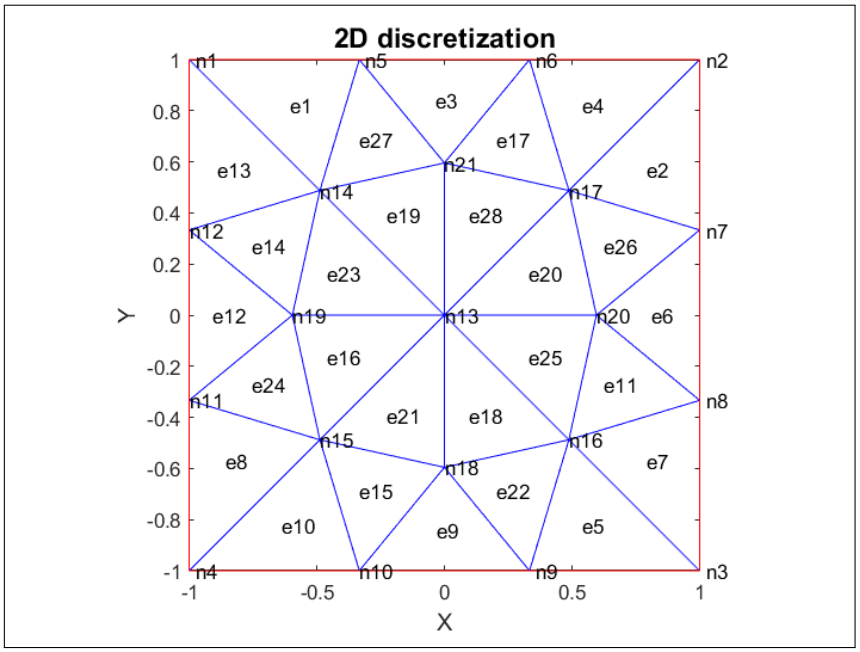


Figure 3.2: The 2 dimensions discretization where n_i represent the number of the node and e_i represents the element number. In this example there are 21 nodes and 28 elements

The **Divergence theorem**:

$$\int_{\Omega} (\nabla \cdot F) = \int_{\Gamma} (F \cdot n)$$

Where n is the normal to the domain boundary Γ. Defining $F = (k\nabla u)\,v$ and using the divergence theorem to get the **weak formulation**:

$$\int_{\Omega} k\nabla u \cdot \nabla v \, d\Omega - \int_{\Gamma} k\frac{\partial u}{\partial n} v \, d\Gamma = \int_{\Omega} f v \, d\Omega$$

By approximating the solution u to be a linear combination of basis functions ϕ_j:

$$u^n(\underline{x}) = \sum_{j=1}^{n} a_j \phi_j(\underline{x})$$

and substituting u in the weak form to get the next equation:

$$\sum_{j=1}^{n} \Big(\int_{\Omega} k\nabla\phi_j \cdot \nabla v \, d\Omega - \int_{\Gamma} k\frac{\partial \phi_j}{\partial n} v \, d\Gamma \Big) a_j = \int_{\Omega} f v \, d\Omega$$

By choosing n functions as $v = \phi_i$ to get:

$$\sum_{j=1}^{n} \Big(\int_{\Omega} k\nabla\phi_j \cdot \nabla\phi_i \, d\Omega - \int_{\Gamma} k\frac{\partial \phi_j}{\partial n}\phi_i \, d\Gamma \Big) a_j = \int_{\Omega} f\phi_i \, d\Omega$$

In our problem the boundary conditions are Dirichlet boundary conditions (and choosing: $\phi_i|_{\Gamma} = 0$). Therefore:

$$\int_{\Gamma} k\frac{\partial \phi_j}{\partial n}\phi_i \, d\Gamma = 0$$

Discretizing the equation to get the next linear equation:

$$A(k)\,u = f$$

Where u is the desired state solution, A is the stiffness matrix, and f is the flux density of the source vector. To add the **Dirichlet boundary condition** to the discretiazation A(k) at the rows and colums of the boundary nodes will be equal to zero, except for the nodes A(i,i) = 1, where i is in Boundary Nodes. And f(i) = 0. Where i is in boundary nodes.
The stiffness matrix is constructed in the following manner:

$$[A]_{i,j} = \int_{\Omega} k\nabla\phi_i \cdot \nabla\phi_j \, \mathrm{d}\Omega$$

Since the domain $\Omega = \sum_{j=1}^{n} e_j$ where e_j is the j element. And the integral can be estimated on each element with the Quadrature method as:

$$\int_{e_i} k\nabla\phi_j \cdot \nabla\phi_i dv = k_i(vol_{e_i}) AV(\nabla\phi_j \cdot \nabla\phi_i)|_{All\,nodes\,of\,element\,i}$$

where k is homogeneous over the element, vol_{e_i} is the volume of the element e_i and $AV()$ represents the mean function.

To conclude each entree in the stiffness matrix is discretized as the following:

$$[A]_{i,j} = \sum_{i=1}^{n_{elm}} k_i(vol_{e_i}) AV(\nabla\phi_j \cdot \nabla\phi_i)|_{All\,nodes\,of\,element\,i}$$

and n_{elm} represents the number of elements.

The vector f:

$$[f]_i = \int_{\Omega} f\phi_i$$

and is discretized in the same manner:

$$[f]_i = \sum_{m=1}^{N_{elm}} f_{e_i}(vol_{e_m})(AV(\phi_i)|_{All\,nodes\,of\,element\,m})$$

where f_{e_i} is homogeneous over the element. [18]

The basis function used in this projects are linear functions such as:

$$\sum_{i=1}^{dim} a_i x_i + b = 0$$

and dim denotes the dimension of the problem, i.e. in 2 dimensions in space $dim = 2$. The basis functions will be defined as:

$$\phi_i(\underline{x_j}) = \delta_{i,j}$$

Therefore:

$$\nabla\phi_j \cdot \nabla\phi_i =< a_i, a_j >$$

Where a_i, a_j are the coefficients of the functions ϕ_i, ϕ_j respectively.

3.2.2. One Dimensional Problem

1D Basis Functions

In one dimension the basis functions looks like: $a_1 x_1 + a_2 = 0$.

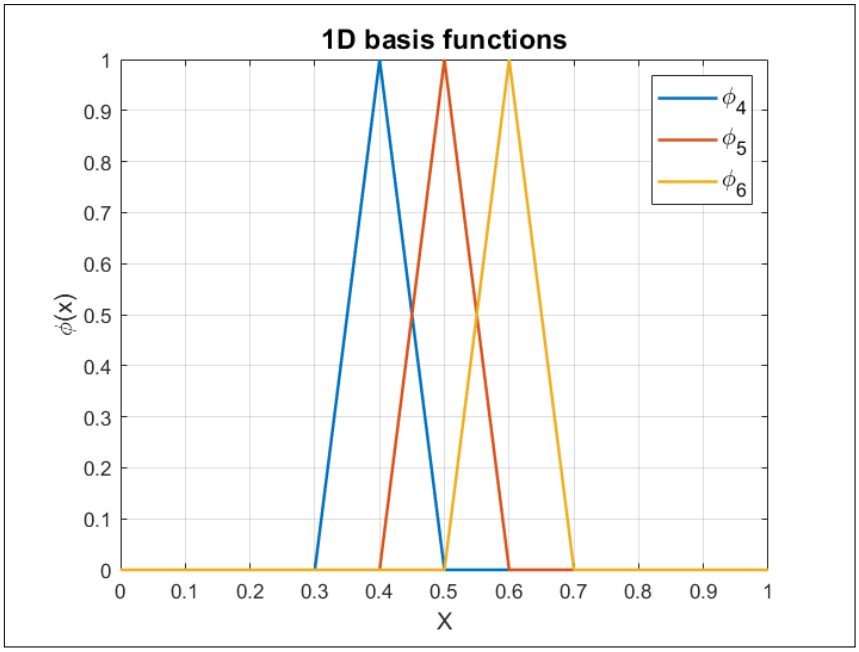


Figure 3.3: 1D basis functions example. The basis function ϕ_4, ϕ_5, ϕ_6 has a value of 1 at the points x_4, x_5, x_6 respectively and zeros on the other nodes. Between the nodes the functions are linear

3.2.3. Two Dimensional Problem

2D basis functions

In two dimension the basis functions looks like: $a_1 x_1 + a_2 x_2 + a_3 = 0$.

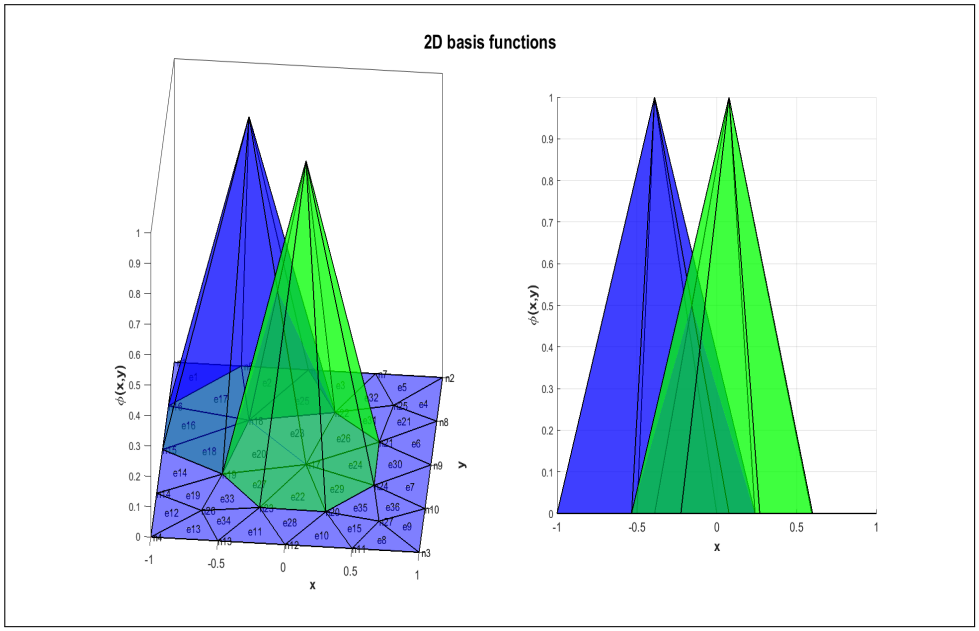


Figure 3.4: 2D basis function examples. In green and blue are the basis function ϕ_{17}, ϕ_{18} respectively. The basis function ϕ_{17}, ϕ_{18} has a value of 1 at the nodes x_{17}, x_{18} respectively and zeros on the other nodes. Between the nodes the functions are linear

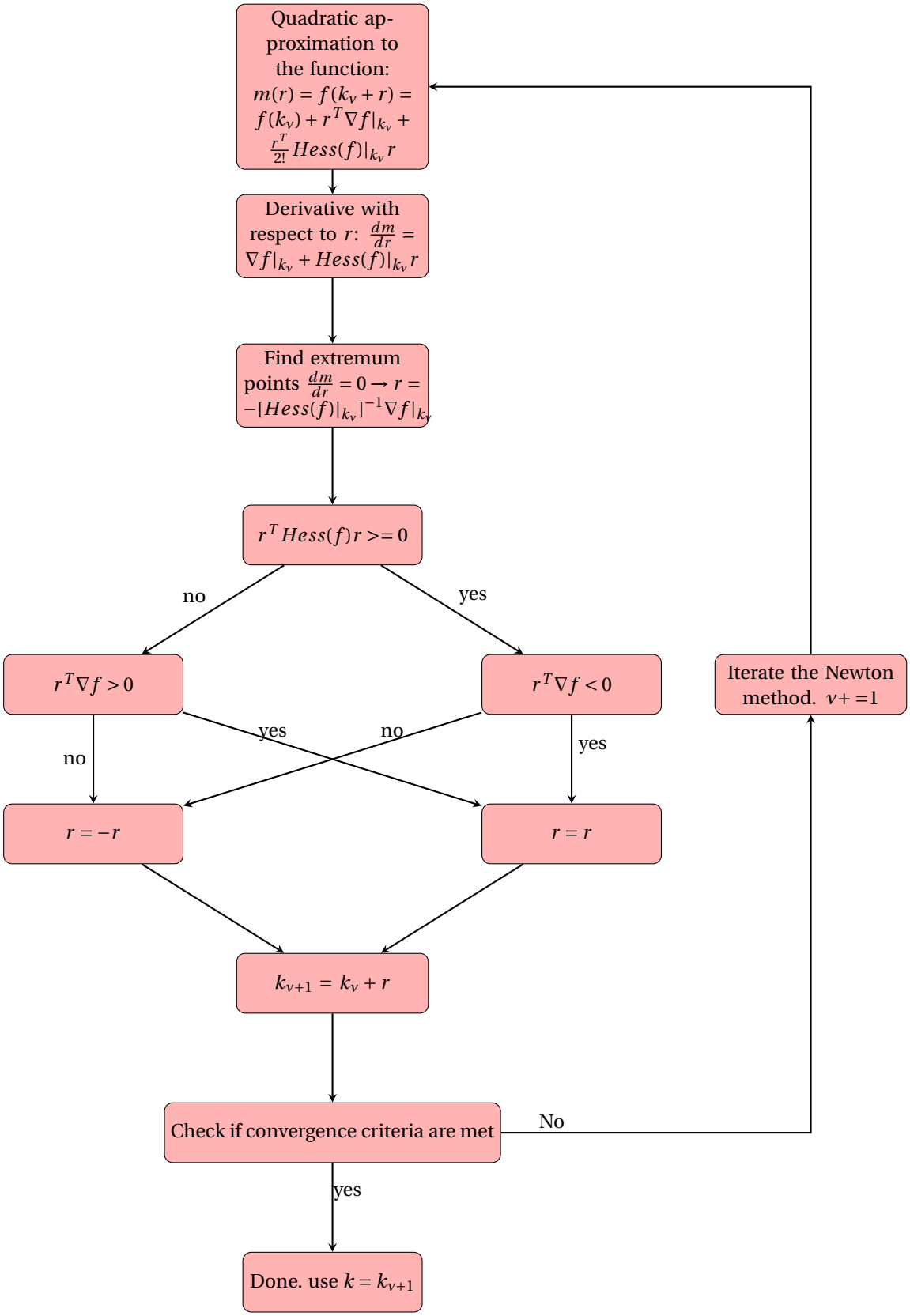
3.3. Non-Linear Optimization Methods for Non-Linear Inversion

Optimization algorithms are meant to help one to solve a problem where the solution is difficult or unknown but can be approximated by several techniques. These techniques are called mathematical optimization. Each problem usually has a few algorithms that can help solve the problem. In the case of this thesis, the least square problem nature is non-linear, and therefore the use of optimization algorithms for non-linear problems is needed. The two methods to be used are Newton and Newton Trust Region.

3.3.1. Newton Method

The Newton Method in optimization is an iteration solver for a non-linear optimization problem. The idea of the newton method in calculus is to find the roots of an equation. In optimization, we apply the calculus Newton method to find the roots of the derivative of the function of the stationary points of the function, which are the maximum (max) and minimum (min) of the function. Newton method is a local method that will find a local min/max but not the global min and max. The global min or max are the smallest and biggest values of the function over the space the function is defined on. The advantage of the method is that if the method converges, it can converge very fast. The setbacks of the Newton method is that it will not promise a convergence if the initial guess is far from the min/max point. The globalized Newton method can help solve this issue and is discussed at 3.3.2.
The Newton Method is described by the next algorithm: Starting at initial guess k_0 and looking at the quadratic approximation for the cost function $f(k)$: $m(r) = f(k+r) = f(k) + r^T \nabla f|_k + \frac{r^T}{2!} Hess(f)|_k r$ one would like to minimize m(r). Therefore to find a minimum one takes a derivative with respect to r and gets the next step of the Newton iteration loop: $0 = \nabla f|_k + Hess(f)|_k r \rightarrow r = -[Hess(f)|_k]^{-1} \nabla f|_k$. Describing $r = k_{v+1} - k_v$ one finally gets: $k_{v+1} = k_v + r$. By choosing a stopping criteria one can stop the Newton loop.

The Newton Algorithem is:



3.3.2. Newton with Trust Region Method

The Trust Region algorithm aims to guarantee convergence of the optimization problem. Therefore it is sometimes called a global convergence method. It is essential to notice that the name global convergence can be confusing since the technique will not find definitely **the** global max/min of the problem, but it will find **a** max/min for sure. This is why the Newon with Trust Region method got the name of a **global solver**.

The Trust Region method is an iterative solver as the Newton solver. The difference is in the way one chooses the next step of the iteration and also a constraint to the minimization problem.

In the Trust Region method, the function is being approximated (in this thesis by a quadratic approximation). Then an optimization for the sub-problem will be solved inside the region. The solution to the optimization will be used to check how good was the approximation of the function compared to the objective function. If the estimate is good, the region will be increased, and a new optimization problem will be solved in the new region. The point where the function was approximated will be the new point found from the last step.

The Trust Region problem is:

$$\begin{aligned} \underset{r}{\text{minimize}} \quad & m(r) = f(k) + r^T \nabla f|_k + \frac{r^T}{2!} Hess(f)|_k r \\ \text{subject to} \quad & \|r\| \leq \Delta \end{aligned} \tag{3.1}$$

A way to solve the minimization problem (3.1) can be found in [15]. This approach to solving the minimization problem is also implemented in the Julia package Optim.jl as explained in 6.3.

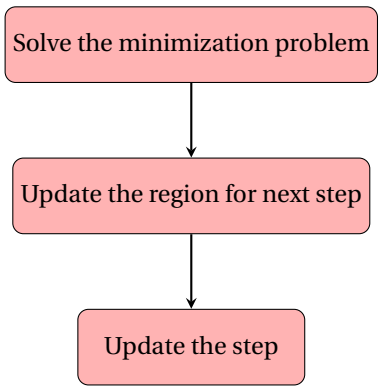
After finding r we can evaluate the next expression:

$$\rho_v = \frac{f(k_v) - f(k_v + r_v)}{m_v(0) - m_v(r_v)}$$

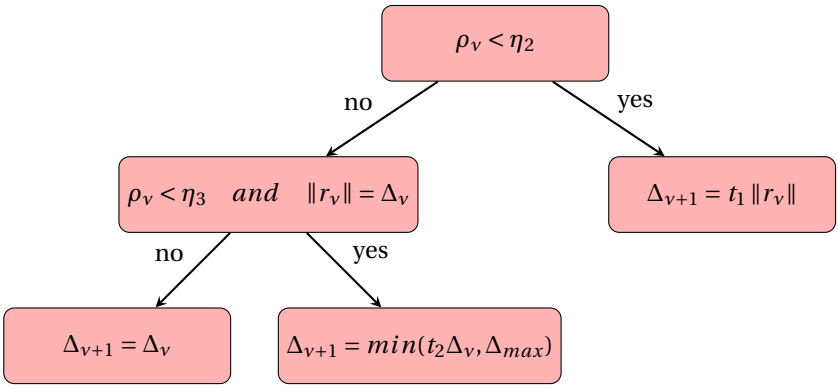
This expression gives the ratio that shows how good is the quadratic approximation to the objective function. The nominator will be called **the actual reduction**, and the denominator is **the predicted reduction**. If the ratio $\rho_v \approx 1$, the model is a good representation for the objective function, and one should increase the Trust Region. If the ratio $\rho_v \approx 0$, it means the model is not a good representation. Notice, the denominator is positive since one minimizes the problem on a space, which includes $r = 0$. If ρ_k is negative it means that the next step $f(k_v + r > f(x_v)$ didn't manage to minimize the objective function. Therefore one will not accept the step.

The Trust Region algorithm:

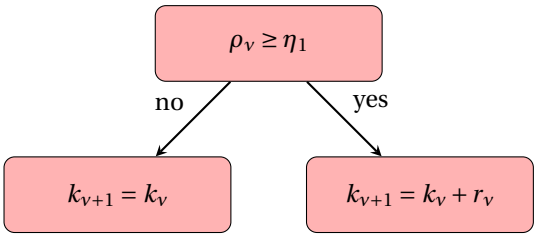
1. Solving a minimzation problem:



2. Updating the Trust Region:



3. Updating the step:



Typical values for the parameters from [15] are: $0 \leq \eta_1 \leq \eta_2, \eta_2 = 0.25, \eta_3 = 0.75, t_1 = 0.25, t_2 = 2$

3.3.3. Criterias for the different methods

stopping criteria

Max Iter	25
x_{tol}	$1e-6$
∇_{tol}	$1e-9$

Table 3.1: Optimization stopping criteria

Trust Region parameters

$\Delta_{initial}$	1
Δ_{max}	100
η_1	0.1
η_2	0.25
η_3	0.75

Table 3.2: Trust Region parameters

4

Parameter Estimation for the Poisson Equation

4.1. Introduction

In this chapter, the Poisson Parameter estimation problem will be described. The Poisson problem is formulated to a Non-Linear Least Square problem. Next, the Ill-posedness of the inverse problem is discussed. To solve the Ill-Posedness, a regularization term will is used. A way to choose the regularization function and its weight (the regularization parameter) will be explained. The adjoint method and Numerical differentiation are the methods used to calculate the Gradient and Hessians. The cost of computing the Gradient and Hessian will be presented as well.

4.2. Problem formulation

Looking at the elliptic equation:

$$\nabla \cdot (k \nabla u) = f$$

discretizing using FEM as in 3.2.1 to get the next linear problem:

$$A(k)u = f$$

k is the design variables, u is the state variables and A is the diffusion operator:

$$A = -\nabla \cdot (k(\underline{x}) \nabla(\cdot))$$

If one assumes that the observed data d can be described as:

$$d = Cu + \eta$$

C is the state to observation map such as $(Cu)_i = u(x_i)$, $\forall i = 1, ..., n$ and n is the amount of points that u was observed in the experiment.

One would like to find k that minimizes the next least squares minimization problem:

$$\underset{u \in U, k \in K}{min} \frac{1}{2} \parallel Cu - d \parallel_2^2$$

adding a regulariztion term to get the minimization problem that will be solved as part of the scope of this thesis:

$$\underset{u \in U, k \in K}{min} \frac{1}{2} \parallel Cu - d \parallel_2^2 + \alpha J(k)$$

J is the regularization functional, and α is the positive regularization parameter.

Since $u = A^{-1} f$ and assuming the problem in 4.2 is well posed one can denote $F(k) = CA(k)^{-1} f$ to get the next minimization problem:

$$\underset{k \in K}{min} \frac{1}{2} \parallel F(k) - d \parallel_2^2 + \alpha J(k)$$

Measurements can be taken from different experiments for different sources f. Then the minimization problem is:

$$\min_{u\in U,k\in K} \sum_{i=1}^{n} \left(\frac{1}{2} \parallel F(k) - d \parallel_2^2 + \alpha J(k) \right)$$

Where n is the number of experiments. By denoting $\alpha = \frac{\alpha}{n}$ one can write the problem as:

$$\min_{u\in U,k\in K} \left(\sum_{i=1}^{n} \frac{1}{2} \parallel F(k) - d \parallel_2^2 \right) + \alpha J(k)$$

For the scope of this work, the minimization problem will be solved using the Newton and Newton with Trust Region methods. Both methods involve the use of the Gradient and Hessian of the function. One can sum the Gradient and Hessian of each experiment due to derivative rules. Moreover, one can add the Gradient and Hessian of the regularization term.

4.2.1. One dimensional problem

The Elliptic problem in 1 dimension is described as:

$$-\frac{\partial}{\partial x}(k\frac{\partial u}{\partial x}) = f$$

where k,u and f are function of space x.

As a test case for the inverse problem, a comparison to the problem at [19] is conducted. The details of the problem are listed below:

Number of x points	65
x positions	equidistant between [0,1]
Number of observations	5
Location of Observations	[0.1 , 0.3, 0.5 , 0.7, 0.9]
Boundary Conditions	Dirichlet on edges ($u_\Gamma = 0$)
Number of experiments	2

Table 4.1: 1D problem parameters

Forward model and source functions

The source function used is a Dirac delta in different locations, the locations of the nodes before $x = \frac{1}{3}, \frac{2}{3}$. Each source will be used one time as an experiment. Later the two experiments will be used together to solve the inverse problem.

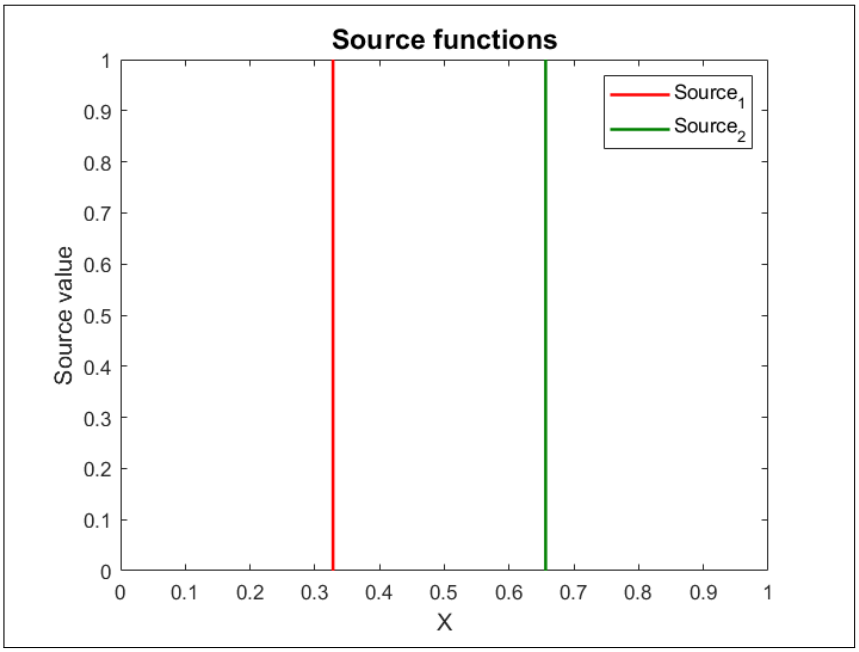


Figure 4.1: The sources of the two experiments. The sources are located at $x = \frac{1}{3}, \frac{2}{3}$ for the two experiments respectively. Both sources are a Dirac delta which has a value of 1 at one point grid and everywhere else are equal to zero

The design variables k values for the model will have a Symmetric Gaussian shape:

$$Gauss = e^{\frac{-(x-c)^2}{2\sigma^2}}$$

where $c = 0.45, \sigma = \frac{1}{12}$ and the Design Variables are:

$$k = -0.2Gauss + 1$$

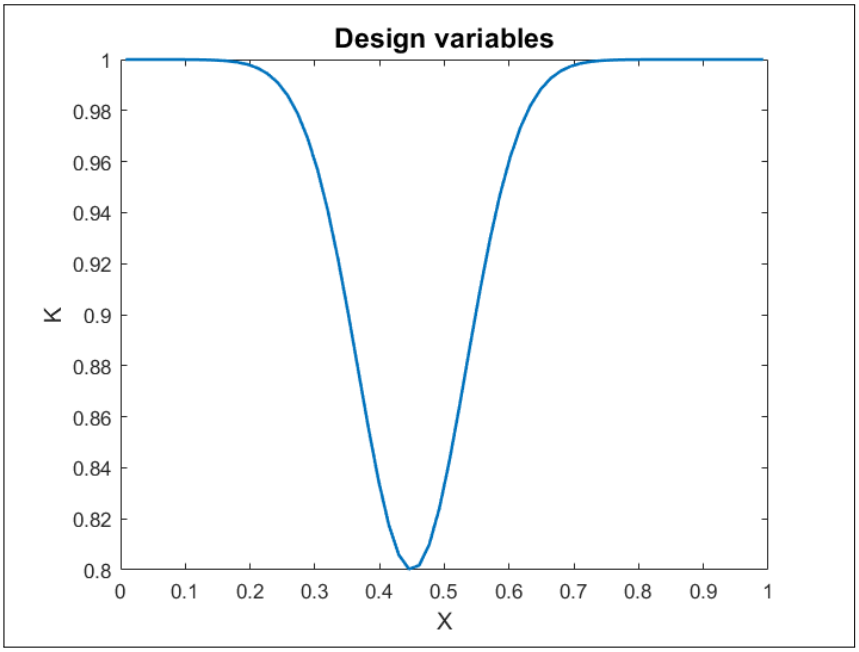


Figure 4.2: The exact design variable plot has a gaussian shape

Discretizing the elliptic equation using FEM as in 3.2.1 and then solving $u(k) = A(k)^{-1} f$ to find the state variable forward problem values. Notice, the solution u includes the noise of the FEM solution. This noise will later be used as the observation noise for the inverse problem. The forward model solution and the observations taken for the inverse problem are:

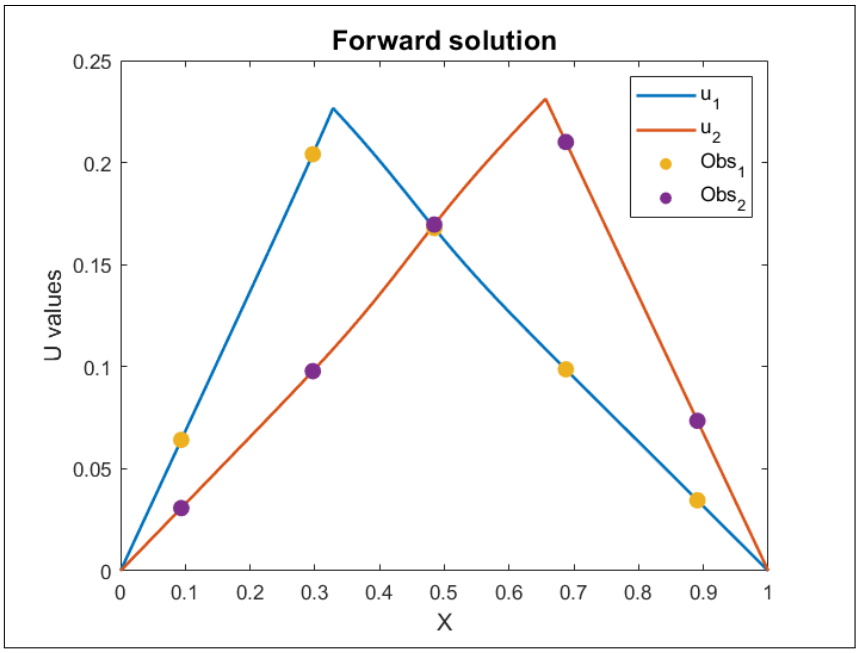


Figure 4.3: Forward model and sampled data for two experiments, The circles are the observations taken from each experiment. The observations includes the noise of the FEM solution

4.2.2. Two dimensional problem

The details of the problem are listed below:

Number of nodes	49
Number of elements	76
Number of observations	5
Boundary Conditions	Dirichlet ($u_\Gamma = 0$)
Number of experiments	5

Table 4.2: 2D problem parameters

The discretization and observation locations:

The sources will be located at the closest nodes to $[-\frac{1}{2},-\frac{1}{2}],[-\frac{1}{2},\frac{1}{2}],[0,0],[\frac{1}{2},-\frac{1}{2}],[\frac{1}{2},\frac{1}{2}]$. In each experiment, there will be used one source.

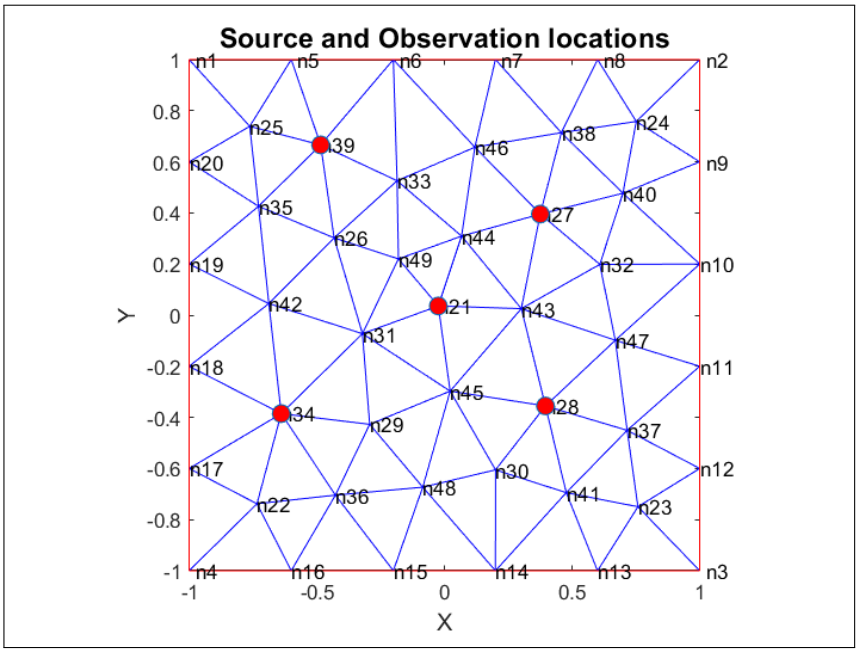


Figure 4.4: The mesh includes 49 nodes and 76 elements, the maximum distance between two nodes is Hmax = 0.45. the observations used in each experiment are the red circles

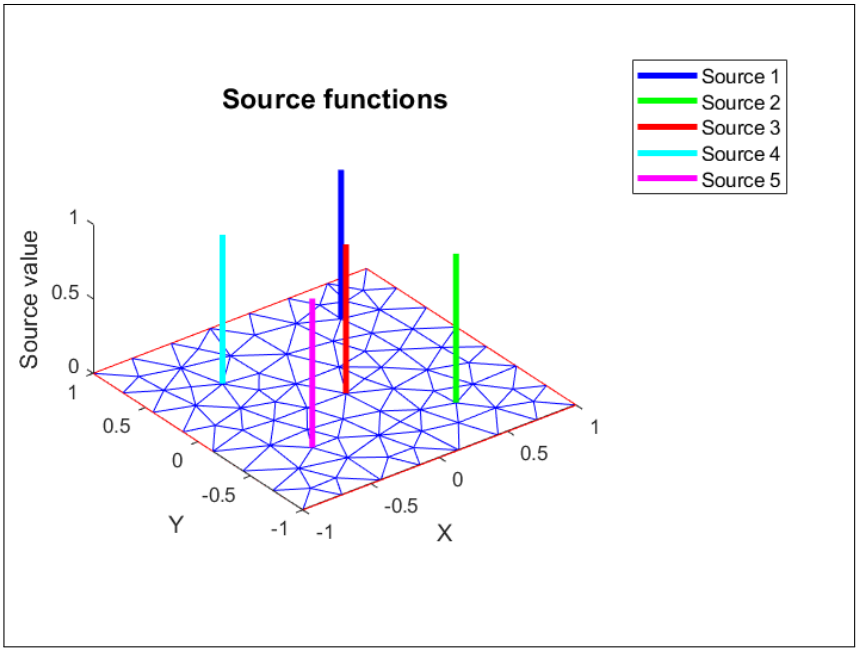


Figure 4.5: The sources in 2D, each source represent a Dirac delta which has a value of 1 at one point grid (the source location) and everywhere else are equal to zero. Each source correspond to one experiment

The exact solution has the shape of a ball inside a rectangle. Since the values of the design variables are not smooth over space, it will add complexity to the problem.

As in the 1D case above, after discretizing and solving using the FEM, one gets the forward solution, which includes the noise of the FEM. The exact solution is:

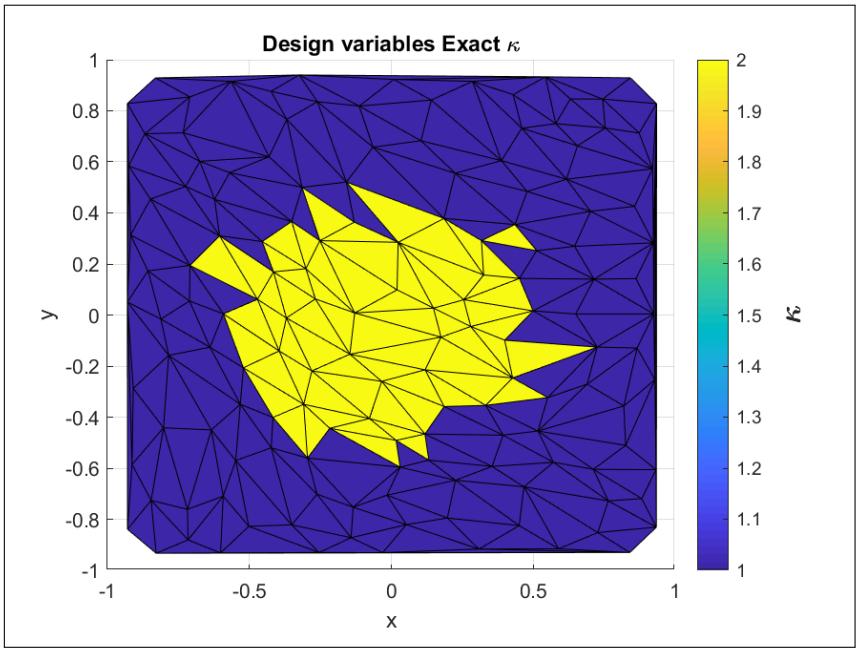


Figure 4.6: Exact solution for design variables where there is a circle inside a rectangle. This will be also used as an input for the forward modeling

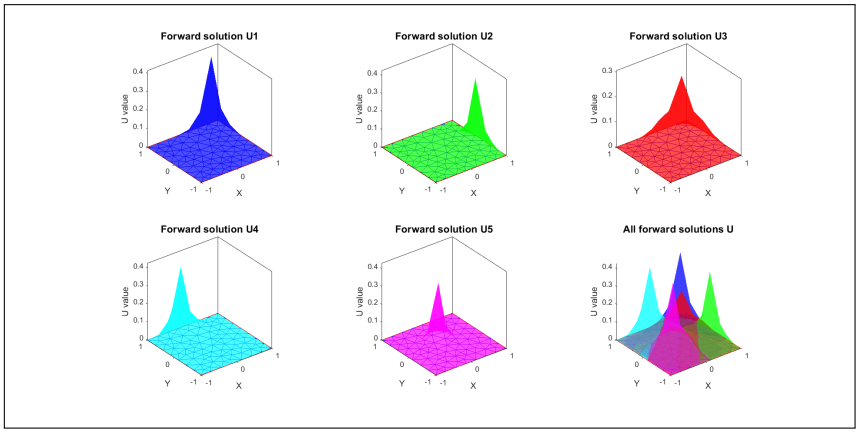


Figure 4.7: Forward model experiments solutions. As expected in each solution the peak is at the location of the source term

4.3. Illposedness and Regularization

4.3.1. Well and Ill-Posed problems

Jacques Hadamard introduced the term Well Posed problem. A well-posed problem is a problem that satisfies the next three rules:

- A solution exists
- The solution is unique
- The solution's behavior changes continuously with the initial conditions (Or in other words the solution is stable)

Jacques Hadamard was a mathematician among his work is the mathematical representation of physical phenomenons.

4.3.2. Ill-posed problem

A problem that violates any of the terms to be well-posed is an Ill-posed Problem.

Most inverse problems are Ill-posed. Among them are also the least-squares of the elliptic inverse equation. The reason the equation is Ill-posed is that it violates the rules of well-posedness.

- Existence - In some cases where we have a measurement error, a solution might not exist.
- Uniqueness - The solution is not unique as there could be more than one solution in locations where we do not have a measurement
- Stability - Small changes in the observation can cause a big change in the inverse solution due to measurements error

Solving the Ill-posedness intuition from linear Problems:

- Existence - instead of solving: $Ax = b$ to find x one solves $\underset{x}{min} \left\| (Ax) - (b) \right\|_2$ Now a solution will exist.
- Uniqueness & Stability - adding a regularization term, which causes the problem to become slightly different. The new problem will be unique and stable. The regularization term will force one solution to the problem also in locations where measurements do not exist. Moreover, the regularization term will smooth the solution and remove the noise that causes the problem to be unstable.

Motivation from Linear Problems to explain how the regularization solves the stability issue:
Looking at the problem:

$$Ax = b$$

where A is known, x us the state variable solution one would like to find, and b are the measurements.

Using Singular Value Decomposition for the matrix A will lead to the next analysis:

$$A = U\Sigma V^T = \sum_{i=1}^{n} u_i \sigma_i v_i^T$$

where U is an $m \times m$ unitary matrix , V is an $n \times n$ unitary matrix. Σ is a diagonal $m \times n$ matrix with non-negative real numbers (σ_i - the singular values) on the diagonal.

Looking at the inverse problem $x = A^{-1}b$ and combining the SVD analysis one gets:

$$x = A^{-1}b = \sum_{i=1}^{n} \frac{u_i^T b}{\sigma_i} v_i$$

b can be written as the real value of the measurement while adding the noise. Therefore:

$$b = \overline{b} + e$$

where e is the noise. In that case one can write the problem as:

$$x = \sum_{i=1}^{n} \frac{u_i^T \overline{b}}{\sigma_i} v_i + \frac{u_i^T e}{\sigma_i} v_i$$

Which means that if $u_i^T b$ does not decay faster to zero, then σ_i the error will be much more dominant for smaller σ_i and convergence will not be reached to the desired solution. [9]

Smoothing norms:

A smoothing norm can act as the regularization term one would like to add to his problem to solve the stability and uniqueness conditions.

A smoothing norm can be chosen based on the problem that one is trying to solve. The weighted Sobolov norm is:

$$S(k) = \sum_{i=0}^{n} \alpha_i ||(k^{(i)})||_2^2$$

Where the norm is the L_2 norm and $k^{(i)}$ is the i-th derivative of k, i.e. $k^{(2)}$ is the second derivative of k.

The smoothing norm will be minimum if the curve represents the order of the problem, which means, for example, that in the second-order smoothing norm, the minimum of the function is presented by using a quadrature approximation to the problem at the points where there are no measurements.

The regularization term chosen for this thesis is the first order norm $S(k) = ||(k^{(1)})||_2^2$ which is a first order smoothing norm.

4.3.3. Tikhonov Regularization

Tikhonov has offered a way to solve the Ill-posedness of a problem by adding a regularization term. The regularization term is another constraint to the problem. The regularization will have a weight as well, which will be called the regularization parameter.

If one looks at the **linear problem** introduced in 4.3.2, the new problem to be solved is:

$$\min_x \left\| \begin{pmatrix} A \\ \alpha L \end{pmatrix} x - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2^2 = \min_x \|Ax-b\|_2^2 + \alpha \|Lx\|_2^2$$

since:

$$\left\| \begin{pmatrix} y \\ z \end{pmatrix} \right\|_2^2 = \begin{pmatrix} y \\ z \end{pmatrix}^T \begin{pmatrix} y \\ z \end{pmatrix} = y^T y + z^T z = \|y\|_2^2 + \|z\|_2^2$$

Doing so added a constraint for the minimization problem, which is: $\alpha Lx = 0$, L is the smoothing norm. Looking at the representation of the equations as described by Hansen [9]:

$$\begin{pmatrix} A \\ \alpha L \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix} \Rightarrow \begin{pmatrix} A \\ \alpha L \end{pmatrix}^T \begin{pmatrix} A \\ \alpha L \end{pmatrix} = \begin{pmatrix} A \\ \alpha L \end{pmatrix}^T \begin{pmatrix} b \\ 0 \end{pmatrix} \Rightarrow x = (A^T A + \alpha^2 L L^T)^{-1} A^T b$$

In the last formulation and by using SVD formulation [9] and assume for the example that $L = I$ which is zero order smoothing. :

$$x = V(\Sigma^2 + \alpha^2 I)^{-1} \Sigma U^T b$$

Therefore the inverse part of the equation has higher singular values due to the addition of $\alpha^2 I$, which will help to solve the issue of lower singular values that make the noise not controllable.

In the scope of this thesis to try and do a parameter identification one solves:

$$\min_x \left\| \begin{pmatrix} CA(k)^{-1} f \\ \alpha Lk \end{pmatrix} - \begin{pmatrix} d \\ 0 \end{pmatrix} \right\| = \min_x \left\| CA(k)^{-1} f \right\|_2^2 + \alpha \|Lk\|_2^2$$

By solving the problem above, one tries to find a balance between the first term that measures if one has a good fit to the model and the second term that measures how regularized is the solution.

4.3.4. Regularization functional

The regularization functional J will be chosen to be:

$$J = \frac{1}{2} || \frac{d}{dx} k||^2 = \frac{1}{2} k^T L k$$

where k is the design variables one wants to find, $\frac{d}{dx}$ is the forward difference with homogenous boundary conditions operator and L is the discrete Laplacian matrix with Neumann boundary conditions.

The matrix L is coming from the form:

$$|| \frac{d}{dx} k||^2 = < \frac{d}{dx} k, \frac{d}{dx} k > = < k, \frac{d}{dx}^T \frac{d}{dx} k > = < k, Lk >$$

Where $L = \frac{d}{dx}^T \frac{d}{dx}$. The matrix L is the Laplacian matrix which is the discrete form to the next continues problem:

$$-\nabla \cdot \nabla u = f$$

The problem is discretized to:

$$Lu = f$$

where L is the **Negetive** Laplacian matrix, u is the solution vector and f is the source.

The **Gradient** of the regularization functional is: Lk. The **Hessian** of the regularization functional is simply: L. α the regularization parameter will be chosen based on the L curve method at 4.3.5

An example from the 1D case is for a regularization functional that represents a smoothing norm of the first order:

$$\frac{d}{dx} = \begin{bmatrix} \frac{-1}{h_1} & \frac{1}{h_1} & & & \\ & \frac{-1}{h_2} & \frac{1}{h_2} & & \\ & & \ddots & \ddots & \\ & & & \frac{-1}{h_n} & \frac{1}{h_n} \end{bmatrix}$$

$$\frac{d}{dx}^T \frac{d}{dx} = \begin{bmatrix} \frac{-1}{h_1} & & & \\ \frac{1}{h_1} & \frac{-1}{h_2} & & \\ & \ddots & \ddots & \\ & & \frac{1}{h_{n-1}} & \frac{-1}{h_n} \\ & & & \frac{1}{h_n} \end{bmatrix} \begin{bmatrix} \frac{-1}{h_1} & \frac{1}{h_1} & & & \\ & \frac{-1}{h_2} & \frac{1}{h_2} & & \\ & & \ddots & \ddots & \\ & & & \frac{-1}{h_n} & \frac{1}{h_n} \end{bmatrix} = \begin{bmatrix} \frac{1}{h_1^2} & \frac{-1}{h_1^2} & & & \\ \frac{-1}{h_2^1} & \frac{1}{h_1^2}+\frac{1}{h_2^2} & \frac{-1}{h_2^2} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{-1}{h_{n-1}^2} & \frac{1}{h_{n-1}^2}+\frac{1}{h_n^2} & \frac{-1}{h_n^2} \\ & & & \frac{-1}{h_{n^2}} & \frac{1}{h_{n^2}} \end{bmatrix} = L$$

4.3.5. Choosing the regularzation parameter

The L-curve method

Trying to approximate the best estimation for the inverse solution, one tries to find the most suitable regularization parameter α. The regularization parameter is the weight given to the regularization term.

One method to choose the regularization parameter is by using the L-Curve method. The L-Curve method tries to find a compensation between the regularization error and the perturbation error. In the linear case of the least square problem one was trying to solve the problem $Ax = b$ and the minimization problem with regularization was $min||Ax-b||^2 + \alpha||Tx||^2$, using the L-curve method one would plot the residual vs the regularization $(log(||Ax-b||_2^2), log(||Tx||_2^2))$

The reason the method plots the terms in a log-log plot is to be able to focus on the major changes in the relationship between the perturbation and regularization.

The compensation between the regularization error and the perturbation error can be found by finding the maximum curvature in the graph as described in [10]. In the fig. 4.8 one can see a typical L-curve plot where the regularization functional T is the identity matrix.

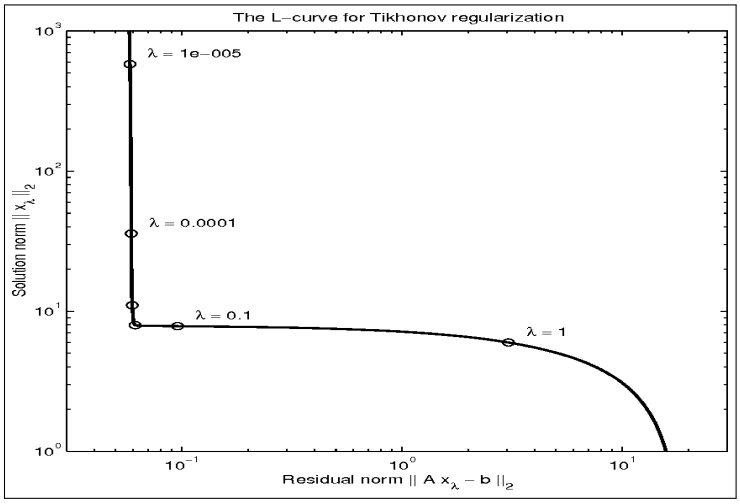


Figure 4.8: L-curve method, the figure was taken from Hansen [10], The Vertical part is dominated by the perturbation error and the horizontal part is dominated by the regularization error

In the graph, one can look at the bottom right corner where the regularization parameter is the largest, and see that the residual is constant. The regularization term tends to zero since the weight of the regularization term is big enough to cause the problem to try and **minimize** mainly this part. In the second part, moving with the curve to the left one can see a decrease in the regularization parameter and a reduction in the residual while the change in the regularization term is minor compared to other changes. In this part, one reduces the regularization and by that removing perturbations from the problem and getting closer to

the best approximation. In the third part (the vertical part), there is not enough reduction of the noise, and that causes the solution to be perturbations dominant. **Therefore, the best point is the point of maximum curvature, which is a minimization of the regularization term and the residual**. This point is between the second and third parts.

The work on the L curve method for the Non-linear Least Square problem can be found in the work of [7], where one could see that the same features of the L-curve exist in the non-linear case and are relevant as well. Therefore, in the case of this thesis where the problem is as in 4.2 one plots: $log(||Cu(k)-d||^2), log(||Lk||^2)$ To find the maximum curve, one can use the Triangle method.

The triangle method:

The triangle method is introduced in [2] The triangle method allows one to find the maximum curvature of the L-curve method within a set of measurement. First, the method checks the monotonicity of the points (norm of the residual decreases, and the norm of regularization term increases where the regularization parameter decreases). It discards any point that doesn't satisfy these conditions. Second, The method creates triangles between the last point measured and all sets of two other points. Then it tries to find the largest angle on the vertex of the triangle. The largest angle, which also holds the criteria of being smaller than $\frac{7\pi}{8}$, will be the point of maximum curvature. The method will also check the convexity at the point of interest by checking the orientation of the triangle using the determinant of the area.

4.4. Adjoint Equation

The adjoint derivation as in [19]. The left side of the minimization problem 4.2 is:

$$J_{ls} = \frac{1}{2} \parallel F(k) - d \parallel_2^2$$

where $F(k) = CA(k)^{-1}f$. One can write the left side as:

$$J_{ls} = \frac{1}{2} < F(k) - d, F(k) - d >$$

The derivative with repect to k_i is descried as $\nabla_i =< \frac{dF(k)}{dk_i}, r >$

This equation can be written as:

$$\nabla_i =< \frac{d}{d\tau} F(k + \tau e_i)|_{\tau->0}, r >$$

Now using the definition for $F(k)$ and the fact that C, f are not dependent on k_i the equation can be written as:

$$\nabla_i =< C\frac{d}{d\tau} A(k + \tau e_i)^{-1}|_{\tau->0} f, r >$$

The calculation for the inverse of the derivative is as follows:

$$AA^{-1} = I$$

$$\frac{dA}{dk}A^{-1} + A\frac{dA^{-1}}{dk} = 0$$

$$\frac{d}{dk}A(k)^{-1} = -A(k)^{-1}\frac{dA}{dk}A(k)^{-1}$$

Using the fact of the next relationship of the derivative:

$$\frac{d}{d\tau}A(k + \tau e_i)|_{\tau=0} = \frac{dA}{dk}e_i$$

$\frac{dA}{dk}$ is a tensor with size of $n \times n \times k_n$.

$\frac{dA}{dk}ei = \left(\frac{dA}{dk}\right)_i$ which is a matrix with the size of $n \times n$

Each entry of the third dimension of the tensor $\left(\frac{dA}{dk}\right)_i$ is the same as $\frac{dA}{dk_i}$.

$$\frac{dA}{dk}ei = \frac{dA}{dk_i} = \begin{bmatrix} \frac{da_{1,1}}{dk_i} & \frac{da_{1,2}}{dk_i} & \cdots & \frac{da_{1,n}}{dk_i} \\ \frac{da_{2,1}}{dk_i} & \frac{da_{2,2}}{dk_i} & \cdots & \frac{da_{2,n}}{dk_i} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{da_{n,1}}{dk_i} & \frac{da_{n,2}}{dk_i} & \cdots & \frac{da_{n,n}}{dk_i} \end{bmatrix}$$

from the definition of the derivative of the inverse and the definitiaiton of the derivative of the matrix A:

$$\nabla_i = < -CA^{-1}\frac{dA}{dk}e_i A^{-1}f, r > = < \frac{dA}{dk}e_i A^{-1}f, -A^{-1^*}C^* r > \Rightarrow \nabla_i = < \frac{dA}{dk}e_i A^{-1}f, -(A^{-1})^T C^T r >$$

Where the first equality comes from the inner product rules and the second equality comes from the fact that under $\mathbb{R}^n$ the next equality holds: $A^* = A^T$. Defining the adjoint as z in the next equation:

$$A^* z = -C^* r$$

and using the discrete equation $Au = f$ leads to the gradient:

$$\nabla_i = < (\frac{dA}{dk}e_i)u, z >$$

4.4.1. Gradient Computation

Using the additivity of the gradient:

$$\nabla = \nabla_{ls} + \nabla_{reg}$$

By definition of the gradient:

$$g_{ls} \equiv \frac{d}{d\tau}J_{ls}(k + \tau e_i)|_{\tau=0} \qquad i = 1, ..., n$$

As described in 4.4 yields to:

$$[g_{ls}]_i = \left\langle (\frac{dA}{dk}e_i)u, z \right\rangle_U, \qquad i = 1, ..., n$$

z will denote the costate equation:

$$A^*(k)z = -C^* r(k)$$

and $r(k) = F(k) - d$.

The gradient of the regularization term is simply:

$$\nabla_{reg} = (Lk)$$

4.4.2. Hessian Computation

Using the additivity of the Hessian:

$$H = H_{ls} + H_{reg}$$

The evaluation of the matrix-vector product can be done using finite differences. By applying the matrix vector product to the standard basis vector e_i. One gets:

$$H(k)e_i \approx \frac{g(k + \tau e_i) - g(k)}{\tau}$$

H, g denotes the Hessian and Gradient respectively. Now the Hessian can be constructed using the next fact:

$$[H(k)]_{i,j} = \langle H(k)e_i, e_j \rangle$$

The regularization functional Hessian is simply the matrix $H_{reg} = L$

4.5. Numerical Differentiation

To find the Gradient and Hessian of a function one can use Numerical Differentiation. Numerical Differentiation is a central difference method applied on a function. It can be described as:

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

h can be chosen as $h = \epsilon^{\frac{1}{3}}$ where ϵ is the machine epsilon which is around 10^{-16}.

4.6. Adjoint Method and Numerical Differentiation

To check the validity of the adjoint method, one can compare the Adjoint method and numerical differentiation to compute the Gradients. The advantage of using the Adjoint method compared to Numerical Differentiation is that the Adjoint method is much less expensive. To compute the Adjoint method, one needs to solve 1 PDE, which includes Matrix assembly, LU factorization, and inversion of the LU. And adding to that one LU linear equation solve in order to get the adjoint. In Numerical differentiation, one needs to solve 2N PDE's. A one PDE solve for the cost function and 2 cost functions for each degree of freedom.

First, one can see that both methods match:

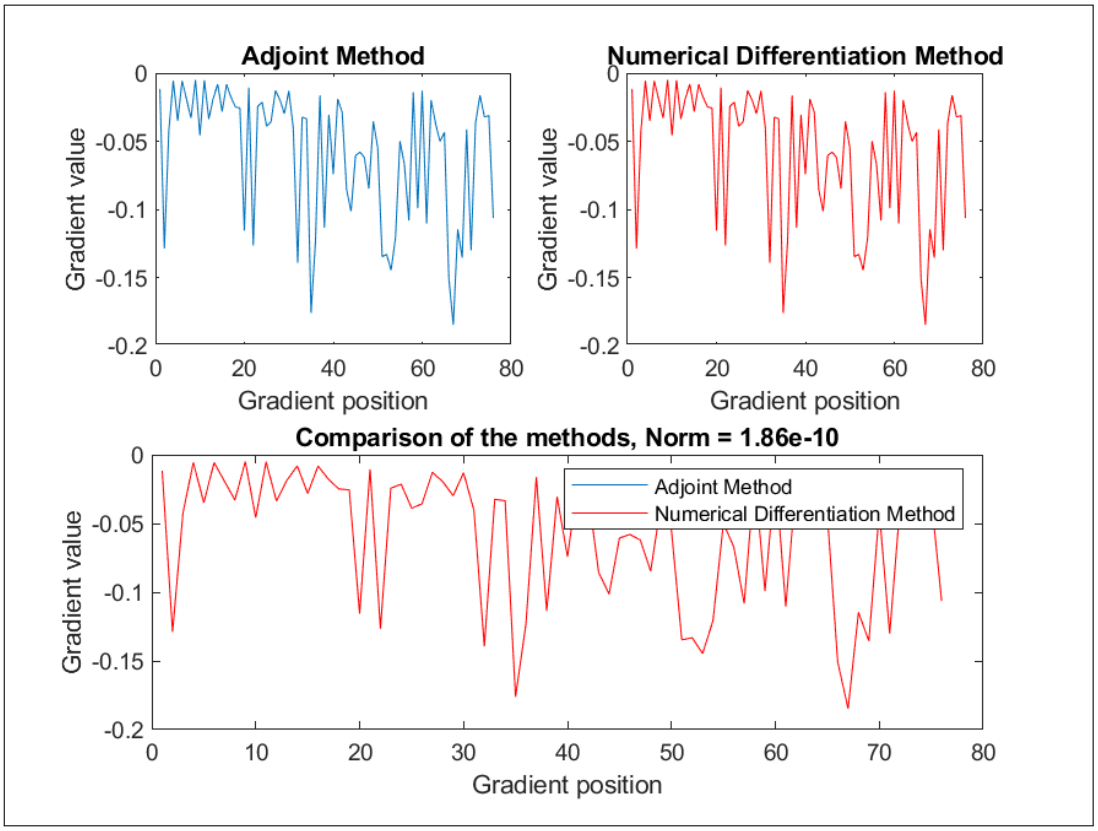


Figure 4.9: System variable for different alpha, Number of elements is 76

Second, looking at the CPU time needed to compute both methods: Numerical Differentiation takes: 2.3 - 3 minutes, where the Adjoint method takes 0.1-0.2 minutes.

4.7. Solving Non-Linear Least Squares Problem

The minimization of the Least Squares method is a criterion that helps approximate a solution to match a function output to a set of observations. The minimization problem will find the inputs of the problem that can cause the smallest value in the least-squares sense. The list square sense is the summation of the squared differences between the outputs of the function and the observations.

To solve the Least Squares minimization problem, one can use different non-linear optimization methods during the scope of this thesis, two methods will be used. The first is the **Newton method**, and the second is the **Newton with Trust Region method**.

4.7.1. Cost of computation

The computation effort needed to compute the cost function is a 1 PDE solve for **each source**. After computing the cost function, the gradient can come for free since the residual is already calculated in the cost function calculation, and the adjoint can be computed in a much easier manner when using the matrix factorization for the stiffness matrix used in the cost function computation. Hessian computation has two parts. The first one is by using the gradient that was already computed. The second part is the calculation of gradient in each degree of freedom. The cost of computation is mentioned in tab. 4.3

1_{LU}	1 LU Factorization
$1_{Matrix\ assembly}$	1 FEM matrix asembly
$1_{LU\ linear\ equation\ solve}$	1 LU linear equation solve
$1_{PDE\ Solve}$	$1_{LU} + 1_{Matrix\ assembly} + 1_{LU\ linear\ equation\ solve}$
$1_{Adjoint}$	$1_{LU\ linear\ equation\ solve}$
N_{∇}	Number of degrees of freedom
$N_{experiments}$	Number of experiments
$Cost$	$1_{PDE\ Solve}$
$Gradient$	$(Cost + 1_{Adjoint}) \cdot N_{experiment}$
$Hessian$	$Gradient + N_{experiments} \cdot N_{\nabla} \cdot (Gradient)$

Table 4.3: Gradient and Hessian cost table

4.8. Convergence Result

4.8.1. Convergence

Denoting n as the number of experiments, and assume one collects the next measurements:

$$d_n = F(k_{true}) + \eta_n$$

Where d is the measurement, $F(k)$ s the true value and η is the noise. Assuming that the noise $||\eta|| = \delta_n$ and converge to zero as $n \to \infty$. By assuming the next three assumptions one can achieve convergence:

$$\alpha_n \to 0 \qquad as \qquad n \to \infty$$

$$\frac{\delta_n^2}{\alpha_n} \to 0 \qquad as \qquad n \to \infty$$

$$F(k) \neq F(k_{true}),\ whenever:\ k \neq k_{true}$$

A proof can be found at [19]. For the scope of this thesis, the last criteria will not be met since measurements will not be taken at all grid locations. Therefore, the exact solution using an inverse problem is not expected to be reached.

5

Multi-Level Method

5.1. Introduction

5.1.1. The Multi-Level approach

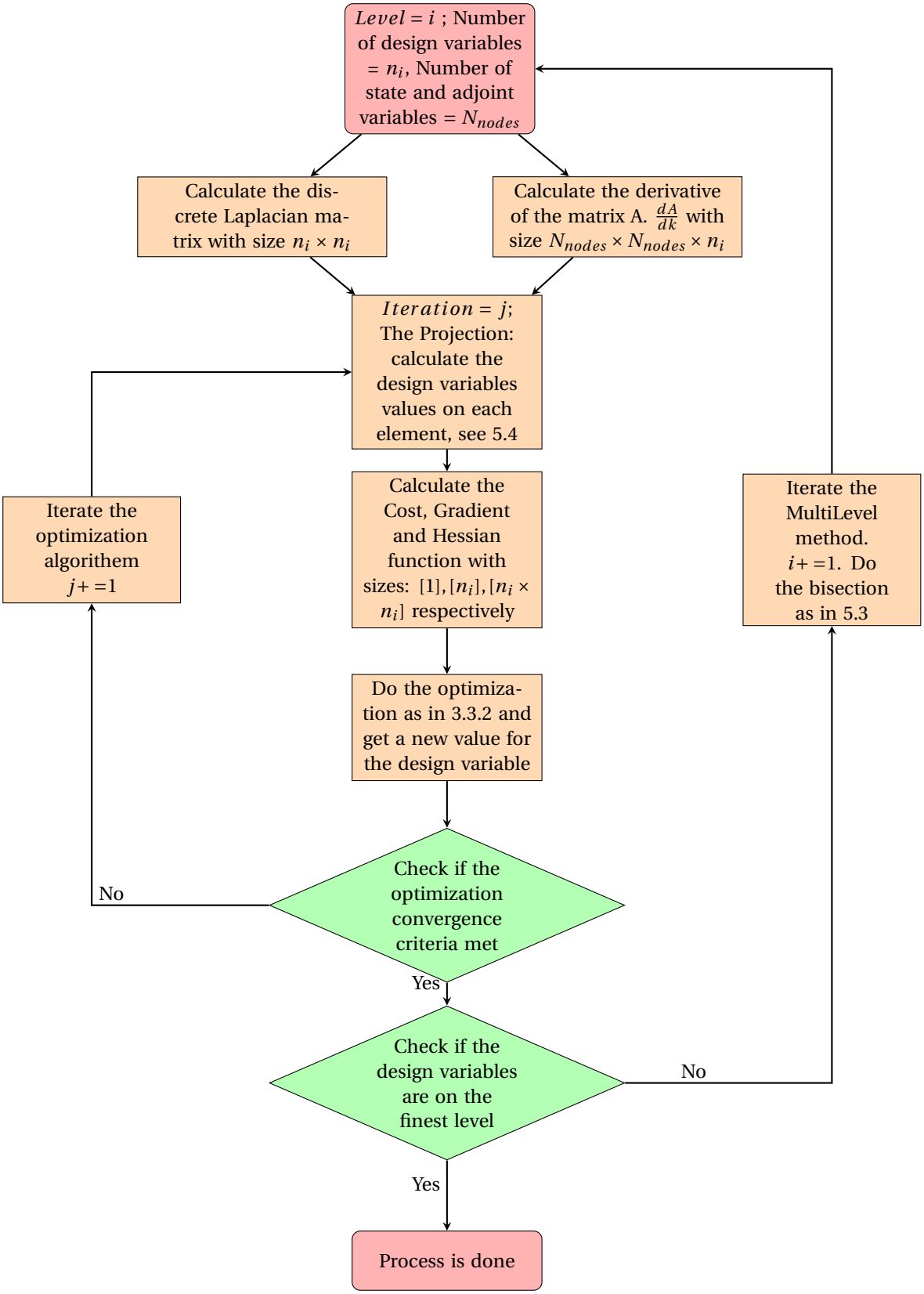
The Multi-Level approach helps to reach a faster convergence yet accurate results compared to the methods described in 4, where a single-level method is described. The Multi-Level approach is an iterative optimization solver where, for each level, the design variables space changes from coarse to fine. The state variables stay at the finest scale for all iterations. The reduction of degrees of freedom of the design variable has a positive effect on the direction the optimizer "is going" while looking for a solution and forces less computation as well. The Multi-Level gives a faster yet reliable approach to solve the Poisson equation.

The calculation of the cost and gradient functions include solving a partial differential equation in the state variable space. Therefore, on each level, a projection of the design variables is made to the state variable space. While moving between the different levels, the fining of the design variables space is being done by bisecting the grid of design variables. For example, in the one dimension in the state space problem, each element is bisected at the center to two elements.

5.1.2. Why is it faster

The multi-level approach is faster due to two main reasons. First, while iterating the reduction of the degrees of freedom allows the solution on the coarse level to have similar shape of the real solution and to highlight the important difference in the different grid cells, later when fining the level a solution can be reached in an easier manner since the initial guess from the last level includes some information of the important features in the solution. Second, it involves fewer degrees of freedom when computing the gradient and especially when computing the Hessian, where the number of degrees of freedom can cause a very high computation need.

5.2. The Multi-Level Algorithm



5.3. The Bisection:

The Bisection is the method of moving between the different levels while iterating in the algorithm. The main idea is that when fining between the levels, the new center of the element gets the value of the design variable, it used to be inside in the coarser level.

5.3.1. 1D in space

Figure 5.3 shows how, on each level, the number of design variables is multiplied by two, and each design variable is cut to two at the center of the design variable line. The value of the design variables carries on when bisecting. For example, when moving from the coarse level to the medium level, one can see how k_1 in the medium-scale will be at the same locations of earlier k_1 in the coarse scale.

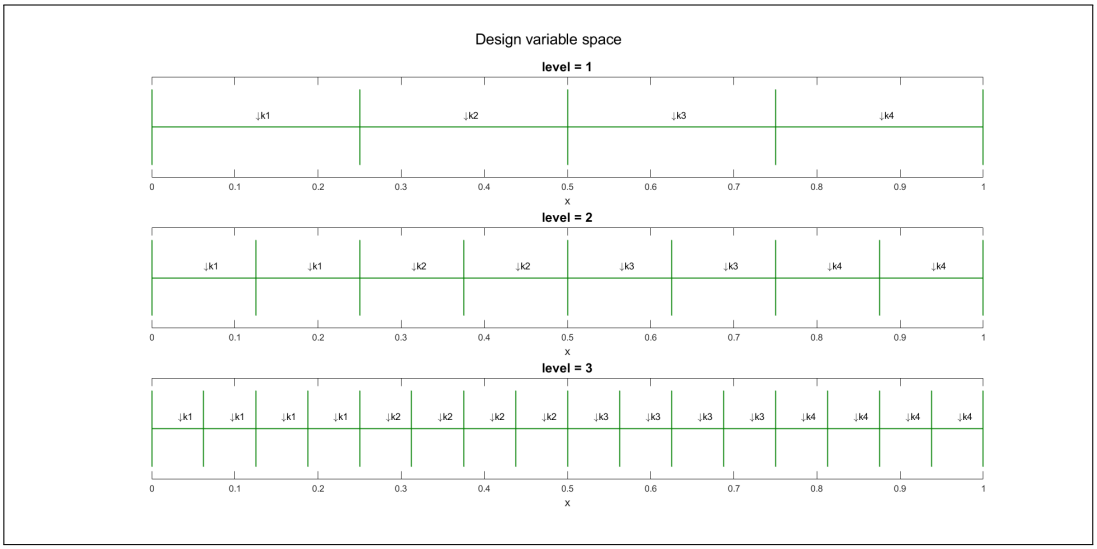


Figure 5.1: Bisection in the 1D state space for 3 different levels

5.3.2. 2D in space

Figure 5.2 shows how, on each level, the number of design variables is multiplied by four, and each design variable is divided into four equal-area squares. The value of the design variables carries on when bisecting. For example, when moving from the coarse level to the medium level one can see how k_1 in the left top corner is projected to the medium-scale at the same locations of the coarse-scale (i.e. in the medium-scale it is the 4 top left squares)

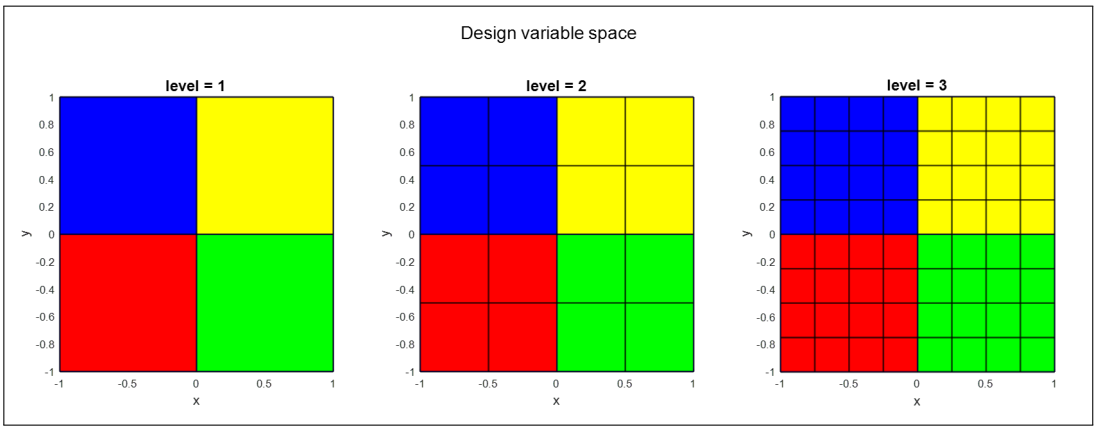


Figure 5.2: Bisection in the 2D state space for 3 levels

5.4. The Projection Prolongation operator:

The projection is needed in order to be able and solve the elliptic PDE on the state variable space. The idea is to project the design variables values to each element in the state space. The Projection checks what is the center of each triangular element in the state space, check where this center will be located in the design variable space and assign that design variable value to the state space element.

5.4.1. 1D in space

Figure 5.3 illustrates 1D in space case. The grids are different for the state and design spaces. The space grid is at the finest level. 3 levels are described for the design variables where the Coarse, Medium, Fine levels have 4, 8, 16 design variables, respectively. The value of each design variable is described as k_i. The elements in the space grid are described as e_i. The projection of the design variables k_i on the state variables in space is described as well where one can see that for example at the Coarsest level elements e_1, e_2, e_3, e_4 will have a design variable value of k_1.

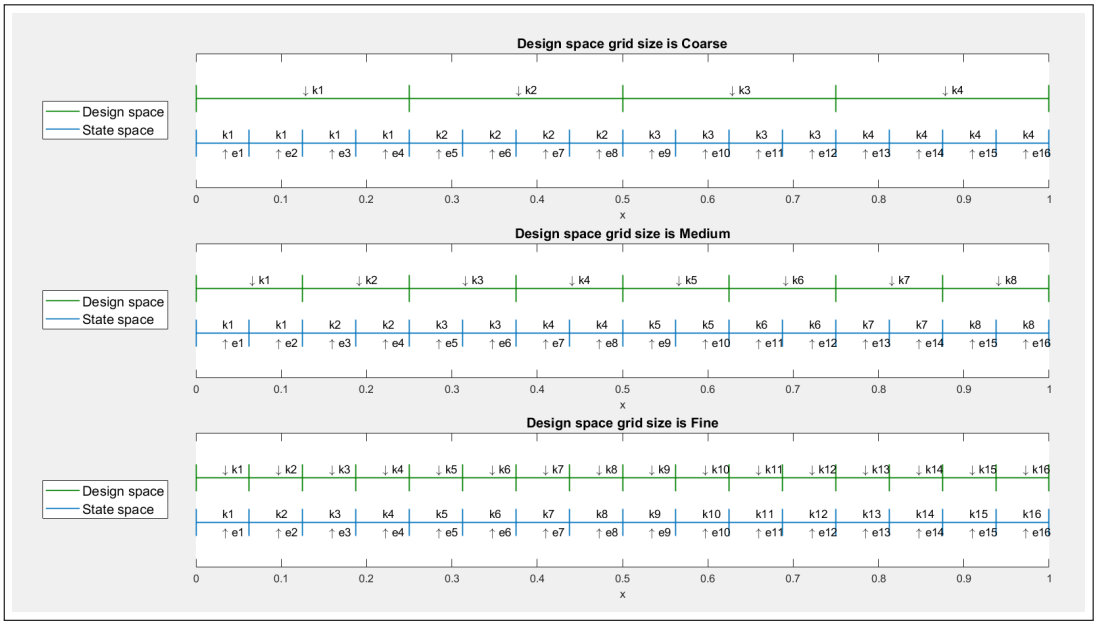


Figure 5.3: Projection of design variable mesh to the state variable mesh in the 1D state space

5.4.2. 2D in space

Figure 5.4 illustrates 2D in space case. The grids are different for the state and design spaces. The state grid is at the finest level. Three levels are described for the design variables where the Coarse, Medium, Fine levels have 4, 16, 64 design variables, respectively. The value of each design variable is described as k_i. The elements in the space grid are described as e_i. The projection of the design variables k_i on the state variables in space is made by taking the center of the triangular element and checking inside which design variable square it is. Then the value of k_i will be equal for the whole element.

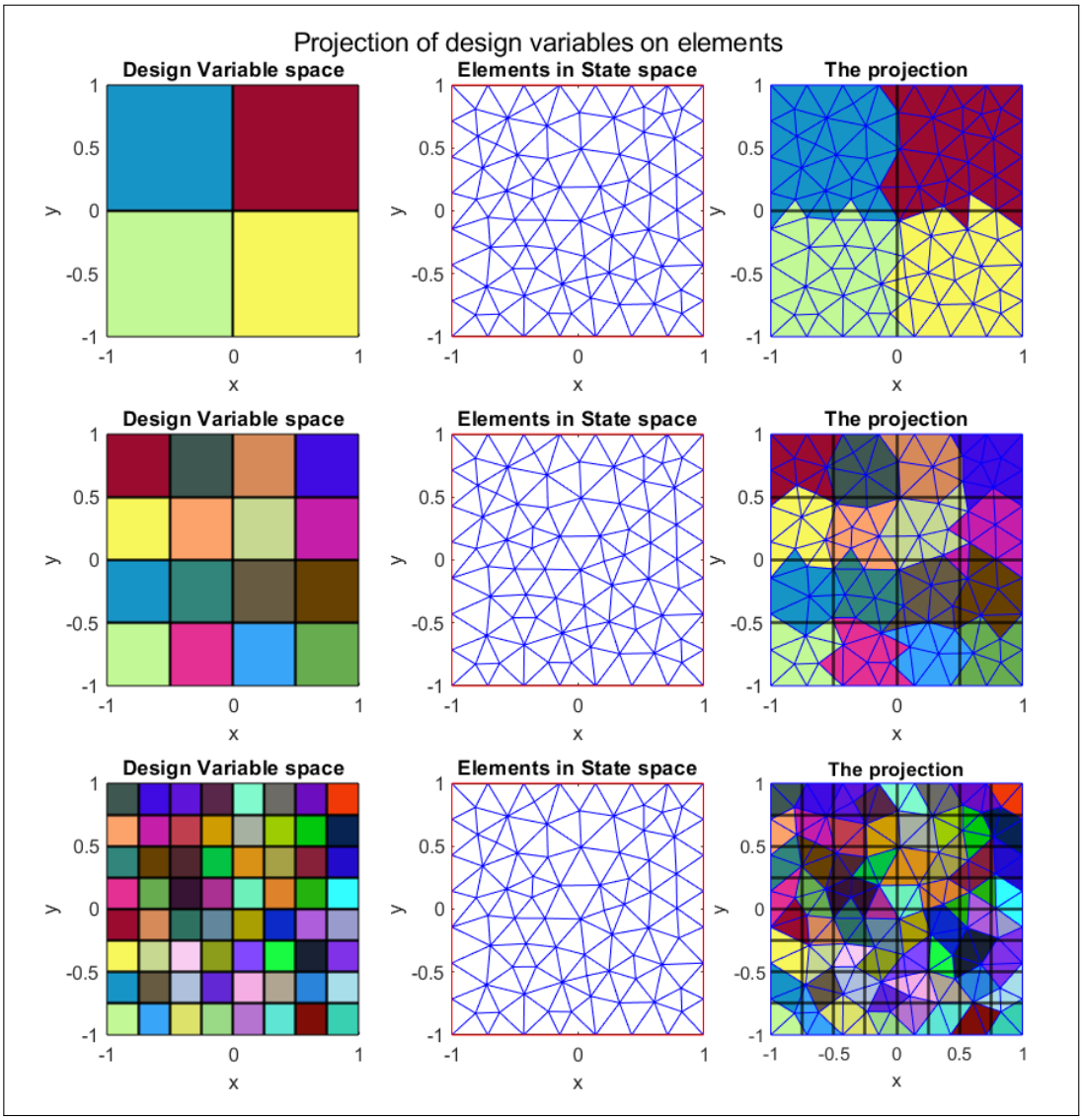


Figure 5.4: Projection of design variable mesh to the state variable mesh in the 1D state space

6

Numerical Results for the Poisson Equation

6.1. Introduction

Julia language is a high-level programming language that was launched in 2012. Julia was designed for high-performance computation in science applications. Julia is an open-source, free language. Julia allows the user to write the code easily but still allows an efficient code. Further, Julia supports different options for the use of libraries, packages, and functions from other programming languages such as C, Fortran, Phyton, Matlab, etc. Julia compiler is doing the compilation just-in-time, which allows the software to compile while running the program. Parallelization is a major advantage in Julia, which allows different methods of parallelization, including parallelization from the command line.

6.2. Julia as a programming language

Julia language is a high-level programming language that was launched in 2012. Julia was designed for high-performance computation in science applications. Julia is an open-source, free language. Julia allows the user to write the code easily but still allows an efficient code. Further, Julia supports different options for the use of libraries, packages, and functions from other programming languages such as C, Fortran, Phyton, Matlab, etc. Julia compiler is doing the compilation just-in-time, which allows the software to compile while running the program. Parallelization is a major advantage in Julia, which allows different methods of parallelization, including parallelization from the command line. Multiple dispatch is a key point in the programming paradigm of Julia. Multiple dispatches allow a function or method to be called, and while the program runs, it can dynamically decide what process of the function to run based on attributes of the argument [1]. Julia includes a built-in package manager that includes over 2400 registered Julia packages. The packages allow the user to save time and the user doesn't need to program all from scratch.

6.3. Implementation in Julia

During the work of this thesis, the code was implemented using the Julia language. The main packages used are JuliaFEM, Optim, LinearAlgebra, MATLAB, SparseArrays and DelimitedFiles.

- JuliaFEM - to calculate finite element models [6]. The JuliaFEM allows the user to create a PDE problem with the type of his choice. Then the user creates the elements as wishes with the desired parameters in each element. Such as geometry, conductivity(/permeability), flux, boundary conditions, etc. The last step is to assemble the matrixes which JuliaFEM has a built-in function for.
- Optim - to use the Trust region Newton method algorithm [13]. The function allows the user to build his gradient and hessian functions and to use them as part of the optimizer. Further, it enables the user to choose between a few optimization algorithms
- MATLAB - to use built-in MATLAB functions that don't exist in Julia such as generating unstructured meshes.

- DelimitedFiles - Saving files as a tab-delimited files. It was mainly used to be able to compare cases to Matlab functions and also to use the outputs of the problem by using some of the plotting tools MATLAB allows

- LinearAlgebra - to use special Linear Algebra functions. Used to compute LU decomposition, eigenvalues, SVD, Transpose, etc.

- Sparse Arrays - Operations on sparse arrays. Used to do Linear Algebra operations in sparse arrays in order to save computation time for the Linear Algebra algorithms

6.4. Numerical Results in 1D

6.4.1. Inversion results and comparison of the optimization methods

In figure 6.1, one can see the inversion solution of the Newton and Newton with Trust Region methods. Both solutions match to each other. Moreover, although one solves a minimization problem, which includes the regularization term, it still manages to reach a good approximation to the exact solution.

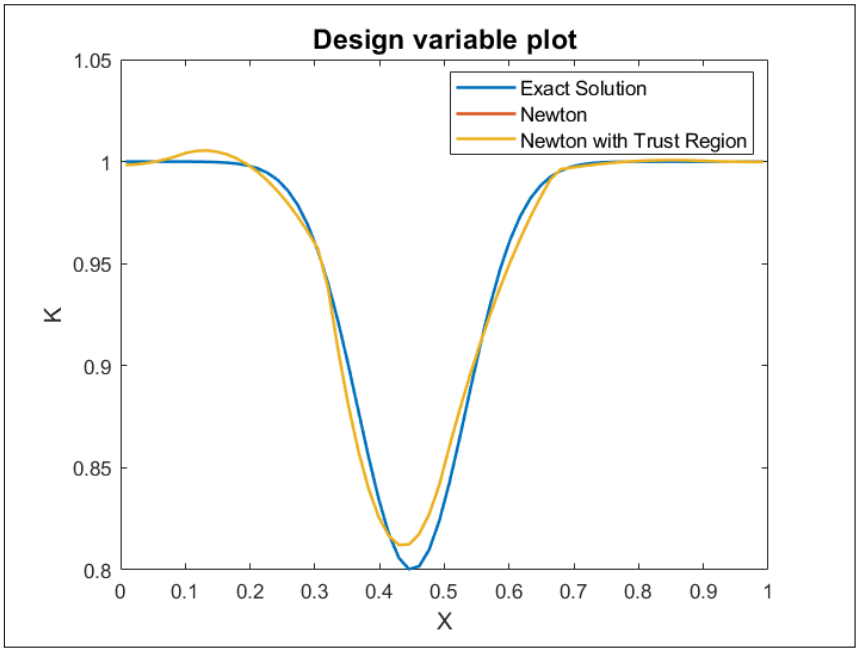


Figure 6.1: Comparison of the Inverse problem solution and the exact solution used to simulate the forward model. Regularization parameter used is $\alpha = 1e-7$, Norm of solutions: $||TrustRegion - Exact|| = 0.06044$, $||Newton - exact|| = 0.06044$

In figure 6.2 a comparison of the two methods for a lower regularization parameters $\alpha = 1e-9$ with the same convergence criteria. The Newton solution doesn't manage to converge while the Newton Trust Region converges in 13 iterations. It shows how robust is the Newton Trust Region method is. For the newton solution, the lack of weight on the regularization term causes the function not to converge and actually after some iteration to travel to incorrect solutions. In a more detailed look, one can see that at iteration number 7, the newton solution travels too far and the quadratic approximation is not good. Since the quadratic approximation is not good the Trust Region method stays constant and decreases the region. Then in iteration number 11 the newton method travels too far and from there it cannot converge to a solution that meets the cost function convergence criteria.

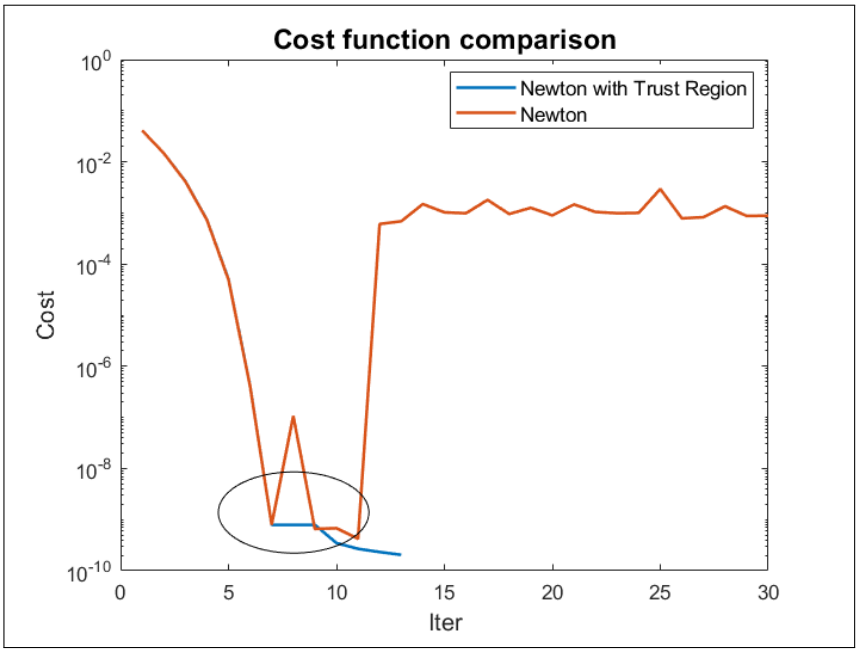


Figure 6.2: Newton and Newton Trust Region cost function comparison with a regularization of $\alpha = 1e-9$

Figure 6.3 the evaluation of the trust-region solution can be studied. In the figure, it can be seen that until iteration number seven, the cost function decreases between iterations, and the region of the trust region method is constant. This is because the approximation to the objective function is good enough by the criteria set earlier. After iteration seven, one can see that the cost function is constant, and the region decreases. It is because the quadratic approximation is not good enough. Therefore no step is being done to compute a new design variable solution and with it a new cost function value. Instead, the region is decreased until the quadratic approximation to the objective function is good enough at iteration 9. There the cost function decreases, and the region is constant.

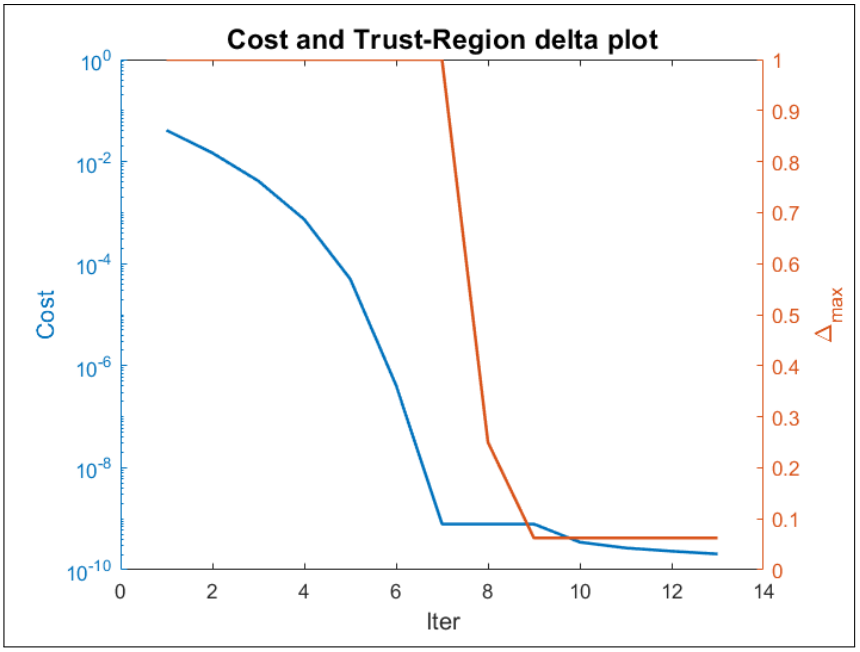


Figure 6.3: The different cost function value and the Newton Trust Region region are calculated for each iteration. The regularization parameter used is $\alpha = 1e-9$

Figure 6.4 the evolution of the trust-region solution for the design variable k is presented. It takes 13 iterations to reach convergence using the Newton Trust Region method and the final $||error|| = 0.1024$. In each iteration, the previous iteration solution is used as the initial guess.

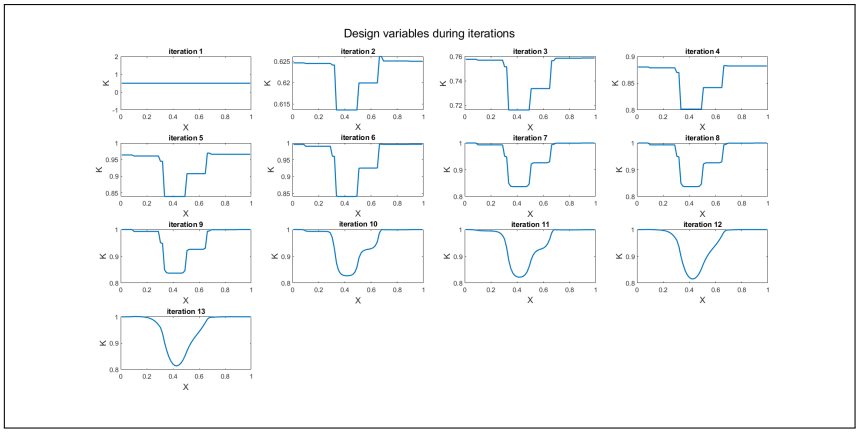


Figure 6.4: The regularization parameter used is $\alpha = 1.56e-9$, the different evaluations of the design variables are presented for the different iterations

6.4.2. Regularization parameter α study

To understand better the effect of the regularization parameter, a test for different values of the regularization parameter α was done in figure 6.5. One can see that when using a bigger value of regularization parameter α the dominant part of the objective function is the regularization part, which causes to minimize mainly based on the regularization criteria. The result is a constant solution to all the one-dimensional space. The more weight applied to the regularization, the more the first derivative of the design variables as a function of space is minimized. By trying to minimize the first derivative one tries to get $\frac{dk}{dx} \rightarrow 0$. Which therefore leads to a constant value for $k(x)$.

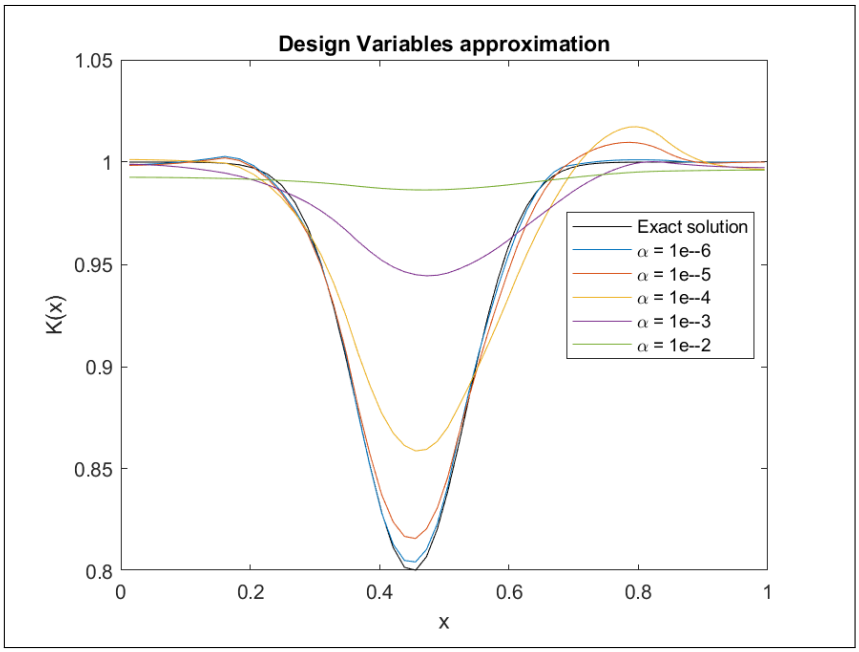


Figure 6.5: Design variable for different regularization parameters α values, as the regularization parameter increases the solution tends to be more constant

Figure 6.6 compares the case where all nodes are an observation point: $\alpha_{reg} = 0, ||error||_2 = 3.4778e-07$. The results are that although no regularization is applied, the results still match. It is because of all grid points

in space include observation. That solves the uniqueness issue that comes with the inverse problem. In the case the noise in the observation will decrease to zero a convergence can be fulfilled as in 4.8.1.

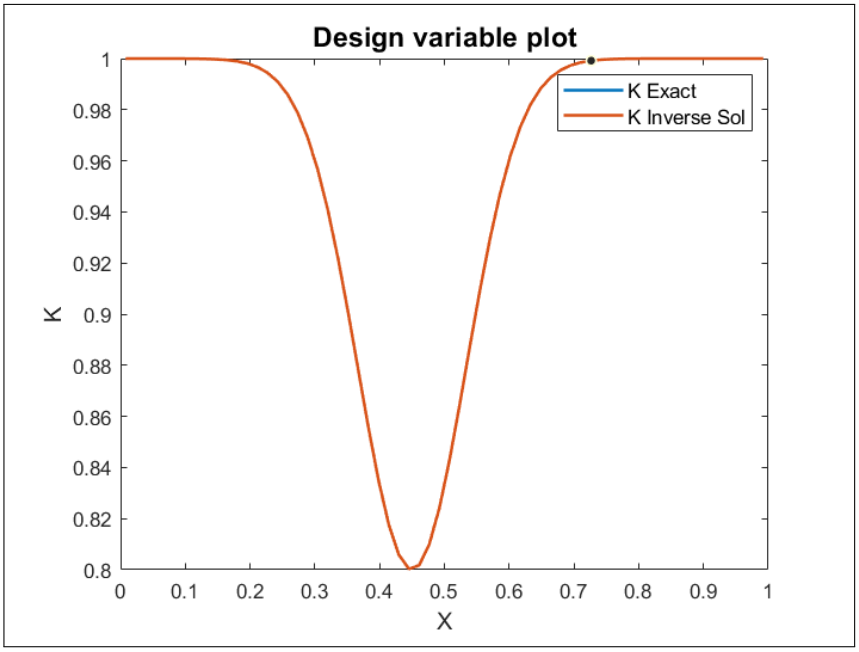


Figure 6.6: Observation at each location of the grid point helps getting closer to convergence, no regularization is needed

6.4.3. L-Curve method

In fig. 6.7, the L-curve method in 1D is described. The triangle method helps to find the best regularization parameter α. The left figure represents the raw values of the (log(residual),log(regularization)) the largest regularisation parameter α is at the bottom right of the L-Curve. At the right figure, the triangle method removes all points that are not monotonic (decrease in α, decrease in the residual, and increase in the regularization term). In fig. 6.7 the best regularization parameter that leads to the best compensation between the residual and the regularization α is the one to the top left at the right subfigure.

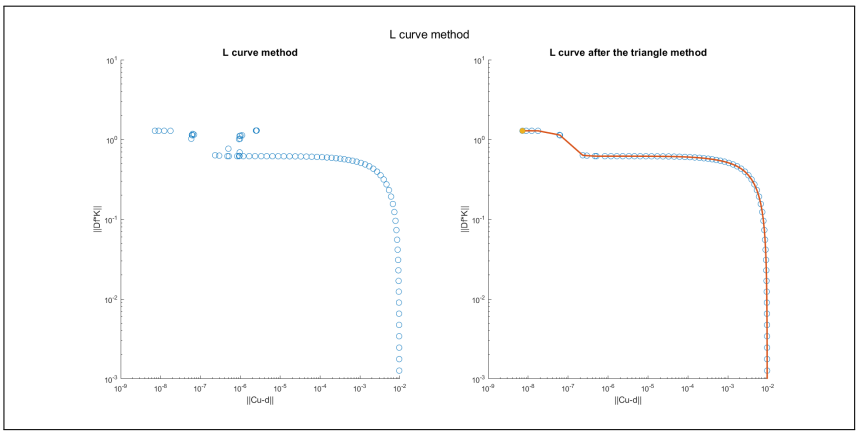


Figure 6.7: The left figure represents the raw graph of the residual and the regularization function for different regularization parameters α, The right figure represents the L curve values after applying the triangle method

6.4.4. Solution using the MultiLevel approach

Figure 6.8 shows the different design variables values during the different iterations of the Multi-Level algorithm. The Multi-Level allows one to get closer to the raw function quicker. As can be seen in the figure,

the important features of the graph and the general shape can be found even at the first and coarsest level. Comparing the evolution of the solution of the Multi-Level to fig.6.4 where the Single-Level method is used it can clearly be seen that at the Single-Level at the first iterations the raw shape of the solution isn't reached. In fig. 6.9 one can see the effect of the coarse level in the Multi-Level methods on the cost function. With a limited number of degrees of freedom at the coarsest level, one doesn't need to do a lot of computation effort to compute the Hessian. At a certain point with enough iterations, both methods match as expected.

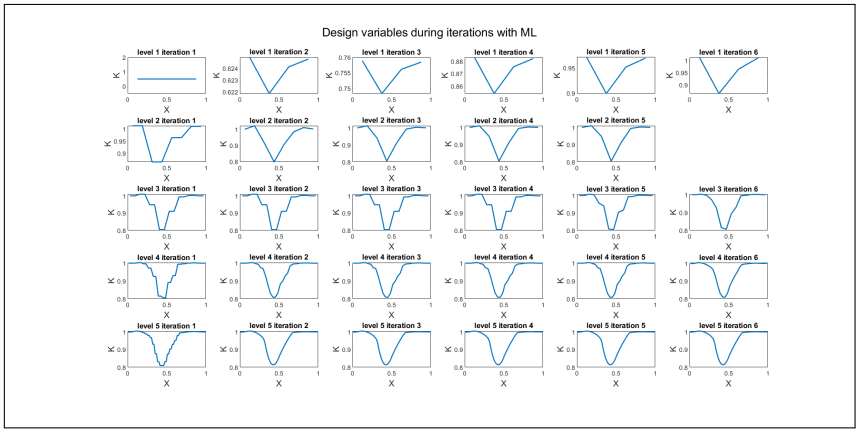


Figure 6.8: Mulit-Level method using Newton with Trust Region, the evolution of the design variables during the different iterations

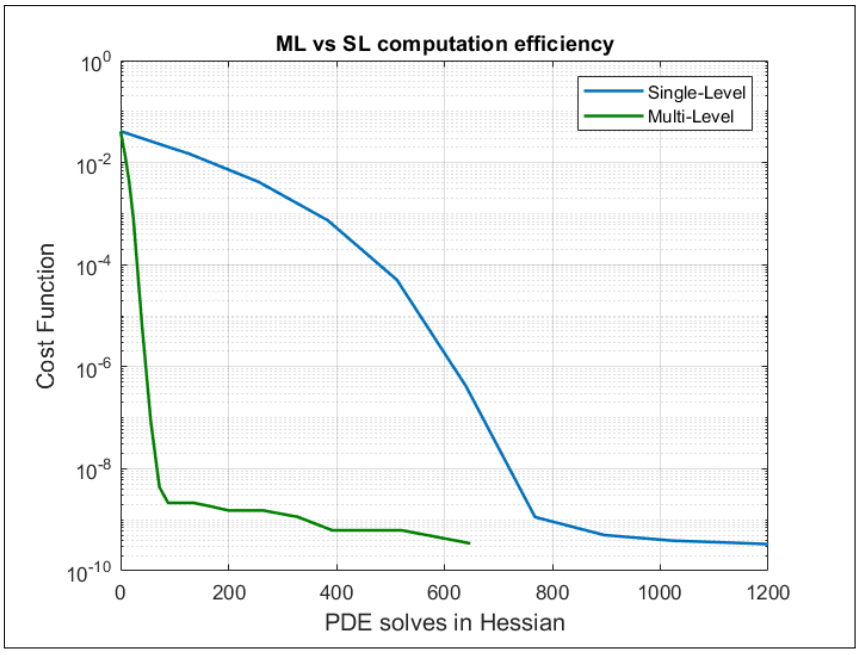


Figure 6.9: Cost function vs the amount of PDE's solve in the Hessian, comparison between the Single-Level and the Multi-Level

6.5. Numerical Results in 2D

6.5.1. 1 design variable

In this subsection, a 2D test case with 1 design variable case is presented. The design variable value is $k = 3$ The discretization will include 76 elements, as can be seen in figure 6.10. The optimization method used will be the Newton Trust Region method.

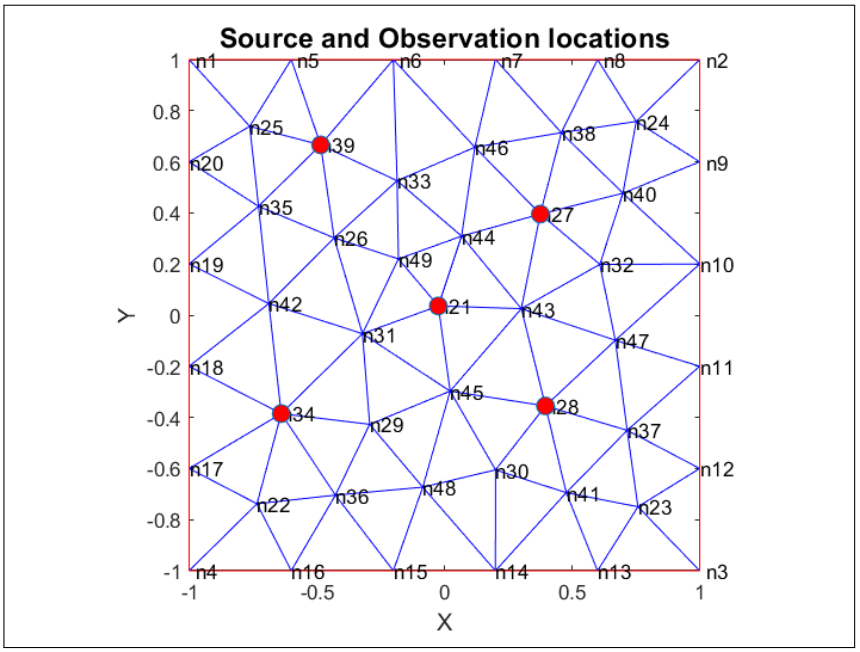


Figure 6.10: The mesh for a 2D case with sources and observations on the same locations denoted as red circles

Figure 6.11 shows how the solution progresses starting from an initial value of $k_{initial} = 100$ moving to the left until it finishes close to $k = 3$. To explain the Newton quadratic approximation, one can view figures 6.12,6.13 where a quadratic approximation at a certain iteration is plotted. The green line, which represents the quadratic approximation function, is also cut at the limit of the region that is found in the Trust Region algorithm.

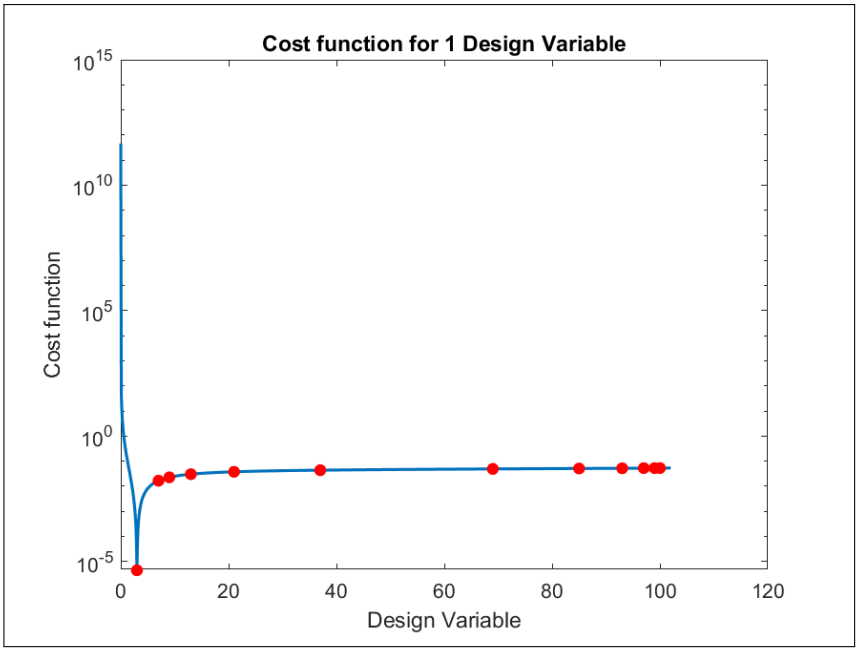


Figure 6.11: Cost function for 1 Design Variable and the iterative solution for the design variable with each iteration denoted in red circles. The iterations start at the most right red point k = 100

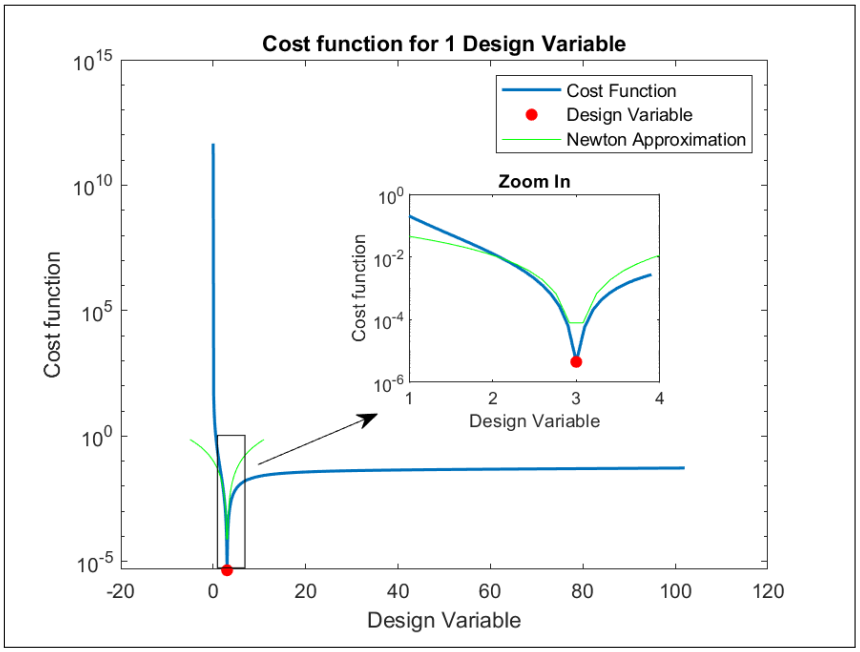


Figure 6.12: The quadratic approximation to the cost function is represented by the green line. The line is being cut at the end of the Newton Trust Region region

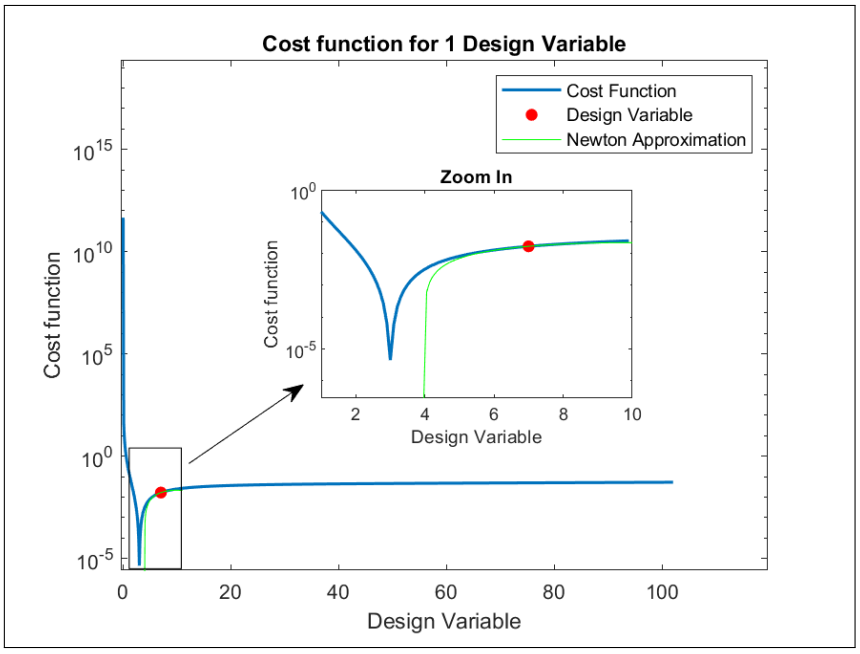
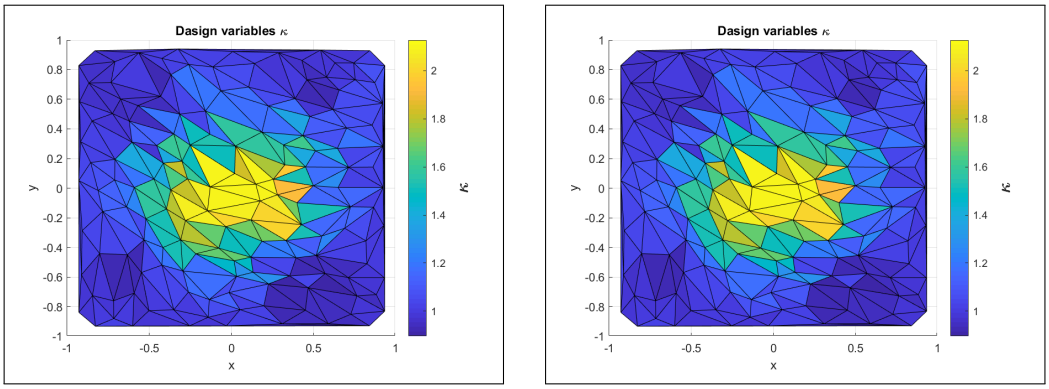


Figure 6.13: A quadratic approximation to the cost function is represented by the green line. The line is being cut at the region of the Trust Region method

6.5.2. General number of design variables

Looking at example 4.2.2 and solving it using Newton and Newton Trust Region methods where a regularization parameter of $\alpha = 1e-6$ is used. In fig. 6.14 and fig. 6.15, a comparison of the two methods is done for the inversion. One can see that the solutions are almost the same.



(a) Newton with Trust Region solution

(b) Newton Solution

Figure 6.14: Compare the solutions for the inverse problem for a case of $alpha$ = 1e-6

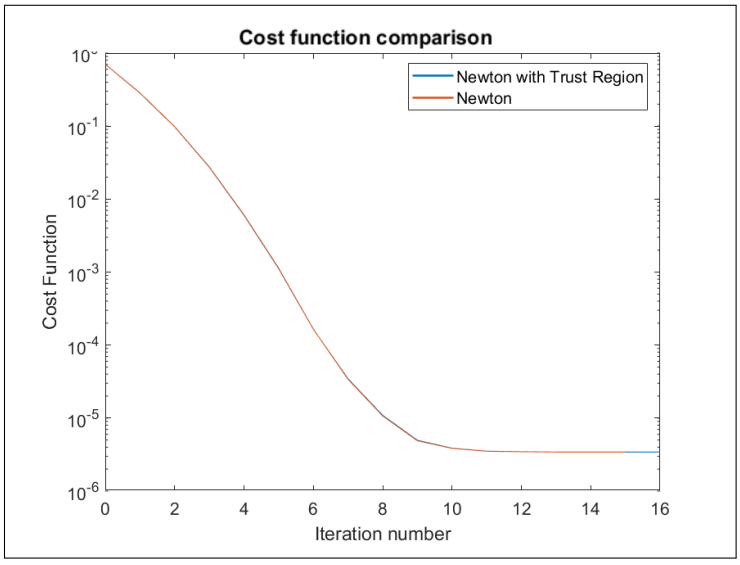


Figure 6.15: Compare Cost Solution for the 2 methods. The results are almost the same and graphs lie on each other

To test the difference between the methods, one can test a smaller regularization parameter $\alpha = 1e-8$. Figure 6.16 shows that until iteration number 3, the solution is the same, but at this iteration, the Newton method step is too big, and the solution jumps much more than needed. There it gets to another local minimum, and it gets stuck on the local neighborhood of a wrong minimum.

On the other hand, looking at figure: 6.17, the Trust region algorithm doesn't increase the region after iteration number 3, which helps the solver to stay on the right track towards the "good" valley - the smallest minimum. After iteration 5, the cost function is constant since each step is being rejected, and the Trust-Region algorithm decreases the region Δ_{max} until the quadratic newton approximation is good enough to keep iterating to convergence.

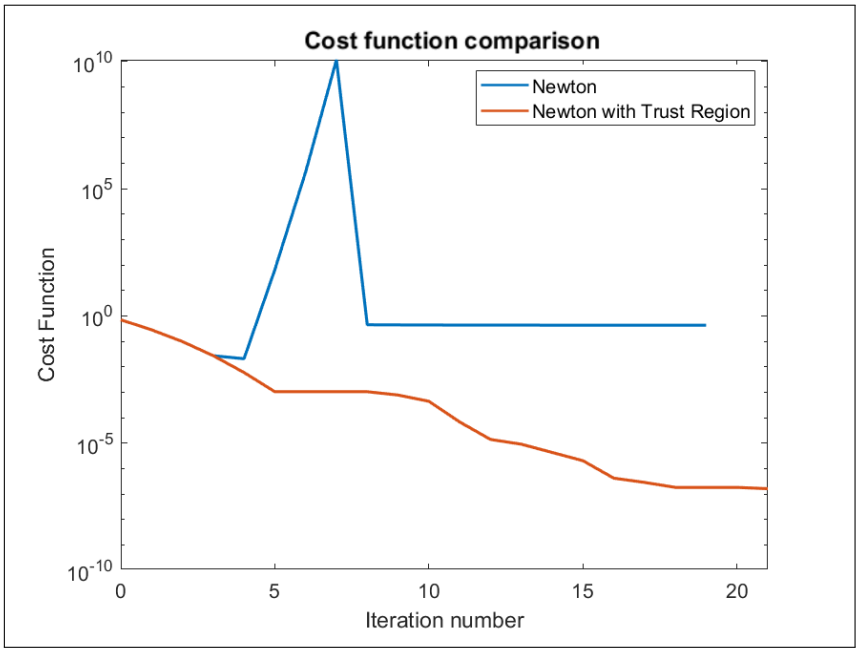


Figure 6.16: Compare Cost function for different iterations, where: $\alpha = 1e-8$

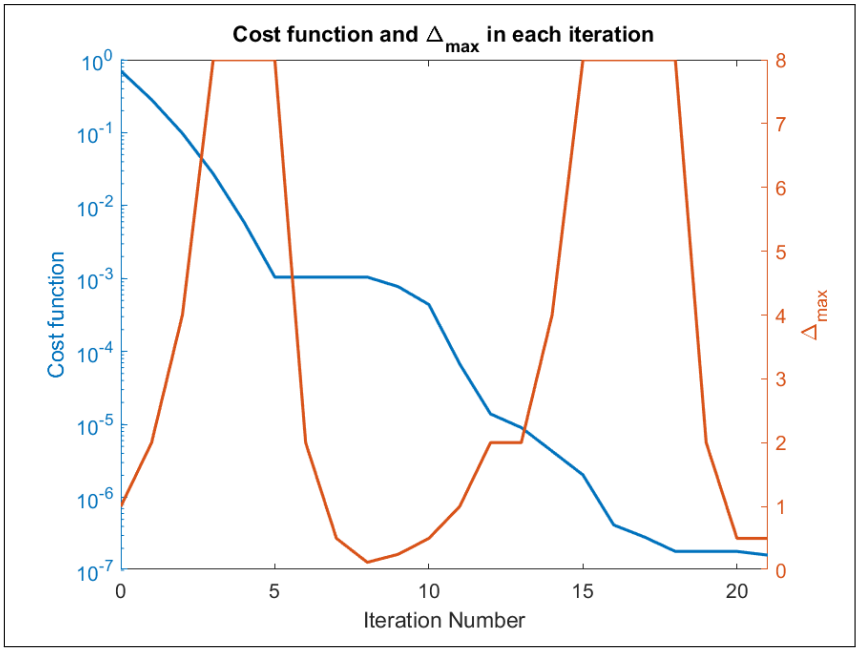
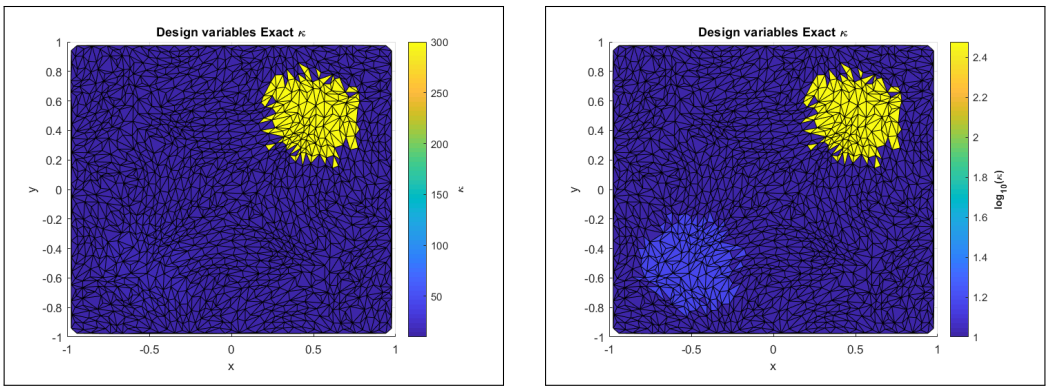


Figure 6.17: Compare Cost Solution and Δ_{max} for $\alpha = 1e-8$

6.5.3. Benchmark problem

The benchmark problem as illustrated in chapter 2 contains a permeability field with two features. One with very high permeability ($k = 300$) and the second with a slightly higher permeability ($k = 15$) compared to the field ($k = 10$). The location of sources and observations is as in 4.2.2, this time a much finer mesh will be used. The number of nodes is 725 and the number of elements is 1368. The problem will be solved using the Newton with Trust Region method.

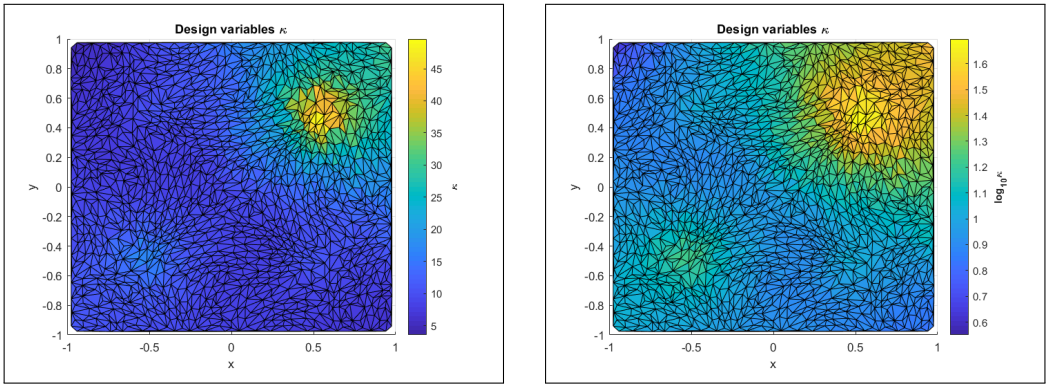


(a) Benchmark problem design variables

(b) Benchmark problem design variables in a log scale

Figure 6.18: Benchmark design variable exact solution

Figure 6.19 presents the solution for the case where a regularization parameter of $\alpha = 1e-8$ is used. The inverse solution manages to capture both locations of different permeabilities but it doesn't manage to converge to the very high permeability value.



(a) Inverse Solution

(b) Inverse Solution in a log scale

Figure 6.19: Inverse solution for the benchmark problem

Due to the use of a fine grid the computation time increased tremendously. The big increase in the degrees of freedom caused the Hessian computation to be very long. In order to test whether a Hessian free method can be used, a Conjugate Gradient (CG) method [15] was tested. The Conjugate Gradient method needs only a gradient function in order to conduct the optimization and therefore is fitted for the purpose of the test. Figure 6.20 shows the result of the CG method. The CG method didn't converge to the approximate solution at all. The CG algorithm has a maximum iteration criteria of 50 iterations while using the Newton Trust Region method a 25 maximum iteration criterion was used.

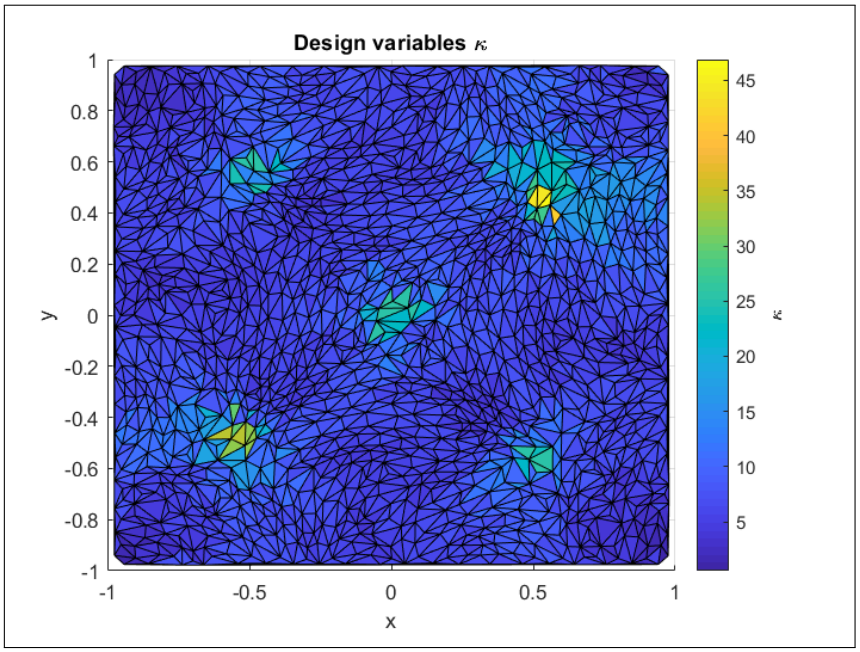


Figure 6.20: Solution with CG

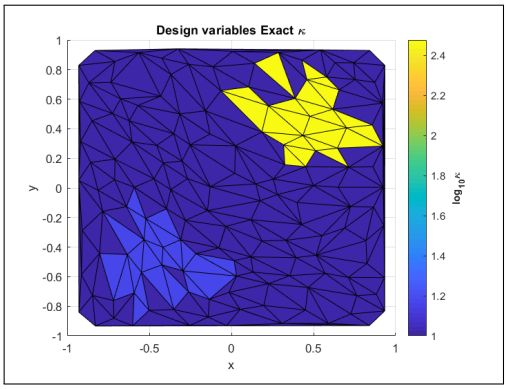
Another try can be the use of **Hessian approximations** in order to reduce the computation time of the Hessian. Using the BFGS algorithm one can estimate the Hessian matrix. Doing so for a small problem with 36 elements and testing based on the benchmark problem exact solution for the modeling. Doing one iteration using the Newton Trust-Region algorithm where the Hessain is being approximated with finite difference method. To calculate the Hessain of the second iteration two methods are used one with finite difference and one with BFGS algorithm. The BFGS algorithm used can be found at [15]. Table 6.1 shows the differences in the approximations and the design variable solutions using the two methods.

$\|\|Hess_{BFGS}\|\|$	0.161
$\|\|Hess_{FD}\|\|$	0.366
$\|\|Hess_{BFGS} - Hess_{FD}\|\|/\|\|Hess_{FD}\|\|$	0.229
$\|\|Hess_{BFGS} - Hess_{FD}\|\|$	1.42
$\|\|k_{2_{BFGS}} - k_{2_{FD}}\|\|/\|\|k_{2_{FD}}\|\|$	0.216

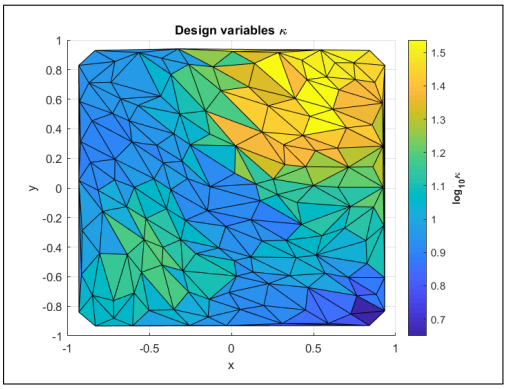
Table 6.1: Hessian approximation using BFGS and Finite Difference methods

For a case of 350 degrees of freedom, the time to compute the BFGS Hessian approximation was 4 seconds and the time to compute the Hessian approximation with finite difference was 90 seconds. The BFGS involve much shorter operations of computation. But, as can be seen in table 6.1 the BFGS approximation is not good enough. This will cause one who tries to use the BFGS algorithm to need and do more iterations in order to converge. Even then, the BFGS doesn't guarantee the same solution as an approach that approximate the Hessian with finite difference method.

To explain the reasoning behind the fact that for this test a finer mesh was used a coarse mesh test was conducted. The coarse mesh had 89 nodes and 148 elements. Figure 6.21 shows the results. The results show that in the coarse scale the solution is much more spread, it happens due to the fact that we try to minimize the first order derivative as part of the regularization. It is much easier to create a slope of decreasing values when a large number of degrees of freedom exist compared to a case where there is a limited number of degrees of freedom.



(a) Inverse Solution coarse mesh

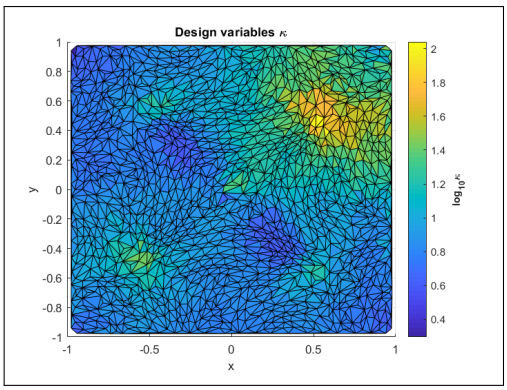


(b) Inverse Solution in a log scale coarse mesh

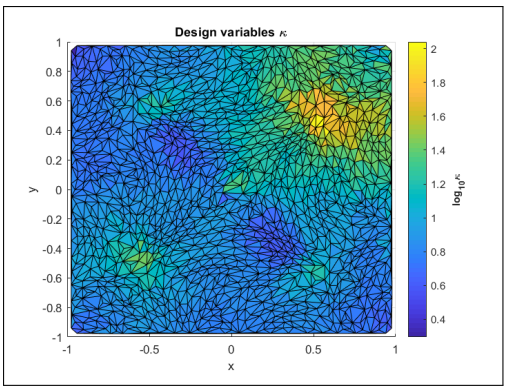
Figure 6.21: Inverse solution for the benchmark problem Hmax 0.3

6.5.4. Multi-Level on the Benchmark problem

In fig.6.22 one can see the Multi-Level and Single-Level solutions for the benchmark problem. The number of iterations on levels [1,2,3,4,5] are [5,3,3,2,2] respectively. Level 6 is the finest level and will have a max iteration stopping criteria of 30 iterations. A smaller regularization parameter of $\alpha = 1e-10$ was used compared to a larger regularization parameter as in fig. 6.19 on the finest scales. Figure 6.23 shows the difference in the number of PDE solves that are needed between the different levels. The Multi-Level methods accelerate the search for the solution.



(a) Benchmark solution for the single level



(b) Multi-Level solution for the benchmark problem

Figure 6.22: Multi-Level and Single-Level methods on the benchmark problem with low regularization parameter for the finest level of $\alpha = 1e-10$

2D cost function comparison:

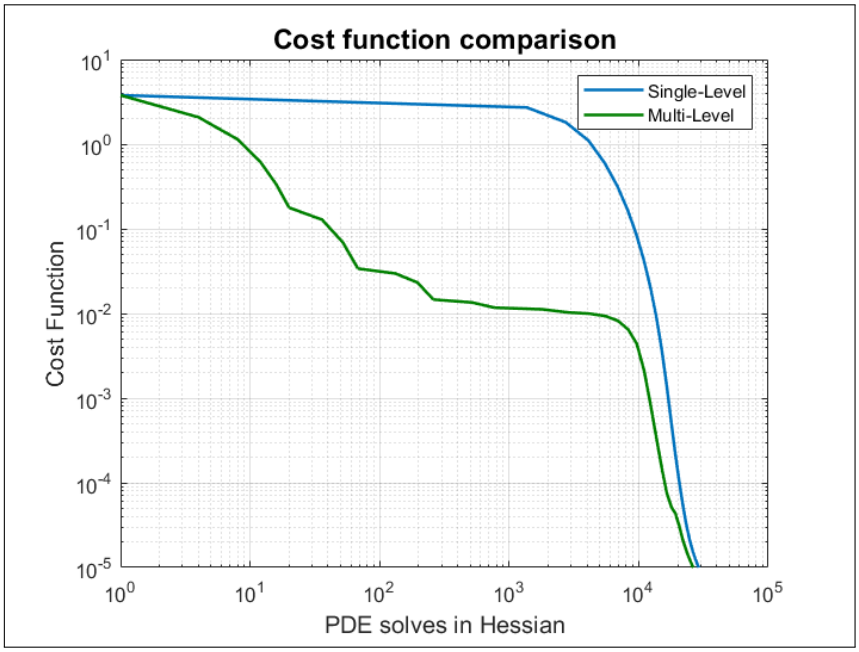
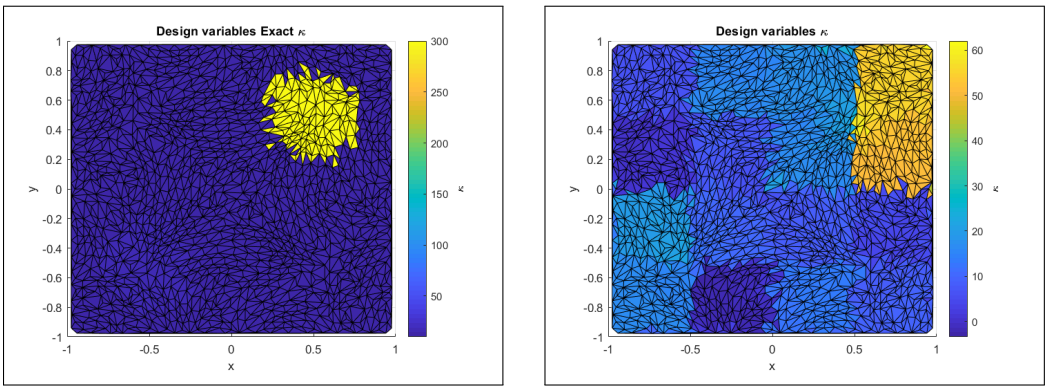


Figure 6.23: Cost function vs the amount of PDE's solve in the Hessian, comparison between the Single-Level and the Multi-Level for the 2D Benchmark Problem

Multi-Level limitations: One of the Multi-Level method limitations is presented in fig. 6.24. In the figure, the Multi-Level solution converged to a coarse level solution. The reason for this behavior is the fact that too many iterations where made on the coarser levels. Thererfore, the Trust-Region minimizer found a neighborhood of a solution which makes the cost function to decrease when moving to finer levels but still is not the minimizer that one would look for when solving this type of a problem.



(a) The benchmark exact solution

(b) Limitation of the Multi-Level

Figure 6.24: Limitations of the Multi-Level method

7

Conclusion

This thesis introduces the use of Multi-Level approaches in a combination of mesh decoupling to solve parameter estimation problems. Parameter estimation methods are large and costly to solve. The Multi-Level method allows one to reduce the degrees of freedom while searching for the solution and yet stay robust. The use of Multi-Level and upscaling methods are introduced in Moraes et al. [14] and in Echeverria et al. [5], this thesis illustrates the robustness of upscaling in the design variable space. Further, this thesis demonstrates the use of inversion to find the reservoir characteristics. The method should be used to reduce uncertainty in subsurface characterization. The Optimization uses the Newton Trust Region algorithm as a robust optimizer for this type of problem, as described by Toint et al. [17]. The thesis tests the assumptions by an example in one dimension taken from Vogel [19]. To test a more realistic case of geology and subsurface problems, a similar example to Haber [8] is used for the two-dimension case. In this example, the reservoir characteristics have a large variance to illustrate different geological bodies as part of the problem.

The conclusions from this study are:

- Multi-Level method allows one to reduce computation cost and still stay robust. While using the Multi-Level, the degrees of freedom of the problem is reduced. This allows the search to focus on the most important features in each coarse cell. While moving between the levels, the initial guess is closer to the real solution since it already includes the main components of the solution
- The Multi-Level method should be used carefully. One has to use a limited amount of iterations on a coarser level in order to converge to the desired minimum and not to converge to the coarser level minimum
- Non-Linear Least Square inversion to find the reservoir parameters is useful. The methods help identify the different bodies and differences in the bodies that lie in the subsurface. There is a lack of theoretical results regarding that type of non-linear problems. Yet, the numerical results show that the method helps remove uncertainty regarding parameter estimation
- Refining the mesh is a crucial part of being able to get a more accurate result and remove part of the sensitivity that comes from the use of the regularization. If the grid is not fine enough, the smoothing norm will smooth the variations in a large area instead of with a small area that includes many design variables
- Finding the best regularization parameter α is a costly operation. To reduce the cost, a search for the regularization parameter should be conducted on a coarse scale. As a rule of thumb, the parameter will be lower for finer meshes. The triangle method [2] is a convenient tool to find the biggest curvature in the L-Curve method
- The Trust Region algorithm is robust. It allows one to choose smaller regularization parameters and yet converge to the desired solution. The cost of using the Trust Region algorithm is negligible compared to the cost of computing the Hessian. Where the Hessian needs to be computed for the Newton and the Newton with Trust-Region methods

- Hessian approximation computation is needed to find the right solution. The curvature of the objective function is important in the search procedure of the optimization algorithm. The study on the use of the Conjugate Gradient, which is a Hessian free method, illustrated that point
- Julia language is a strong tool to write efficient code in speed while minimizing the development time of the code. Julia offers a wide range of packages that save a crucial amount of time. Julia is recommended to use in any computational science environment

Recommendation:

- Multi-Level methods should be further investigated and tested with adding adapted parameters to the inversion while iterating. Those parameters can be the selection of the region for the trust region, an adaptive mesh refinement in a location with large variations, and regularization parameter selection for the different levels
- Study on methods that try to avoid the limitation that comes from a Mulit-Level method when too many iterations are used on the coarser levels
- The use of finite differences to compute the Hessian is expensive. Hessian approximation should be studied in more detailed to reduce the cost of the Hessian approximation.
- The Inversion can be extended to more realistic problems such as transport equations. Moreover, One can add restrictions to the minimization problem by taking into account data found from the well cores. Doing so will help the algorithm to reach more accurate solutions
- Julia's use can be extended. New features such as parallelization and cloud computation can be useful in order to reduce the computing time of the costly inverse problem

Bibliography

[1] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

[2] J Longina Castellanos, Susana Gómez, and Valia Guerra. The triangle method for finding the corner of the l-curve. *Applied Numerical Mathematics*, 43(4):359–373, 2002.

[3] Guy Chavent. *Nonlinear least squares for inverse problems: theoretical foundations and step-by-step guide for applications*. Springer Science & Business Media, 2010.

[4] Andrew R Conn, Nicholas IM Gould, and Ph L Toint. *Trust region methods*, volume 1. Siam, 2000.

[5] D Echeverría and PW Hemker. Manifold mapping: a two-level optimization technique. *Computing and Visualization in Science*, 11(4-6):193–206, 2008.

[6] Tero Frondelius and Jukka Aho. JuliaFEM - open source solver for both industrial and academia usage. *Rakenteiden Mekaniikka*, 50(3):229–233, 2017. doi: 10.23998/rm.64224. URL `https://rakenteidenmekaniikka.journal.fi/article/view/64224`.

[7] Mårten Gulliksson and Per Wedin. Using the nonlinear l-curve and its dual. 09 1998.

[8] Eldad Haber. *Computational methods in geophysical electromagnetics*, volume 1. SIAM, 2014.

[9] P. Hansen. *Discrete Inverse Problems*. Society for Industrial and Applied Mathematics, 2010. doi: 10.1137/1.9780898718836. URL `https://epubs.siam.org/doi/abs/10.1137/1.9780898718836`.

[10] Per Christian Hansen. *The L-Curve and Its Use in the Numerical Treatment of Inverse Problems*, volume 4, pages 119–142. 01 2001.

[11] J. Jensen, L.W. Lake, P.W.M. Corbett, and D. Goggin. *Statistics for Petroleum Engineers and Geoscientists*. Handbook of Petroleum Explorat. Elsevier Science, 2000. ISBN 9780444505521. URL `https://books.google.nl/books?id=mHF_peTBFBIC`.

[12] Charles L Lawson and Richard J Hanson. *Solving least squares problems*, volume 15. Siam, 1995.

[13] Patrick Kofod Mogensen and Asbjørn Nilsen Riseth. Optim: A mathematical optimization package for Julia. *Journal of Open Source Software*, 3(24):615, 2018. doi: 10.21105/joss.00615.

[14] R Moraes, Hadi Hajibeygi, and Jan Dirk Jansen. A multiscale method for data assimilation. In *ECMOR XVI-16th European Conference on the Mathematics of Oil Recovery*, 2018.

[15] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006. ISBN 9780387400655. URL `https://books.google.nl/books?id=VbHYoSyelFcC`.

[16] M.J. Pyrcz and C.V. Deutsch. *Geostatistical Reservoir Modeling*. OUP USA, 2014. ISBN 9780199731442. URL `https://books.google.nl/books?id=wNhBAgAAQBAJ`.

[17] Ph Toint, Serge Gratton, Annick Sartenaer, and Philippe Toint. Numerical experience with a recursive trust-region method for multilevel nonlinear optimization. 07 2006.

[18] JJIM van Kan, A Segal, and FJ Vermolen. *Numerical methods in scientific computing*. VSSD, 2005. ISBN 90-71301-50-8.

[19] C. Vogel. *Computational Methods for Inverse Problems*. Society for Industrial and Applied Mathematics, 2002. doi: 10.1137/1.9780898717570. URL `https://epubs.siam.org/doi/abs/10.1137/1.9780898717570`.